

# **PRACTICAL MANUAL ON DATA STRUCTURE**



**Mr. Naveen Choudhary**

**Dr. Dharm Singh**

## **DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**COLLEGE OF TECHNOLOGY & ENGINEERING**  
**(Maharana Pratap University of Agriculture and Technology, Udaipur)**

# **PRACTICAL MANUAL ON DATA STRUCTURE**



**Mr. Naveen Choudhary  
Associate Professor & Head**

**Dr. Dharm Singh  
Assistant Professor**

**2008**

**DEPARTMENT OF COMPUTER SCIENCE &  
ENGINEERING**  
COLLEGE OF TECHNOLOGY & ENGINEERING  
(Maharana Pratap University of Agriculture. and Technology, Udaipur)

## PREFACE

The practical manual on “**Data Structure**” has been prepared for B.E. Computer Science & Engineering Students. The “**Data Structure**” is increasingly becoming the default choice of the IT industry especially industries involved in software development at system level.

Therefore, for proper development of “**Data Structure**” skills among the students this practical manual has been prepared. The manual contains the exercise programs and their solution for easy & quick understanding of the students. The author has gathered material from Books, Journals and Web resources.

We hope that this practical manual will be helpful for students of Computer Science & Engineering for understanding the subject from the point of view of applied aspects

There is always scope for improvement in the manual. We would appreciate to receive valuable suggestions from readers and users for future use.

Udaipur

Mr. Naveen Choudhary

Dr. Dharm Singh

## TABLE OF CONTENTS

S. No.	Laboratory Exercises	Page No.
1.	<p>Let stack_ptr be a pointer to stack of integers and item be an integer variable. Write function like Push, Pop, Initialize, Empty, and Full for doing the following tasks. [You may declare additional variable in your functions in needed]</p> <ul style="list-style-type: none"> <li>a. Return the top element of the stack and leave the top element unchanged. If the stack is empty, return INT_MAX.</li> <li>b. Return the third element from the top of the stack, provided that the stack contains at least three integers. If not, return INT_MAX. Leave the stack unchanged.</li> <li>c. Returns the bottom element of stack ( or INT_MAX if stack empty), and leave the stack unchanged.</li> <li>d. Delete all occurrences of x from the stack, leaving the other elements of the stack in the same order.</li> </ul>	11
2.	<p>Sometimes a program requires two stack containing the same type of items. If the two stacks are stored in separate arrays. Then one stack might overflow while there was considerable unused space in the other. A neat way to avoid the problem is to put all the space in one array and let one stack grow from one end of the array and the other stack start at the other end and grow in opposite direction i.e., toward the first stack, in this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all the space is actually used. Declare a new structure type <i>Double stack</i> that includes the array and the two indices <i>top A</i> and <i>top B</i>, and write functions <i>Push A</i>, <i>Push B</i>, <i>Pop A</i> and <i>Pop B</i> to handle the two stacks with in one <i>Double Stack</i>.</p>	14
3.	<p>Use the functions developed to write other functions that will</p> <ul style="list-style-type: none"> <li>a. Empty one stack onto the top of another stack</li> <li>b. Move all the items from a queue onto a stack.</li> <li>c. Start with a queue and an empty stack, and use the stack to reverse the order of all the items in the queue.</li> </ul>	17
4.	<p>Write C++ functions to implement queues by the simple method of keeping the head of the queue always in the first position of a linear</p>	21

	array.	
5.	Write C++ functions to implement queues in a circular array with one unused entry in the array. That is, we consider that the array is full when the rear is two position before the front; when the rear is one position before, it will always indicate an empty queue.	22
6.	<p>Write a function that will read one line of input from the terminal. The input is supposed to consist of two parts separated by a colon ':'. As its result your function should produce a single character as follows:</p> <ul style="list-style-type: none"> <li>N      No Colon on the Line.</li> <li>L      The Left part (before the colon) is longer than the right</li> <li>R      The Right part (after the colon) is longer than the left.</li> <li>D      The left and right parts have same length but are Different.</li> <li>S      The Left and Right parts are exactly the same.</li> </ul> <p>Use a queue to keep track of the left part of the line while reading the right part.</p>	24
7.	If we implement a queue as a circularly linked list then we need only one pointer rear (or tail) to locate both the front and rear. Write C++ functions to process a queue stored in this way. <ul style="list-style-type: none"> <li>a.      Initialize</li> <li>b.      Add Node</li> <li>c.      Delete Node</li> </ul>	27
8.	<p>Do the following operations on a given list.</p> <ul style="list-style-type: none"> <li>a.      Write a function that deletes the last entry of a list.</li> <li>b.      Write a function that deletes the first entry of a list</li> <li>c.      Write a function that reverse the order of the entries in a list.</li> <li>d.      Write a function that splits a list into other lists, so that the entries that were in odd-numbered positions are now in one list (in the same relative order as before) and those from even-numbered positions are in the other new list.</li> </ul> <p>The word deque (Pronounced either “deck” or “DQ”) is a shortened from a double- ended queue and denotes a list in which items can be</p>	30

	added or deleted from either the first or the last position of the list, but no changes can be made elsewhere in the list. Thus a deque is a generalization of both a stack and queue.	
9.	Write a function that will reverse a linked list while traversing it only once. At the conclusion, each node should point to the node that was previously its predecessor; the head should point to the node that was formerly at the end, and the node that was formerly first should have a NULL link.	33
10.	Write an algorithm that will split a linked list into two linked lists, so that successive nodes go to different lists.(the first, third, and all odd numbered nodes to the first list, and the second, fourth, and all even-numbered nodes go to the second.)	36
11.	Write a function that will concatenate two circularly linked lists, producing one circularly linked list.	39
12.	Write a function that will split a circularly linked list, into two circularly linked lists.	42
13.	Write a function for manipulating a doubly linked list as follows:  a. add a node after p. b. Add a node before p. c. Delete node p. d. traverse the list.	45
14.	Write and test a recursive function that return the maximum among the first n elements of an array.	50
15.	Write the following function template:  Template <class T> void reverse(stack<T>&);  // reverse the contents of the given stack;	51
16.	Write the following function template:  T& bottom (); // returns the bottom element	53
17.	Use the stack to implement the following function:  Bool is palindrome (string s);	55
18.	Write the following overloaded equality operator for queues:  Bool queue::operator +=(const queue&)	57
19.	Implement the following friend function for the Integer class:  friend istream& operator>> (istream & istr, Integer& x);	60

	// Input x from the istr input stream.	
20.	Write the following function that uses a binary search tree to sort an array a of n elements :  void sort (type* a, int n);	61
21.	Implement the following Tree class member function:  Bool empty() const;  // returns true iff this tree has no elements.	65
22.	Implement the following Tree class member function:  int size() const;  // returns the number of elements in this tree.	68
23.	Implement the following member functions for the Tree class :  int width ( int n);  // returns the total number of nodes at level n in this tree.	71
24.	Implement the following member function for the Tree class:  void defoliate();  // removes all the leaves from this tree.	74
25.	There are 10 records present in a file with the following structures  struct { char itemcode[6] ; char itemname[20]; int qty; };  Write a program to read these records and arrange them in ascending order and write them in a target file.	77
26.	Write a class that implements a circular queue as a linked list.	80
27.	Write a class that implements a Bubble Sorting algorithm on a set of 25 numbers.	82
28.	A class has two member functions as Encript() and Decript(). Implement this class to encrypt a string and decrypt it using your own method for encryption and decryption.	84
29.	Write a program for linked list, which will have the facility to add, modify and delete the records from linked list. The fields to be accepted	86

	<p>from the user include</p> <table style="margin-left: 40px;"> <tr><td>Employee No</td><td>Numeric</td></tr> <tr><td>Name</td><td>String</td></tr> <tr><td>Department</td><td>String</td></tr> </table> <p>Option should be made to view the list generated. Modularize the code to perform the above actions.</p>	Employee No	Numeric	Name	String	Department	String	
Employee No	Numeric							
Name	String							
Department	String							
30.	Write a class that has the functionality to write and read the data from a file line by line.	91						
31.	Write a program that performs the matrix manipulations such as addition and deletion. The matrix class will have the member functions Create, Add, Delete, Display.	93						
32.	Write a program that will print the largest and smallest number from and 5 row by 5 column matrix. The matrix class will have the member functions as Create, Display, Largest, Smallest.	97						
33.	Write a template class for sorting method. Using this class, write a test program for sorting using different data types.	100						
34.	Write a program that will convert the given decimal input in to other number systems Hexadecimal, Octal, and Binary. The class Conversion will have the methods as GetNumber, Print Number, Convert To Hexadecimal, Convert To Octal, Convert To Binary.	101						
35.	Write a class Time that will have the member functions to calculate the difference and addition for given two time values. It will store the time in hours(0-31), minutes (0-59), and seconds (0-59).	105						
36.	Write a class to handle fractions such as “1/3”. Define Addition, Subtraction, Multiplication, and Division operators for these fractions. (Hint: use operator overloading).	108						
37.	Write a Date class that allows you to add, subtract, read and print simple dates of the form dd/mm/yy. Considering leap year calculations will be an added advantage.	112						
38.	<p>Define a “string_match” base class .</p> <pre style="margin-left: 40px;">class string_matcher { public :</pre>	115						

	<pre>// Returns TRUE if the string matches, false if not.  int match(const char string);  ..... }</pre> <p>Define the derived classes that match words, numbers and strings.</p>	
39.	Write a base Class Number that holds a single integer value and contains one member function (print it). Define three derived classes to print the value in hex, octal, and binary.	116
40.	Write a template min that returns the minimum of two values. Make sure you handle the strings correctly.	118
41.	Write a program to reverse a given string using stack technique.	119
42.	<p>Write a class called Clock that simulates the keeping of time. Use three private class members:</p> <p>hours, minutes, and seconds. Your class should be able to :</p> <p>Set() that starting time.</p> <p>Increment() the time by one second.</p> <p>Display() the time.</p> <p>The function should take an argument with a default value of zero to imply military time. If this value is something other than zero, display the time in standard Am and PM notation. For example, 4 minutes and 31 seconds past 7 PM should be displayed either 19:04:31 or 7:04:31 PM and 5 minutes past midnight should be displayed as either 00:05:</p>	120
43.	<p>Write a class called Date that keeps track of current date. Your class should be able to :</p> <p>Set() the starting date.</p> <p>Increment() the day by 1. If no. of days overflows then corresponding changes in month and year should be considered.</p> <p>Display() the date in MM/DD/YY format.</p>	123
44.	<p>Write a program that reads in string input from the user, reverse the case of the letters, and then echoes the string back to the user. The class Reverse Case contains a character buffer up to 80 bytes in length as the member variable and Read(), Convert() and Print() as member functions.</p> <p>Define Read and print functions as inline functions.</p>	126
45.	Create base class media. This class has two fields one for title and the other for price. Derive two specific classes called tape and book, tape	128

	has a field called length and book has got one field called pages. Add two member functions to base class, one get_data() to initialize base class data members and second display_data() which is virtual to display values. Override display_data() function in both derived classes to display data specific to them.	
46.	An election is contested by five candidates. The candidates are numbered from 1 to 5 and voting is done by accepting the candidate number from voter. Write a program which will accept the votes and count votes for each candidate. If the number typed by voter is outside the range the vote is discarded and the program will keep track of the number of such discarded votes.	131
47.	Implement circular link list. Include the following functions ; a. Adding nodes to the list. b. Traversing the whole list c. Deleting a node form the list.	133
48-49	Write a menu-driven program which will accept an array of 10 integer values and sort them with any two sorting algorithms of your choice.	136

Q.1 Let stack\_ptr be a pointer to stack of integers and item be an integer variable. Write function like Push, Pop, Initialize, Empty, and Full for doing the following tasks. [ You may declare additional variable in your functions if needed ]

- e. Return the top element of the stack and leave the top element unchanged. If the stack is empty, return INT\_MAX.
- f. Return the third element from the top of the stack, provided that the stack contains at least three integers. If not, return INT\_MAX. Leave the stack unchanged.
- g. Returns the bottom element of stack ( or INT\_MAX if stack empty),and leave the stack unchanged.
- h. Delete all occurrences of x from the stack, leaving the other elements of the stack in the same order.

//Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define        INT_MAX      999
#define        MAX          10
class cStack
{
    int data[ MAX ];
    int top;
    int Empty( );
    int Full( );
public:
    cStack( )
    {
        top = 0;
    }
    void Push( );
    void Push( int );
    int Pop( );
    void DispStack( );
    int Peek( );
    int Third( );
    int Bottom( );
    void Delete( int );
    void Delete( );
}
int cStack::Empty( )
{
    if( top < 0 )
        return 1;
    return 0;
}
int cStack::Full( )
{
    if( top > MAX )
        return 1;
    return 0;
}
int cStack::Peek( )
{
    if( !Empty( ) )
        return data[ top ];
    return INT_MAX;
}
int cStack::Third( )
{
```

```

        if( top >= 2 )
            return data[ top - 2 ];
        return INT_MAX;
    }
int cStack::Bottom( )
{
    if( !Empty( ) )
        return data[ 1 ];
    return INT_MAX;
}
void cStack::Delete( int x )
{
    for( int i = 0; i <= top; i++ ) // < >
    {
        if( data[ i ] == x )
        {
            for( int j = i; j <= top; j++ ) // < >
                data[ j ] = data[ j + 1 ];
            i--;
            top--;
        }
    }
}
void cStack::Delete( )
{
    int x;
    cout << "Element to be Deleted :";
    cin >> x;
    for( int i = 0; i <= top; i++ ) // < >
    {
        if( data[ i ] == x )
        {
            top--;
            for( int j = i; j <= top; j++ ) // < >
                data[ j ] = data[ j + 1 ];
            i--;
        }
    }
    top--;
}
void cStack :: Push()
int x;
if( !Full( ) )
    while( 1 )
    {
        cout << "Data :";
        cin >> x;
        if( x == 0 )
            break;
        else
            data[ ++top ] = x;
        else
            cout << "Stack Overflow";
    }
void cStack::Push( int x )
{
    if( !Full( ) )
        data[ ++top ] = x;
    else
        cout << "Stack Overflow";
}

```

```

}

int cStack::Pop( )
{
    if( !Empty( ) )
        return ( data[ top-- ] );
    else
        cout << "Stack UnderFlow";
}
void cStack::DispStack( )
{
    for( int i = top; i > 0; i-- ) // < >
        cout << data[ i ] << endl;
}
void main( )
{
    cStack      S1;
    clrscr( );
        /*S1. Push (25);
         S1.Push(2);
         S1.Push(10);
         S1. Push (55);
         S1.Push(20);*/
    S1.Push( );
    clrscr( );
    cout << "stack is" << endl;
    S1.DispStack( );
        /* int x=S1.Third();
           cout<<"III element = "<<x<<endl;
           x=S1.Peek();
           cout<<"I elemn = "<<x<<endl;
           x=S1.Bottom();
           cout<<"Last ele = "<<x<<endl; */
    S1.Delete( 1 );
    cout << "Now Stack" << endl;
    S1.DispStack( );
    getch( );
}
// INPUT :
/*Data:
2
Data:
3
Data:
4
Data:
5
Data:
0
// OUTPUT :
stack is
5
4
3
2
Now Stack
5
4
3
2

```

Q.2 Sometimes a program requires two stack containing the same type of items. If the two stacks are stored in separate arrays. Then one stack might overflow while there was considerable unused space in the other. A neat way to avoid the problem is to put all the space in one array and let one stack grow from one end of the array and the other stack start at the other end and grow in opposite direction, i.e., toward the first stack, in this way, if one stack turns out to be large and the other small, then they will still both fit, and there will be no overflow until all the space is actually used. Declare a new structure type *Double stack* that includes the array and the two indices *topA* and *topB*, and write functions *PushA*, *Push B*, *PopA* and *PopB* to handle the two stacks within one *DoubleStack*.

```
// Example Code :

#include      <iostream.h>
#include      <stdio.h>
#include      <conio.h>
#include      <string.h>
#define       size      5
struct doublestack
{
    int stack[ size ];
    int TopA,
        TopB;
} ;
void pushA( struct      doublestack & s, int  d )
{
    s.stack[ ++( s.TopA ) ] = d;
}
void pushB( struct      doublestack & s, int  d )
{
    s.stack[ --( s.TopB ) ] = d;
}
int popA( struct  doublestack & s )
{
    int popped_element      = s.stack[ ( s.TopA ) ];
    ( s.TopA )--;
    return ( popped_element );
}
int popB( struct  doublestack & s )
{
    int popped_element;
    popped_element = s.stack[ ( s.TopB ) ];
    ( s.TopB )++;
    return ( popped_element );
}
int empty( struct  doublestack & s )
{
    if( ( s.TopA == -1 ) && ( s.TopB == size - 1 ) )
        return 1;
    return 0;
}
int full( struct  doublestack & s )
{
    if( ( s.TopA + 1 == s.TopB ) || ( s.TopA >= size ) // (stack-
ptr==s+size-1)
        || ( s.TopB <= 0 ) )
        return 1;
    return 0;
}
void display( struct  doublestack & s )
{
    if( empty( s ) )

```

```

{
    cout << "Stack is empty";
}
else
{
    cout << "\nStack B is";
    for( int i = s.TopB; i <= size - 1; ++i ) //      < >

        cout << "    " << s.stack[ i ];
    cout << "\nStack A is";
    for( i = s.TopA; i >= 0; --i )
        cout << "    " s.stack [i] ;
}
}

void main( )
{
    struct doublestack twostack;
    int item;
    char choice;
    int q = 0;
    twostack.TopA = -1;
    twostack.TopB = size;
    clrscr( );
    do
    {
        cout << "\nPush to stack A->1";
        cout << "\nPush to stack B->2";
        cout << "\nPop to stack A->3";
        cout << "\nPop to stack B->4";
        cout << "\nTo Quit ->5";
        cout << "\nInput the choice .     ";
        do
        {
            cin >> choice;
        }
        while (strchr (11 12345 ",choice) ==NULL);
        switch( choice )
        {
            case '1':
                cout << "\n Input the element to pushed:";
                cin >> item;
                if( !full( twostack ) )
                {
                    pushA( twostack, item );
                    cout << "\n After inserting into stack A";
                }
                else
                    cout << "\n Stack is now full";
                break;
            case '2':
                cout << "\n Input the element to pushed:";
                cin >> item;
                if( !full( twostack ) )
                {
                    pushB( twostack, item );
                    cout << "\n After inserting into stack B";
                }
                else
                    cout << "\n Stack is now full";
                break;
            case '3':
                if( !empty( twostack ) )
                {
                    item = popA( twostack );

```

```

        cout << "\n Data -is popped: " << item;
        cout << "\n Rest data in stack is as
follows:\n";
    }
    else
        cout << "\n Stack underflow";
    break;
case '4':
    if( !empty( twostack ) )
    {
        item = popB( twostack );
        cout << "\n Data is popped: " << item;
        cout << "\n Rest data in stack is as
follows:\n";
    }
    else
        cout << "\n Stack underflow";
    break;
case '5':
    q = 1;
}
display( twostack );
} while( !q );

```

// INPUT & OUTPUT :

Push to stack A->1

Push to stack	B->2
Pop to stack	A->3

Pop to stack	B->4
--------------	------

To Quit ->5	
-------------	--

Input the choice	.
------------------	---

1

Input the element to pushed:2

After inserting into stack A

Stack B is

Stack A is	2
------------	---

Push to stack	A->1
---------------	------

Push to stack	B->2
---------------	------

Pop to stack	A->3
--------------	------

Pop to stack	B->4
--------------	------

To Quit ->5	
-------------	--

Input the choice	.
------------------	---

2

Input the element to pushed:8

After inserting into stack BStack is empty

Push to stack	A->1
---------------	------

Push to stack	B->2
---------------	------

Pop to stack	A->3
--------------	------

Pop to stack	B->4
--------------	------

To Quit ->5	
-------------	--

Input the choice	.
------------------	---

3

Stack underflowStack is empty

Push to stack	A->1
---------------	------

Push to stack	B->2
---------------	------

Pop to stack	A->3
--------------	------

Pop to stack	B->4
--------------	------

To Quit ->5	
-------------	--

Input the choice	.
------------------	---

- Q.3 Use the functions developed to write other functions that will
- d. Empty one stack onto the top of another stack
  - e. Move all the items from a queue onto a stack.
  - f. Start with a queue and an empty stack, and use the stack to reverse the order of all the items in the queue.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define       MAX    10
class      cQueue;
class      cStack
{
    int data[ MAX ];
    int top;
    int Full( );
    int Empty( );
public:
    cStack( )
    {
        top = 0;
    }
    void Push( int );
    int Pop( );
    void DispStack( );
    void Transfer( cStack & );
    friend   void Que_to_Stk( cStack &, cQueue & );
    friend   void Que_Rev( cStack &, cQueue & );
} ;
class      cQueue
{
    int items[ MAX ];
    int front,
        rear,
        count;
    int Full( );
    int Empty( );
public:
    cQueue( )
    {
        rear = front = count = 0;
    }
    void Add( int );
    void DispQueue( );
    int Del( );
    friend   void Que_to_Stk( cStack &, cQueue & );
    friend   void Que_Rev( cStack &, cQueue & );
} ;
int cStack::Empty( )
{
    if( top < 0 )
        return 1;
    return 0;
}
int cStack::Full( )
{
    if( top > MAX )
        return 1;
    return 0;
}
void cStack::Push( int x )
```

```

{
    if( !Full( ) )
        data[ ++top ] = x;
    else
        cout << "Stack Overflow";
}
int cStack::Pop( )
{
    if( !Empty( ) )
        return ( data[ top-- ] );
    else
        cout << "Stack UnderFlow";
}
void cStack::DispStack( )
{
    for( int i = top; i > 0; i-- )                                // < >
        cout << data[ i ] << "\t";
    cout << endl;
}
int cQueue::Full( )
{
    if( count >= MAX )
        return 1;
    return 0;
}
int cQueue::Empty( )
{
    if( count == 0 )
        return 1;
    return 0;
}
void cQueue::Add( int x )
{
    if( !Full( ) )
    {
        count++;
        items[ rear++ ] = x;
    }
    else
        cout << "Queue Full..... Add Aborted!" << endl;
}
void cQueue::DispQueue( )
{
    for( int i = front; i < count; i++ )                                // < >
        cout << items[ i ] << "\t";
    cout << endl;
}
int cQueue::Del( )
{
    int x = -999;
    if( !Empty( ) )
    {
        x = items[ front ];
        for( int j = front; j < rear; j++ )                                // < >

            items[ j ] = items[ j + 1 ];
        count--;
    }
    else
        cout << "Queue is Empty ..... Delete Aborted!" << endl;
    return x;
}

```

```

void Que_to_Stk( cStack & S, cQueue & Q )
{
    if( !( Q.Empty( ) ) )
    {
        while( Q.count > 0 )
            S.Push( Q.Del( ) );
    }
    else
        cout << "Queue Empty ..... Aborting! " << endl;
}
void Que_Rev( cStack & S, cQueue & Q )
{
    if( !( Q.Empty( ) ) )
    {
        while( Q.count > 0 )
            S.Push( Q.Del( ) );
        Q.rear = 0;
        while( S.top > 0 )
            Q.Add( S.Pop( ) );
    }
    else
        cout << "Queue Empty ..... Aborting! " << endl;
}
void cStack::Transfer( cStack & S2 )
{
    if( !( Empty( ) ) )
    {
        while( top >= 0 )
            S2.Push( Pop( ) );
    }
    else
        cout << "First Stack empty ..... Aborting! " << endl;
}
void main( )
{
    cStack      S,
                S1;
    cQueue      Q;
    Q.Add( 23 );
    Q.Add( 10 );
    Q.Add( 5 );
    Q.Add( 56 );
    clrscr();
    Q.DispQueue( );

/*     S1.Push(14);
    S1.Push(9) ;
    S1.Push(3);
    S1.Push(2);
    clrscr(); */
//    Que_to_Stk(S,Q);
//    cout<<"Copied Stack" << endl;
//    S1.Dispstack();
//    cout<< "Second Stack" << endl;
//    S1.Transfer(S); */
    Que_Rev( S, Q );
    cout << endl;
    Q.DispQueue( );
    getch( );
}

// INPUT & OUTPUT :
23          10          5          56
IN STACK REVERSE THE ORDER OF ALL THE ITEMS IN THE QUEUE
56          5           10          23

```

Q.4 Write C++ functions to implement queues by the simple method of keeping the head of the queue always in the first position of a linear array.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define        MAX    10
class cQueue
{
    int items[ MAX ];
    int front,
        rear,
        count;
    int Full( );
    int Empty( );
public:
    cQueue( )
    {
        rear = front = count = 0;
    }
    void Add( );
    void Add( int );
    void Disp( );
    int Del( );
    int Peek( );
}
int cQueue::Full( )
{
    if( count >= MAX )
        return 1;
    return 0;
}
int cQueue::Empty( )
{
    if( count == 0 )
        return 1;
    return 0;
}
void cQueue::Add( )
{
    int x;
    if( !Full( ) )
    {
        count++;
        cout << "Data:" ;
        cin >> x;
        items[ rear++ ] = x;
    }
    else
        cout << "Queue Full..... Add Aborted! " << endl;
}
void cQueue::Add( int x )
{
    if( !Full( ) )
    {
        count++;
        items[ rear++ ] = x;
    }
    else
        cout << "Queue Full..._ Add Aborted!" << endl;
}
```

```

int cQueue::Del( )
{
    int x = -999;
    if( !Empty( ) )
    {
        x = items[ front ];
        for( int j = front; j < rear; j++ )           // < >

            items[ j ] = items[ j + 1 ];
        count--;
    }
    else
        cout << "Queue is Empty ..... Delete Aborted!" << endl;
    return x;
}
int cQueue::Peek( )
{
    if( !Empty( ) )
        return items[ front ];
    else
        cout << "queue is Empty ..... Delete Aborted!" << endl;
    return -999;
}
void cQueue::Disp( )
{
    for( int i = front; i < count; i++ )           // < >

        cout << items[ i ] << endl;
}
void main( )
{
    cQueue Q;
    clrscr( );
// Q.Add();
    Q.Add( 45 );
    Q.Add( 67 );
    Q.Add( 56 );
    Q.Add( 10 );
    Q.Add( 45 );
    Q.Add( 78 );
    clrscr( );
    Q.Disp( );
    int x = Q.Del( );
// cout<<endl<<x;
    x = Q.Del( );
// cout<<endl<<x;
    cout << endl << "Queue after deletion" << endl;
    Q.Disp( );
    x = Q.Peek( );
// cout<<endl<<x;
    getch( );
}
// INPUT & OUTPUT :
45
67
56
10
45
78
Queue after deletion
56
10
45
78

```

Q.5 Write C++ functions to implement queues in a circular array with one unused entry in the array. That is, we consider that the array is full when the rear is two position before the front; when the rear is one position before, it will always indicate an empty queue.

// Example Code:

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define       MAX      5
class cCirQueue
{
    int items[ MAX ];
    int front,
        rear;
    int Empty( );
    int Full( );
public:
    cCirQueue( )
    {
        front = 0;
        rear = -1;
    }
    void Add( );
    void Add( int );
    void Disp( );
}
int cCirQueue::Empty( )
{
    if( MAX - rear == 1 )
        return 1;
    return 0;
}
int cCirQueue::Full( )
{
    if( MAX - rear == 2 )
        return 1;
    return 0;
}
void cCirQueue::Add( )
{
    while( 1 )
    {
        int x;
        cout << "Item (0-Exit) : ";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
            if( !Full( ) )
            {
                rear          = ( rear + 1 ) % MAX;
                items[ rear ] = x;
            }
            else
            {
                cout << "Queue Full my Dear ..... Aborting" <<
endl;
                break;
            }
        }
    }
}
```

```

        }
    }
void cCirQueue::Disp( )
{
    for( int i = front; i <= rear; i++ )           //  < >
        cout << items[ i ] << endl;
}
void main( )
{
    cCirQueue Q;
    clrscr();
    Q.Add();
    Q.Disp();
    getch();
}

// INPUT & OUTPUT :
Item (0-Exit) :1
Item (0-Exit) :2
Item (0-Exit) :3
Item (0-Exit) :4
Item (0-Exit) :5
Queue Full my Dear ..... Aborting
1
2
3
4

```

Q.6 Write a function that will read one line of input from the terminal. The input is supposed to consist of two parts separated by a colon'::'. As its result your function should produce a single character as follows:

N	No Colon on the Line.
L	The Left part (before the colon) is longer than the right.
R	The Right part (after the colon) is longer than the left.
D	The left and right parts have same length but are different.
S	The Left and Right parts are exactly the same

Use a queue to keep track of the left part of the line while reading the right part

// Example Code :

```
#include    <iostream.h>
#include    <iomanip.h>
#include    <conio.h>
#include    <string.h>
#define      MAX   50
#include    <stdio.h>
class cQueue
{
    char items[ MAX ];
    int   front,
          rear,
          count;
    int Full( );
    int Empty( );
public:
    cQueue( )
    {
        rear = front = count = 0;
    }
        void Add( char );
        void Disp( );
        char Del( );
    friend    char Read( cQueue & );
} ;
int cQueue::Full( )
{
    if( count >= MAX )
        return 1;
    return 0;
}
int cQueue::Empty( )
{
    if( count == 0 )
        return 1;
    return 0;
}
void cQueue::Add( char  x )
{
    if( !Full( ) )
    {
        count++;
        items[ rear++ ] = x;
    }
    else
        cout << "Queue Full..... Add Aborted!" << endl;
}
char cQueue::Del( )
{
    char x      = '!';
    if( Empty( ) )
        cout << "Queue Empty..... Del Aborted!" << endl;
    else
        x = items[ front++ ];
    return x;
}
```

```

if( !Empty( ) )
{
    x = items[ front ];
    for( int j = front; j < rear; j++ ) // < >
        items[ j ] = items[ j + 1 ];
    count--;
}
else
    cout << "Queue is: Empty ..... Delete Aborted ! " << endl;
return x;
}
void cQueue::Disp( )
{
    for( int i = front; i < count; i++ ) // < >
        cout << items[ i ] << endl;
}
char Read( cQueue & Q )
{
    char * str;
    int len;
    cout << "Length of String is :";
    cin >> len;
    str = new char [len + 1];
    cout << "Enter any string .   ";
    gets( str );
    fflush( stdin );
    int idx;
    int flag = 1;
    char ch;
    for( int i = 0; // < >

        str[ i ] != ' ' && str[ i ] != ':'
        && str[ i ] != '\t'; i++ )
        Q.Add( str[ i ] );
    idx = i;
    if( str[ idx ] == ' ' || str[ idx ] == '\t' )
        ch = 'N';
    else
    {
        for( ++i; str[ i ]; i++ )
        {
            if( str[ i ] != Q.Del( ) )
                flag = 0;
            break;
        }
    }
    //Copying left part to Queue again
    Q.count = Q.rear = 0;
    for( int i = 0; // < >

        str[ i ] != ' ' && str[ i ] != ':'
        && str[ i ] != '\t'; i++ )
        Q.Add( str[ i ] );
    if( flag ) // //
    if left is exactly same as

    //      right
    ch = 'S';
    else
    {
        for( ++i; str[ i ]; i++ )
            if( ( i - idx - 1 ) == Q.count )
                ch = 'D';
}

```

```
        }
    }
    if( i - idx - 1 > Q.count )
        ch = 'R';
    else
        ch = 'L';
    return ch;
}
void main( )
{
    cQueue      Q;

    clrscr( );
    char ch   = Read( Q );
    clrscr( );
    Q.Disp( );
    cout << endl << "Result : " << ch;
    getch( );
}
```

Q.7 If we implement a queue as a circularly linked list then we need only one pointer rear (or tail) to locate both the front and rear. Write C++ functions to process a queue stored in this way.

- b. Initialize
- c. AddNode
- d. DeleteNode

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cQueue;
class cNode
{
    char     name[ 15 ];
    int      age;
    float    basic;
    cNode   * next;
public:
    cNode( )
    {
        strcpy( name, '\0' );
        age   = 0;
        basic = 0.0;
        next  = '\0';
    }
    cNode( char *, int, float );
    void DispNode( );
    friend      class  cQueue;
}      ;
class cQueue
{
    cNode   * rear;
public:
    cQueue( )
    {
        rear     = '\0';
        rear->next = '\0';
    }
    void Add( );
    void Delete( );
    void DispQueue( );
}
cNode::cNode( char * n, int  a, float  b )
{
    strcpy( name, n );
    age   = a;
    basic = b;
    next  = '\0';
}
void cNode::DispNode( )
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 15 ) << name;
    cout.setf( ios::right, ios::adjustfield );
    cout << setw( 3 ) << age << setw( 6 ) << basic << endl;
}
void cQueue::Add( )
{
    char  nm[ 15 ];
```

```

int    a;
float b;
while( 1 )
{
    cout << "Name      (Quit):      ";
    cin >> nm;
    if( strcmp( nm, "Quit" ) == 0 )
        break;
    else
    {
        cout << "Age :";
        cin >> a;
        cout << "Basic :";
        cin >> b;
        cNode * NewNode = new cNode( nm, a, b );
        if( !( rear->next ) )
        {
            rear      = NewNode;
            rear->next = NewNode;
        }
        else
        {
            NewNode->next = rear->next;
            rear->next   = NewNode;
            rear          = NewNode;
        }
    }
}
void cQueue::Delete( )
{
    cNode * ptr = rear->next;
    rear->next = rear->next->next;
    delete ptr;
}
void cQueue::DispQueue( )
{
    int flag = 0;
    for( cNode * ptr = rear->next; // < >

        !( flag && ptr == rear->next );
        ptr = ptr->next, flag = 1 )
        ptr->DispNode( );
}
void main( )
{
    cQueue      L;
    clrscr( );
    L.Add( );
    clrscr( );
    L.DispQueue( );
    cout << endl;
    L.Delete( );

    L.DispQueue( );
    cout << queue after deletion;
    getch( );
}

// INPUT :

Name (Quit): enter
Age:

```

```
21
Basic:
34
Name (Quit): xyz
Age:
23
Basic:
56
Name (Quit): Quit
```

```
// OUTPUT :
enter      21      34
xyz        23      56
```

```
QUEUE AFTER DELETE OPERATION
```

```
xyz      23      56
```

Q.8 Do the following operations on a given list.

- e. Write a function that deletes the last entry of a list.
- f. Write a function that deletes the first entry of a list
- g. Write a function that reverse the order of the entries in a list.
- h. Write a function that splits a list into other lists, so that the entries that were in odd-numbered positions are now in one list (in the same relative order as before) and those from even-numbered positions are in the other new list.

The word deque (Pronounced either "deck" or "DQ") is a shortened from a double- ended queue and denotes a list in which items can be added or deleted from either the first or the last position of the list, but no changes can be made elsewhere in the list. Thus a deque is a generalization of both a stack and queue.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define       MAX    10
class cDeque
{
    int items[ MAX ];
    int front,
        rear;
    int Empty( );
    int Full( );
public:
    cDeque( )
    {
        rear = front = 0;
    }
    void AddRear( int );
    void AddFront( int );
    int DelFront( );
    int DelRear( );
    void Reverse( );
    void Disp( );
    void Split( cDeque &, cDeque & );
}
int cDeque::Full( )
{
    if( rear > MAX )
        return 1;
    return 0;
}
int cDeque::Empty( )
{
    if( front == rear )
        return 1;
    return 0;
}
void cDeque::AddRear( int x )
{
    if( !Full( ) )
        items[ rear++ ] = x;
    else
        cout << "Queue Full      ..... Aborting!" << endl;
}
void cDeque::AddFront( int x )
```

```

{
    if( !Full( ) )
    {
        for( int i = rear; i >= front; i-- )           // < >
            items[ i ] = items[ i - 1 ];
        items[ front ] = x;

        rear++;
    }
    else
        cout << "Queue Full ..... Aborting!" << endl;
}
int cDeque::DelFront( )
{
    int x;
    if( !Empty( ) )
    {
        x = items[ front ];
        for( int i = front; i < rear; i++ )           // < >
            items[ i ] = items[ i + 1 ];
        rear--;
    }
    else
    {
        cout << "Queue Empty ..... Aborting!" << endl;
        x = -9999;
    }
    return x;
}
int cDeque::DelRear( )
{
    int x;
    if( !Empty( ) )
    {
        x = items[ rear - 1 ];
        rear--;
    }
    else
    {
        cout << "Queue Empty ..... Aborting!" << endl;
        x = -9999;
    }
    return x;
}

void cDeque::Split( cDeque & Q1, cDeque & Q2 )
{
    for( int i = front; i < rear; i++ )           // < >
    {
        if( ( i + 1 ) % 2 )
            Q1.AddRear( items[ i ] );
        else
            Q2.AddRear( items[ i ] );
    }
}
void cDeque::Disp( )
{
    for( int i = front; i < rear; i++ )           // < >
        cout << items[ i ] << endl;
}

```

```

}
void main( )
{
    cDeque      D,D1,D2;
    clrscr( );
    D.AddRear( 1 );
    D.AddRear( 2 );
    D.AddRear( 3 );
    D.AddRear( 4 );
    D.AddRear( 5 );
    D.AddRear( 6 );
    D.Disp( );
    cout << endl;
    D.Split( D1, D2 );
    cout << "List 1" << endl;
    D1.Disp( );
    cout << "List 2" << endl;
    D2.Disp( );
/*cout<<D.DelFront();
cout<<endl<<endl;
D.Disp();
D.DelRear();
D.Disp();*/
    getch( );
}

```

// Input & Output :

```

1
2
3
4
5
6

```

List 1

```

1
3
5
List 2
2
4
6

```

Q.9 Write a function that will reverse a linked list while traversing it only once. At the conclusion, each node should point to the node that was previously its predecessor; the head should point to the node that was formerly at the end, and the node that was formerly first should have a NULL link.

//Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cNode
{
    int        x;
    cNode *   next;
public:
    void      MakeNode( int );
    void      PutNext( cNode * );
    cNode *   GetNext( );
    void      DispNode( );
} ;
class cList
{
    cNode *   start,
              * last;
    int       TotNodes;
public:
    cList( );
    void Add( );
    void Add( int );
    void DispList( );
    void Reverse( );
} ;
void cNode::MakeNode( int a )
{
    x      = a;
    next = '\0';
}
void cNode::PutNext( cNode * p )
{
    next = p;
}
cNode * cNode::GetNext( )
{
    return next;
}
void cNode::DispNode( )
{
    cout << x << endl;
}
cList::cList( )
{
    start = last = '\0';
    TotNodes = 0;
}
void cList::Add( )
{
    int a;
    while( 1 )
    {
        cout << "Data      (0 to exit) :";
        cin >> a;
        if( a == 0 )
            break;
        cNode * p = new cNode;
        p->x = a;
        if( start == '\0' )
            start = p;
        else
            last->next = p;
        last = p;
    }
}
```

```

        if( a == 0 )
            break;
        else
        {
            cNode * newnode = new cNode;
            newnode->MakeNode( a );
            if( !start )
                start = last = newnode;
            else
            {
                last->PutNext( newnode );
                last = newnode;
            }
        }
    }
void cList::Add( int a )
{
    cNode * newnode = new cNode;
    newnode->MakeNode( a );
    if( !start )
        start = last = newnode;
    else
    {
        last->PutNext( newnode );
        last = newnode;
    }
}
void cList::Reverse( )
{
    cNode * ptr,
           * tmp;
    tmp = start->GetNext( );
    start->PutNext( '\0' );
    for( ptr = tmp; ptr; ptr = tmp )
    {
        tmp = ptr->GetNext( );
        ptr->PutNext( start );
        start = ptr;
    }
}
void cList::DispList( )
{
    for( cNode * ptr = start; ptr;                  // < >
          ptr = ptr->GetNext( ) )
        ptr->DispNode( );
}
void main( )
{
    cList L;
    clrscr( );
    L.Add( );
    L.DispList( );
    L.Reverse( );
    L.DispList( );
    getch( );
}

// INPUT :
/*Data      ! 0 to exit):1
Data      ! 0 to exit):2
Data      ! 0 to exit):3

```

```
Data ! 0 to exit):4
Data ! 0 to exit):5
Data ! 0 to exit):6
Data ! 0 to exit):7
Data ! 0 to exit):8
Data ! 0 to exit):9
Data ! 0 to exit):10
Data ! 0 to exit):0
```

```
// OUTPUT :
```

```
1
2
3
4
5
6
7
8
9
10
```

```
reverse list
```

```
10
9
8
7
6
5
4
3
2
1*/
```

Q.10 Write an algorithm that will split a linked list into two linked lists, so that successive nodes go to different lists.(the first, third, and all odd numbered nodes to the first list, and the second, fourth, and all even-numbered nodes go to the second.)  
 //Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cNode
{
    int      x;
    cNode   * next;
public:
    void     MakeNode( int );
    void     PutNext( cNode * );
    cNode   * GetNext( );
    int GetData( )
    {
        return x;
    }
    void DispNode( );
} ;
class cList
{
    cNode   * start,
            * last;
    int      TotNodes;
public:
    cList( );
    void Add( );
    void Add( int );
    void DispList( );
    void Split( cList &, cList & );
} ;
void cNode::MakeNode( int a )
{
    x      = a;
    next = '\0';
}
void cNode::PutNext( cNode * p )
{
    next = p;
}
cNode   * cNode::GetNext( )
{
    return next;
}
void cNode::DispNode( )
{
    cout << x << endl;
}
cList::cList( )
{
    start = last = '\0';
    TotNodes = 0;
}
void cList::Add( )
{
    int a;
    while( 1 )
```

```

    {
        cout << "Data      ! 0 to exit):";
        cin >> a;
        if( a == 0 )
            break;
        else
        {
            cNode * newnode = new cNode;
            newnode->MakeNode( a );
            if( !start )
                start = last = newnode;
            else
            {
                last->PutNext( newnode );
                last = newnode;
            }
        }
    }
void cList::Add( int a )
{
    cNode * newnode = new cNode;
    newnode->MakeNode( a );
    if( !start )
        start = last = newnode;
    else
    {
        last->PutNext( newnode );
        last = newnode;
    }
}
void cList::DispList( )
{
    for( cNode * ptr = start; ptr;                      //  < >
          ptr = ptr->GetNext( ) )
        ptr->DispNode( );
}
void cList::Split( cList & L1, cList & L2 )
{
    int cnt = 1;
    cNode * ptr;
    for( ptr = start; ptr;
          ptr = ptr->GetNext( ), cnt++ )
    {
        if( cnt % 2 )
            L1.Add( ptr->GetData( ) );
        else
            L2.Add( ptr->GetData( ) );
    }
}
void main( )
{
    cList L,
           L1,
           L2;
    clrscr( );
    L.Add( );
    L.DispList( );
    L.Split( L1, L2 );
    cout << "First List is : " << endl;
    L1.DispList( );
    cout << "Second List is : " << endl;
    L2.DispList( );
}

```

```
        getch( );
}

// INPUT :
Data ! 0 to exit):1
Data ! 0 to exit):2
Data ! 0 to exit):3
Data ! 0 to exit):4
Data ! 0 to exit):5
Data ! 0 to exit):6
Data ! 0 to exit):7
Data ! 0 to exit):8
Data ! 0 to exit):9
Data ! 0 to exit):10
Data ! 0 to exit):0

// OUTPUT :
1
2
3
4
5
6
7
8
9
10

First List is :
1
3
5
7
9

Second List is :
2
4
6
8
10
```

Q.11 Write a function that will concatenate two circularly linked lists, producing one circularly linked list.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cCList;
class cNode
{
    char     name[ 15 ];
    int      age;
    float    basic;
    cNode   * next;
public:
    cNode( )
    {
        strcpy( name, '\0' );
        age   = 0;
        basic = 0.0;
        next  = '\0';
    }
    cNode( char *, int, float );
    void DispNode( );
    friend      class  cCList;
}      ;
class cCList
{
    cNode   * start,
            * last;
public:
    cCList( )
    {
        start = last = '\0';
    }
    void  Add( );
    void  Add( char *, int, float );
    void  DispList( );
    cCList      Concat( cCList & );
}      ;
cNode::cNode( char * n, int  a, float  b )
{
    strcpy( name, n );
    age   = a;
    basic = b;
    next  = '\0';
}
void cNode::DispNode( )
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 15 ) << name;
    cout.setf( ios::right, ios::adjustfield );
    cout << setw( 3 ) << age << setw( 6 ) << basic << endl;
}
cCList      cCList::Concat( cCList & CL2 )
{
    cCList      tmp;
    int   flag  = 0;
    for( cNode * ptr  = last->next;                                //  < >
          !( flag && ptr == last->next );
```

```

        ptr = ptr->next, flag = 1 )
        tmp.Add( ptr->name, ptr->age, ptr->basic );
flag = 0;
for( ptr = CL2.last->next;
      !( flag && ptr == CL2.last->next );
      ptr = ptr->next, flag = 1 )
        tmp.Add( ptr->name, ptr->age, ptr->basic );
return tmp;
}
void cCList::Add( )
{
    char nm[ 15 ];
    int a;
    float b;
    while( 1 )
    {
        cout << "Name (Quit) . ";
        cin >> nm;
        if( strcmp( nm, "Quit" ) == 0 )
            break;
        else
        {
            cout << "Age      ";
            cin >> a;
            cout << "Basic    ";
            cin >> b;
            cNode * NewNode = new cNode( nm, a, b );
            if( !start )
            {
                start = last = NewNode;
                last->next = NewNode;
            }
            else
            {
                NewNode->next = start;
                last->next = NewNode;
                last = NewNode;
            }
        }
    }
}
void cCList::Add( char * nm, int a, float b )
{
    cNode * NewNode = new cNode( nm, a, b );
    if( !start )
    {
        start = last = NewNode;
        last->next = NewNode;
    }
    else
    {
        NewNode->next = start;
        last->next = NewNode;
        last = NewNode;
    }
}
void cCList::DispList( )
{
    int flag = 0;
    for( cNode * ptr = last->next; // < >
          !( flag && ptr == last->next );
          ptr = ptr->next, flag = 1 )
        ptr->DispNode( );
}

```

```

}

void main( )
{
    cCList      L,
                L1,
                L2;
    char name[ 15 ];
    clrscr( );
    L1.Add( );
    L2.Add( );
    clrscr( );
    L1.DispList( );
    cout << endl;
    L2.DispList( );
    L = L1.Concat( L2 );
    cout << endl;
    L.DispList( );
    getch( );
}

// INPUT :
Name (Quit) . ENTER
Age   21
Basic 34
Name (Quit) . DATA
Age   76
Basic 65
Name (Quit) . SYSTEM
Age   34
Basic 89
Name (Quit) . Quit
Name (Quit) . xyz
Age   23
Basic 45
Name (Quit) . abc
Age   56
Basic 78
Name (Quit) . Quit

// OUTPUT :
FIRST LIST

ENTER        21      34
DATA          76      65
SYSTEM        34      89

SECOND LIST

xyz          23      45
abc          56      78

AFTER CONCATENATION OF THESE TWO LISTS

ENTER        21      34
DATA          76      65
SYSTEM        34      89
xyz          23      45
abc          56      78

```

Q.12 Write a function that will split a circularly linked list, into two circularly linked lists.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cCList;
class cNode
{
    char     name[ 15 ];
    int      age;
    float    basic;
    cNode   * next;
public:
    cNode( )
    {
        strcpy( name, '\0' );
        age   = 0;
        basic = 0.0;
        next  = '\0';
    }
    cNode( char *, int, float );
    void DispNode();
    friend     class  cCList;
} ;
class cCList
{
    cNode   * start,
            * last;
public:
    cCList( )
    {
        start = last = '\0';
    }
    void Add();
    void Add( char *, int, float );
    void DispList();
    void Split( cCList &, cCList & );
} ;
cNode::cNode( char * n, int  a, float  b )
{
    strcpy( name, n );
    age   = a;
    basic = b;
    next  = '\0';
}
void cNode::DispNode()
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 15 ) << name;
    cout.setf( ios::right, ios::adjustfield );
    cout << setw( 3 ) << age << setw( 6 ) << basic << endl;
}
void cCList::Split( cCList & CL1, cCList & CL2 )
{
    int cnt  = 1;
    int flag  = 0;
    for( cNode * ptr  = last->next;                                //  < >
          !( flag && ptr == last->next );
          ptr = ptr->next, flag = 1, cnt  )

```

```

    {
        if( cnt % 2 )
            CL1.Add( ptr->name, ptr->age, ptr->basic );
        else
            CL2.Add( ptr->name, ptr->age, ptr->basic );
    }
}

void cCList::Add( )
{
    char nm[ 15 ];
    int a;
    float b;
    while( 1 )
    {
        cout << "Name      (Quit)      .      ";
        cin >> nm;
        if( strcmp( nm, "Quit" ) == 0 )
            break;
        else
        {
            cout << "Age      ";
            cin >> a;
            cout << "Basic      ";
            cin >> b;
            cNode * NewNode = new cNode( nm, a, b );
            if( !start )
            {
                start = last = NewNode;
                last->next = NewNode;
            }
            else
            {
                NewNode->next = start;
                last->next = NewNode;
                last = NewNode;
            }
        }
    }
}

void cCList::Add( char * nm, int a, float b )
{
    cNode * NewNode = new cNode( nm, a, b );
    if( !start )
    {
        start = last = NewNode;
        last->next = NewNode;
    }
    else
    {
        NewNode->next = start;
        last->next = NewNode;
        last = NewNode;
    }
}

void cCList::DispList( )
{
    int flag = 0;
    for( cNode * ptr = last->next; // < >
         !( flag && ptr == last->next );
         ptr = ptr->next, flag = 1 )
        ptr->DispNode( );
}

```

```

void main( )
{
    cCList      L,
                L1,
                L2;
    char   name[ 15 ];
    clrscr( );
    L.Add( );
    clrscr( );
    L.DispList( );
    cout << endl;
    L.Split( L1, L2 );
    L1.DispList( );
    cout << endl;
    L2.DispList( );
    getch( );
}

// INPUT :
Name  (Quit)      .          GHJ
Age   4
Basic 5
Name  (Quit)      .          ABC
Age   3
Basic 4
Name  (Quit)      .          XYZ
Age   3
Basic 6
Name  (Quit)      .          SYSTEM
Age   5
Basic 4
Name  (Quit)      .          ENTER
Age   5
Basic 8
Name  (Quit)      .          Quit

// OUTPUT :
GHJ           4          5
ABC           3          4
XYZ           3          6
SYSTEM        5          4
ENTER         5          8

GHJ           4          5
ABC           3          4
XYZ           3          6
SYSTEM        5          4
ENTER         5          8

```

Q.13 Write a function for manipulating a doubly linked list as follows:

- e. add a node after p.
- f. Add a node before p.
- g. Delete node p.
- h. traverse the list.

```
// Example Code :  
#include    <iostream.h>  
#include    <iomanip.h>  
#include    <conio.h>  
#include    <stdio.h>  
#include    <string.h>  
class cDList;  
class cNode  
{  
    char    name[ 15 ];  
    int     age;  
    float   basic;  
    cNode  * next,  
           * prev;  
public:  
    cNode( )  
    {  
        strcpy( name, '\0' );  
        age = 0;  
        basic = 0.0;  
        next = prev = '\0';  
    }  
    cNode( char *, int, float );  
    void DispNode();  
    friend    class  cDList;  
};  
class cDList  
{  
    cNode  * start,  
           * last;  
public:  
    cDList( )  
    {  
        start = last = '\0';  
    }  
    void AddAfter( char *, int, float, char * );  
    void AddBefore( char *, int, float, char * );  
    void DispList();  
    void Delete( char * );  
    void Search( char * );  
    void Add();  
};  
cNode::cNode( char * n, int  a, float  b )  
{  
    strcpy( name, n );  
    age = a;  
    basic = b;  
    next = prev = '\0';  
}  
void cNode::DispNode()  
{  
    cout.setf( ios::left, ios::adjustfield );  
    cout << setw( 15 ) << name;  
    cout.setf( ios::right, ios::adjustfield );  
    cout << setw( 3 ) << age << setw( 6 ) << setw( 7 ) << basic << endl;  
}  
void cDList::AddAfter( char * n, int  a, float  b,  
                      char * s )
```

```

{
    cNode * NewNode = new cNode( n, a, b );
    for( cNode * ptr = start;                                // < >
          ptr && strcmp( ptr->name, s );
          ptr = ptr->next )
    ;
    if( ptr )
    {
        if( strcmp( ptr->name, s ) == 0 )
        {
            ptr->next->prev = NewNode;
            NewNode->next     = ptr->next;
            ptr->next         = NewNode;
            NewNode->prev     = ptr;
        }
        else
            cout << "Name not Found!";
    }
    else
    {
        cout << "List not Existing .....Append Node ? (y/n) :";
        char ch = getchar();
        fflush( stdin );
        if( ch == 'Y' || ch == 'y' )
            start = last = NewNode;
    }
}
void cDList::AddBefore( char * n, int a, float b,
                      char * s )
{
    cNode * NewNode = new cNode( n, a, b );
    for( cNode * ptr = start;                                // < >
          ptr && strcmp( ptr->name, s );
          ptr = ptr->next )
    ;
    if( ptr )
    {
        if( strcmp( ptr->name, s ) == 0 )
        {
            if( ptr == start )
            {
                NewNode->next = start;
                start->prev   = NewNode->next;
                start         = NewNode;
            }
            else
            {
                NewNode->next     = ptr;
                NewNode->prev     = ptr->prev;
                ptr->prev->next = NewNode;
                ptr->prev         = NewNode;
            }
        }
        else
            cout << "Name not Found !";
    }
    else
    {
        cout << "List not Existing .....Append Node ? (y/n) :";
        char ch = getchar();
        fflush( stdin );
        if( ch == 'Y' || ch == 'y' )

```

```

                start = last = NewNode;
            }
        }
void cDList::Delete( char * n )
{
    for( cNode * ptr = start;                                // < >

          ptr && strcmp( ptr->name, n );
          ptr = ptr->next )
    ;
    if( ptr )
    {
        if( strcmp( ptr->name, n ) == 0 )
        {
            if( ptr == start )
            {
                start      = start->next;
                start->prev = '\0';
            }
            else
            {
                ptr->prev->next = ptr->next;
                ptr->next->prev = ptr->prev;
            }
            delete ptr;
        }
        else
            cout << "Node Not Found... ! " << endl;
    }
    else
        cout << "List does not Exists ...!" << endl;
}
void cDList::Search( char * n )
{
    int count = 1,
        yes = 0;
    if( start )
    {
        for( cNode * ptr = start; ptr;                                // < >

              ptr = ptr->next, count++ )
        {
            if( strcmp( ptr->name, n ) == 0 )
            {
                cout << "Found at Rec # " << count << endl;
                ptr->DispNode( );
                yes++;
            }
        }
        if( !yes )
            cout << "Name not Found ..... !" << endl;
    }
    else
        cout << "List does not exists ...!" << endl;
}
void cDList::Add( )
{
    char name[ 15 ];
    int age;
    float basic;
    while( 1 )
    {
        cout << "Name      (Quit)      :" ;
        cin >> name;

```

```

        if( strcmp( name, "Quit" ) == 0 )
            break;
        else
        {
            cout << "Age      ";
            cin >> age;
            cout << "Basic    ";
            cin >> basic;
            cNode * NewNode = new cNode
                ( name, age,
                  basic );
            if( !start )
                start = last = NewNode;
            else
            {
                last->next = NewNode;
                NewNode->prev = last;
                last = NewNode;
            }
        }
    }
void cDList::DispList( )
{
    for( cNode * ptr = start; ptr; ptr = ptr->next ) // < >
        ptr->DispNode( );
}
void main( )
{
    cDList D;
    clrscr( );
    D.Add( );
    clrscr( );
    D.DispList( );
// D.AddAfter("Ramesh",23,5000,"Mickey");
D.AddBefore("Diwakar",25,9000,"Ramesh");
/*
char n[15];
cout<<"Name to be Searched:" ;
cin>>n;
// D.Search(n);
D.Delete (n);
cout<<endl<<endl; */
    D.DispList( );
    getch( );
}
//INPUT :
Name (Quit) :GHJ
Age 5
Basic 6
Name (Quit) :YOO
Age 6
Basic 7
Name (Quit) :GHGJ
Age 89
Basic 6
Name (Quit) :TJ
Age 8
Basic 9
Name (Quit) :SF
Age 7
Basic 5
Name (Quit) :Quit

```

/ /OUTPUT :

GHJ	5	6
YOO	6	7
GHGJ	89	6
TJ	8	9
SF	7	5
GHJ	5	6
YOO	6	7
GHGJ	89	6
TJ	8	9
SF	7	5

Q.14 Write and test a recursive function that return the maximum among the first n elements of an array.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
void main( )
{
    int IsMax( int, int, int * );
    int n,
        * arr;
    clrscr( );
    cout << "Enter Number of Elements:" ;
    cin >> n;
    arr = new int [n];
    cout << "Enter Elements : " << endl;
    for( int i = 0; i < n; i++ )                                //  < >

        cin >> arr[ i ];
    cout << IsMax( arr[ n - 1 ], n - 1, arr );
    getch( );
}
int IsMax( int      Max, int   n, int * arr )
{
    if( n < 0 )
        return Max;
    else
    {
        int tmp = Max;
        if( tmp < arr[ n ] )
            Max = arr[ n ];
        n--;
        tmp = IsMax( Max, n, arr );
    }
}

// INPUT :
Enter Number of Elements:5
Enter Elements      :
1
2
3
4
5
// OUTPUT :

MAXIMUM NO. IN THE LIST
5
```

Q.15 Write the following function template:

```

Template <class T> void reverse(stack<T>&);
// reverse the contents of the given stack;

// Example Code :

#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define        MAX    10
template<class xType>class cStack
{
    xType  data[ MAX ];
    int    top;
    int    Empty( );
    int Full( );
public:
    cStack( )
    {
        top = 0;
    }
    void Push( xType & );
    xType & Pop( );
    void Disp( );
    void Rev( cStack< xType > & );
}
template<class xType>  void cStack< xType >::Rev(
                                         cStack< xType > & S1 )
{
    while( !Empty( ) )
        S1.Push( Pop( ) );
}
template<class xType>  int cStack< xType >::Empty( )
{
    if( top < 0 )
        return 1;
    return 0;
}
template<class xType>  int cStack< xType >::Full( )
{
    if( top > MAX )
        return 1;
    return 0;
}
template<class xType>  void cStack< xType >::Disp( )
{
    for( int i = top; i > 0; i-- ) //  < >
        cout << data[ i ] << "\t";
}
template<class xType>  void cStack< xType >::Push(
                                         xType & x )
{
    if( !Full( ) )
        data[ ++top ] = x;
    else
        cout << "Stack Overflow";
}
template<class xType>  xType& cStack< xType >::Pop( )
{
    if( !Empty( ) )
        return ( data[ top-- ] );
}

```

```

        else
            cout << "Stack UnderFlow";
    }
void main( )
{
    cStack< char >      S,
                        s;
    clrscr( );
    S.Push( 'a' );
    S.Push( 'n' );
    S.Push( 'a' );
    S.Push( 'n' );
    S.Push( 'g' );
    S.Disp( );
    cout << endl;
    S.Rev( s );
    s.Disp( );
    getch( );
}

```

// INPUT :  
GIVEN STACK

g n a n a

// OUTPUT :  
REVERSE THE CONTENTS OF THE STACK

a n a n g

Q.16 Write the following function template:

```

T& bottom();           // returns the bottom element

```

// example Code :

```

#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define       INT_MAX     999
#define       MAX        10
class cStack
{
    int data[ MAX ];
    int top;
    int Empty();
    int Full();
public:
    cStack()
    {
        top = 0;
    }
    void Push();
    void Push( int );
    void DispStack();
    int Bottom();
}
int cStack::Empty()
{
    if( top < 0 )
        return 1;
    return 0;
}
int cStack::Full()
{
    if( top > MAX )
        return 1;
    return 0;
}
int cStack::Bottom()
{
    if( !Empty() )
        return data[ 1 ];
    return INT_MAX;
}
void cStack::Push()
{
    int x;
    if( !Full() )
    {
        while( 1 )
        {
            cout << "Data : ";
            cin >> x;
            if( x == 0 )
                break;
            else
                data[ ++top ] = x;
        }
    }
    else
        cout << "stack overflow";
}

```

```

void cStack::Push( int x )
{
    if( !Full( ) )
        data[ ++top ] = x;
    else
        cout << "Stack Overflow";
}
void cStack::DispStack( )
{
    for( int i = top; i > 0; i-- )           // < >
        cout << data[ i ] << endl;
}
void main( )
{
    cStack      S1;
    clrscr( );
    S1.Push( );
    clrscr( );
    cout << "stack is" << endl;
    S1.DispStack( );
/* int x=S1.Third();
cout<<"III element = "<<x<<endl;
x=S1.Peek();
cout<<"I elemm = "<<x<<endl; */
    int x = S1.Bottom( );
    cout << "Last ele = " << x << endl;
    getch( );
}

// INPUT :

Data:
1
Data:
2
Data:
3
Data:
4
Data:
5
Data:
0

// OUTPUT :

stack is
5
4
3
2
1
Last ele = 1

```

Q.17 Use the stack to implement the following function:  
Bool is palindrome (string s);

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define        ERR
#define        MAX    10
class cStack
{
    char data[ MAX ];
    int top;
    int Empty( );
    int Full( );
public:
    cStack( )
    {
        top = 0;
    }
    void Push( char );
    int Pop( );
    void DispStack( );
    int IsPal( char * );
}
int cStack::IsPal( char * S )
{
    int Flag = 1;
    for( int i = 0; i < ( strlen( S ) ) / 2; i++ ) // < >
        Push( S[ i ] );
    for( ++i; S[ i ]; i++ )
    {
        if( S[ i ] != Pop( ) )
        {
            Flag = 0;
            break;
        }
    }
    return Flag;
}
int cStack::Empty( )
{
    if( top < 0 )
        return 1;
    return 0;
}
int cStack::Full( )
{
    if( top > MAX )
        return 1;
    return 0;
}
void cStack::Push( char x )
{
    if( !Full( ) )
        data[ ++top ] = x;
    else
        cout << "Stack Overflow";
}
int cStack::Pop( )
```

```

{
    if( !Empty( ) )
        return ( data[ top-- ] );
    else
        cout << "Stack UnderFlow";
}
void cStack::DispStack( )
{
    for( int i = top; i > 0; i-- )           // < >
        cout << data[ i ] << endl;
}
void main( )
{
    cStack      S1;
    clrscr( );
    char str[ 10 ];
    cout << "String :";
    cin >> str;
    if( S1.IsPal( str ) )
        cout << "Pallindrome";
    else
        cout << "Not Pallindrome";
    getch( );
}

// INPUT & OUTPUT :

String:
POP
Pallindrome

String:
PUSH
Not
Pallindrome

```

Q.18 Write the following overloaded equality operator for queues:  
Bool queue::operator +=(const queue&)

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#define        MAX    10
class cQueue
{
    int items[ MAX ];
    int front,
        rear,
        count;
    int Full( );
    int Empty( );
public:
    cQueue( )
    {
        rear = front = count = 0;
    }
    void Add( );
    void Add( int );
    void Disp( );
    int Del( );
    int Peek( );
    int operator ==( const cQueue & );
}
int cQueue::operator ==( const cQueue & Q )
{
    int flag = 1;
    if( count != Q.count )
        flag = 0;
    else
    {
        while( count > 0 )
        {
            if( Del( ) != Q.Del( ) )
            {
                flag = 0;
                break;
            }
        }
    }
    return flag;
}
int cQueue::Full( )
{
    if( count >= MAX )
        return 1;
    return 0;
}
int cQueue::Empty( )
{
    if( count == 0 )
        return 1;
    return 0;
}
void cQueue::Add( )
{
    int x;
    if( !Full( ) )
```

```

{
    while( 1 )
    {
        cout << "Data (0-Exit) :";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
            count++;
            items[ rear++ ] = x;
        }
    }
}
else
    cout << "Queue Full..... Add Aborted! " << endl;
}
void cQueue::Add( int x )
{
    if( !Full( ) )
    {
        count++;
        items[ rear++ ] = x;
    }
    else
        cout << "Queue Full..... Add Aborted!" << endl;
}
int cQueue::Del( )
{
    int x = -999;
    if( !Empty( ) )
    {
        x = items[ front ];
        for( int j = front; j < rear; j++ )           // < >

            items[ j ] = items[ j + 1 ];
        count--;
    }
    else
        cout << "Queue is Empty ..... Delete Aborted!" << endl;
    return x;
}
int cQueue::Peek( )
{
    if( !Empty( ) )
        return items[ front ];
    else
        cout << "Queue is Empty ..... Delete Aborted!" << endl;
    return -999;
}
void cQueue::Disp( )
{
    for( int i = front; i < count; i++ )           // < >

        cout << items[ i ] << endl;
}
void main( )
{
    cQueue      Q1,Q2;
    clrscr( );
    cout << "Enter elements of I Queue" << endl;
    Q1.Add( );
    cout << "Enter elements of II Queue" << endl;
    Q2.Add( );
}

```

```

        clrscr( );
        Q1.Disp( );
        cout << endl << endl;
        Q2.Disp( );
        if( Q1 == Q2 )
            cout << "Equal" << endl;
        else
            cout << "Different" << endl;
        getch( );
    }

// INPUT :
Enter elements of I Queue
Data (0-Exit) :1
Data (0-Exit) :2
Data (0-Exit) :3
Data (0-Exit) :4
Data (0-Exit) :5
Data (0-Exit) :0
Enter elements of II Queue
Data (0-Exit) :1
Data (0-Exit) :2
Data (0-Exit) :3
Data (0-Exit) :4
Data (0-Exit) :5
Data (0-Exit) :0

// OUTPUT :
1
2
3
4
5

1
2
3
4
5
Equal
// INPUT :
Enter
elements of I Queue
Data (0-Exit) :1
Data (0-Exit) :3
Data (0-Exit) :4
Data (0-Exit) :3
Data (0-Exit) :0
Enter elements of II Queue
Data (0-Exit) :9
Data (0-Exit) :5
Data (0-Exit) :4
Data (0-Exit) :4
Data (0-Exit) :0
// OUTPUT :
1
3
4
3
9
5
4
4
Different

```

Q.19 Implement the following friend function for the Integer class:

```

friend istream& operator>> (istream& istr, Integer& x);
// Input x from the istr input stream
```

// Example Code :

```

#include      <iostream.h>
#include      <conio.h>
#include      <iomanip.h>
class cInteger
{
    int num;
public:
    cInteger( )
    {
        num = 0;
    }
    cInteger( int a )
    {
        num = a;
    }
    friend ostream & operator <<( ostream &, cInteger & );
    friend      istream & operator >>( istream &, cInteger & );
    friend      cInteger operator *( const cInteger &, int );
}
ostream & operator <<( ostream & out, cInteger & x )
{
    out << setw( 5 ) << x.num << endl;
    return out;
}
istream & operator >>( istream & in, cInteger & x )
{
    in >> x.num;
    return in;
}
cInteger operator *( const cInteger & I, int x )
{
    cInteger tmp;
    tmp.num = I.num * x;
    return tmp;
}
void main( )
{
    cInteger I1,I2,I3,I4;
    clrscr( );
    cin >> I1 >> I2 >> I3;
    cout << I1 << I2 << I3;
    I4 = I1 * 3;
    cout << I4;
    getch( );
}

// INPUT :
2
3
4

// OUTPUT :
2
3
4
I1*3=
6
```

Q.20 Write the following function that uses a binary search tree to sort an array a of n elements :

```

void sort (type* a, int n);

// Example Code :

#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
enum
{
    LEFT,
    RIGHT
} ;
class cTree;
class cNode
{
    int      data;
    cNode   * lptr,
            * rptr;
public:
    cNode( )
    {
        data = 0;
        lptr = rptr = '\0';
    }
    cNode( int  x )
    {
        data = x;
        lptr = rptr = '\0';
    }
    friend      class  cTree;
} ;
class cTree
{
    cNode   * Root;
public:
    cTree( )
    {
        Root = '\0';
    }
    void sortArray( int *, int );
    void Insert( int );
    int  inOrder( cNode *, int *, int );
    void Insert( );
    void InOrder( cNode * );
    void PostOrder( cNode * );
    void PreOrder( cNode * );
    int  IsEmpty( )const;
    cNode   * GetRoot( )
    {
        return Root;
    }
} ;
void cTree::Insert( int  x )
{
    int  dir;
    cNode   * ptr,
            * prev;
    cNode   * NewNode  = new cNode( x );
    if( !Root )

```

```

        Root = NewNode;
    else
    {
        for( ptr = Root; ptr; )
        {
            prev = ptr;
            if( ( ptr->data ) < ( NewNode->data ) )
            {
                dir = RIGHT;
                ptr = ptr->rptr;
            }
            else
            {
                dir = LEFT;
                ptr = ptr->lptr;
            }
        }
        if( dir == LEFT )
            prev->lptr = NewNode;
        else
            prev->rptr = NewNode;
    }
}
int cTree::inOrder( cNode * xRoot, int * arr, int i )
{
    if( !xRoot )
        return i;
    i           = inOrder( xRoot->lptr, arr, i );
    arr[ i++ ] = xRoot->data;
    i           = inOrder( xRoot->rptr, arr, i );
    return i;
}
void cTree::sortArray( int * arr, int n )
{
    for( int i = 0; i < n; i++ )                                // < >

        Insert( arr[ i ] );
    inOrder( Root, arr, 0 );
}
int cTree::IsEmpty( )const
{
    if( !Root )
        return 1;
    return 0;
}
void cTree::Insert( )
{
    int      x,
            dir;
    cNode  *  ptr,
            *  prev;
    while( 1 )
    {
        cout << "Data : ";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
            cNode  *  NewNode = new cNode( x );
            if( !Root )
                Root = NewNode;
            else
            {

```

```

        for( ptr = Root; ptr; )
        {
            prev = ptr;
            if( ( ptr->data )
                < ( NewNode->data ) )
            {
                dir = RIGHT;
                ptr = ptr->rptr;
            }
            else
            {
                dir = LEFT;
                ptr = ptr->lptr;
            }
        }
        if( dir == LEFT )
            prev->lptr = NewNode;
        else
            prev->rptr = NewNode;
    }
}
void cTree::InOrder( cNode * Curr )
{
    if( Curr )
    {
        InOrder( Curr->lptr );
        cout << Curr->data << endl;
        InOrder( Curr->rptr );
    }
}
void cTree::PostOrder( cNode * Curr )
{
    if( Curr )
    {
        PostOrder( Curr->lptr );
        PostOrder( Curr->rptr );
        cout << Curr->data << endl;
    }
}
void cTree::PreOrder( cNode * Curr )
{
    if( Curr )
    {
        cout << Curr->data << endl;
        PreOrder( Curr->lptr );
        PreOrder( Curr->rptr );
    }
}
void main( )
{
    cTree T;
    int arr[ 10 ];
    clrscr( );
    for( int i = 0; i < 10; i++ ) // < >
    {
        cout << "Enter Value for " << setw( 2 ) << i + 1 << ":" ;
        cin >> arr[ i ];
    }
    T.sortArray( arr, 10 );
    for( i = 0; i < 10; i++ )
        cout << endl << arr[ i ];
}

```

```
    getch( );
}
```

```
// INPUT :
```

```
Enter Value
for 1:1
Enter Value
for 2:2
Enter Value
for 3: 5
Enter Value
for 4:3
Enter Value
for 5:6
Enter Value
for 6:4
Enter Value
for 7:8
Enter Value
for 8:6
Enter Value
for 9:54
Enter Value
for 10:9
```

```
// OUTPUT :
```

```
1
2
3
4
5
6
6
8
9
54
```

Q.21 Implement the following Tree class member function:

```

Bool empty() const;
// returns true iff this tree has no elements

// Example Code :

#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
enum
{
    LEFT,
    RIGHT
} ;
class cTree;
class cNode
{
    int      data;
    cNode   * lptr,
            * rptr;
public:
    cNode( )
    {
        data = 0;
        lptr = rptr = '\0';
    }
    cNode( int  x )
    {
        data = x;
        lptr = rptr = '\0';
    }
    friend      class  cTree;
} ;
class cTree
{
    cNode   * Root;
public:
    cTree( )
    {
        Root = '\0';
    }
    void Insert( );
    void InOrder( cNode * );
    void PostOrder( cNode * );
    void PreOrder( cNode * );
    int  IsEmpty( )const;
    cNode   * GetRoot( )
    {
        return Root;
    }
} ;
int cTree::IsEmpty( )const
{
    if( !Root )
        return 1;
    return 0;
}
void cTree::Insert( )
{
    int      x,
            dir;
    cNode   * ptr,

```

```

        * prev;
while( 1 )
{
    cout << "Data : ";
    cin >> x;
    if( x == 0 )
        break;
    else
    {
        cNode * NewNode = new cNode( x );
        if( !Root )
            Root = NewNode;
        else
        {
            for( ptr = Root; ptr; )
            {
                prev = ptr;
                if( ( ptr->data )
                    < ( NewNode->data ) )
                {
                    dir = RIGHT;
                    ptr = ptr->rptr;
                }
                else
                {
                    dir = LEFT;
                    ptr = ptr->lptr;
                }
                if( dir == LEFT )
                    prev->lptr = NewNode;
                else
                    prev->rptr = NewNode;
            }
        }
    }
}
void cTree::InOrder( cNode * Curr )
{
    if( Curr )
    {
        InOrder( Curr->lptr );

        cout << Curr->data << endl;
        InOrder( Curr->rptr );
    }
}
void cTree::PostOrder( cNode * Curr )
{
    if( Curr )
    {
        PostOrder( Curr->lptr );
        PostOrder( Curr->rptr );
        cout << Curr->data << endl;
    }
}
void cTree::PreOrder( cNode * Curr )
{
    if( Curr )
    {
        cout << Curr->data << endl;
        PreOrder( Curr->lptr );
        PreOrder( Curr->rptr );
    }
}

```

```
        }
    }
void main( )
{
    cTree  T;
    clrscr( );
    T.Insert( );
    T.PostOrder( T.GetRoot( ) );
    getch( );
}
```

// INPUT :

Data :  
1  
Data:

2  
Data:  
3  
Data:  
4  
Data:  
5  
Data:  
0

// OUTPUT :

5  
4  
3  
2  
1

Q.22 Implement the following Tree class member function:

```

int size() const;
// returns the number of elements in this tree

// Example Code :

#include    <iostream.h>
#include    <iomanip.h>
#include    <conio.h>
enum
{
    LEFT,
    RIGHT
} ;
class cTree;
class cNode
{
    int      data;
    cNode   * lptr,
            * rptr;
public:
    cNode( )
    {
        data = 0;
        lptr = rptr = '\0';
    }
    cNode( int  x )
    {
        data = x;
        lptr = rptr = '\0';
    }
    friend     class  cTree;
} ;
class cTree
{
    cNode   * Root;
    int getSize( cNode *, int );
public:
    cTree( )
    {
        Root = '\0';
    }
    void Insert( );
    void InOrder( cNode * );
    void PostOrder( cNode * );
    void PreOrder( cNode * );
    int IsEmpty( )const;
    int Min( )const;
    int Max( )const;
    int size( )const;
    cNode   * GetRoot( )
    {
        return Root;
    }
} ;
int cTree::Min( )const
{
    cNode   * ptr,
            * prev;
    for( ptr = Root; ptr;
         prev = ptr, ptr = ptr->lptr )
    ;
    return prev->data;
}

```

```

int cTree::Max( )const
{
    cNode  *  ptr,
            *  prev;
    for(  ptr = Root;  ptr;
          prev = ptr,  ptr = ptr->rptr )
    ;
    return prev->data;
}
int cTree::getSize( cNode * R, int  ctr )
{
    if( !R )
        return ctr;
    ctr = getSize( R->lptr, ctr );
    ctr++;
    getSize( R->rptr, ctr );
    return ctr;
}
int cTree::size( )const
{
    int cnt  = 0;
    cnt = getSize( Root, 0 );
    return cnt;
}
int cTree::IsEmpty( )const
{
    if( !Root )
        return 1;
    return 0;
}
void cTree::Insert( )
{
    int      x,
            dir;
    cNode  *  ptr,
            *  prev;
    while( 1 )
    {
        cout << "Data : ";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
            cNode  *  NewNode  = new cNode( x );
            if( !Root )
                Root = NewNode;
            else
            {
                for(  ptr = Root;  ptr;  )
                {
                    prev = ptr;
                    if( ( ptr->data )
                        < ( NewNode->data )  )
                    {
                        dir = RIGHT;
                        ptr = ptr->rptr;
                    }
                    else
                    {
                        dir = LEFT;
                        ptr = ptr->lptr;
                    }
                }
            }
        }
    }
}

```

```

        }
        if( dir == LEFT )
            prev->lptr = NewNode;
        else
            prev->rptr = NewNode;
    }
}
void cTree::InOrder( cNode * Curr )
{
    if( Curr )
    {
        InOrder( Curr->lptr );
        cout << Curr->data << endl;
        InOrder( Curr->rptr );
    }
}
void cTree::PostOrder( cNode * Curr )
{
    if( Curr )
    {
        PostOrder( Curr->lptr );
        PostOrder( Curr->rptr );
        cout << Curr->data << endl;
    }
}
void cTree::PreOrder( cNode * Curr )
{
    if( Curr )
    {
        cout << Curr->data << endl;
        PreOrder( Curr->lptr );
        PreOrder( Curr->rptr );
    }
}
void main( )
{
    cTree T;
    clrscr( );
    T.Insert( );
    cout << "Max =" << T.size( );
    getch( );
}

// INPUT :
Data :
1
Data:
2
Data:
3
Data:
4
Data:
5
Data:
6
Data:
0

// OUTPUT :
Max =
1

```

Q.23 Implement the following member functions for the Tree class :

```

int width (int n);
// returns the total number of nodes at level n in this tree

// Example Code :

#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
enum
{
    LEFT,RIGHT
} ;
class cTree;
class cNode
{
    int      data;
    cNode   * lptr,
            * rptr;
public:
    cNode( )
    {
        data = 0;
        lptr = rptr = '\0';
    }
    cNode( int  x )
    {
        data = x;
        lptr = rptr = '\0';
    }
    friend      class  cTree;
} ;
class cTree
{
    cNode   * Root;
public:
    cTree( )
    {
        Root = '\0';
    }
    int  Expand( int );
    int  getNodesAtLeveln( cNode * R, int  cnt,
                           int      curLevei, int  level );
    void Insert( );
    void PreOrder( cNode * );
    int  IsEmpty( )const;
    cNode   * GetRoot( )
    {
        return Root;
    }
    int Level( int );
} ;
int cTree::Expand( int  level )
{
    int n  = getNodesAtLeveln( Root, 0, 1, level );
    return n;
}
int cTree::getNodesAtLeveln( cNode * R, int  cnt,
                           int      curLevel, int  level )
{
    if( !R )
        return cnt;
    if( curLevel == level )
        cnt++;
}

```

```

        cnt = getNodesAtLevel( R->lptr, cnt, curLevel, level );
        cnt = getNodesAtLevel( R->rptr, cnt, curLevel, level );
        return cnt;
    }
int cTree::IsEmpty( )const
{
    if( !Root )
        return 1;
    return 0;
}
void cTree::Insert( )
{
    int      x,
            dir;
    cNode   *  ptr,
            *  prev;
    while( 1 )
    {
        cout << "Data : ";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
            cNode   *  NewNode  = new cNode( x );
            if( !Root )
                Root = NewNode;
            else
            {
                for( ptr = Root; ptr; )
                {
                    prev = ptr;
                    if( ( ptr->data )
                        < ( NewNode->data ) )
                    {
                        dir = RIGHT;
                        ptr = ptr->rptr;
                    }
                    else
                    {
                        dir = LEFT;
                        ptr = ptr->lptr;
                    }
                }
                if( dir == LEFT )
                    prev->lptr = NewNode;
                else
                    prev->rptr = NewNode;
            }
        }
    }
}
void cTree::PreOrder( cNode * Curr )
{
    if( Curr )
    {
        PreOrder( Curr->lptr );
        PreOrder( Curr->rptr );
    }
}
int cTree::Level( int  x )
{
    int      cnt  = 0;
    cNode   *  ptr  = Root;

```

```

        while( ptr && ptr->data != x )
    {
        if( ptr->data > x )
            ptr = ptr->lptr;
        else
            ptr = ptr->rptr;
        cnt++;
    }
    return cnt;
}
void main( )
{
    cTree T;
    int n;
    clrscr( );
    T.Insert( );
    cout << "Enter the Level to Expand :";
    cin >> n;
    int cnt = T.Expand( n );
    cout << "No . of Nodes at Level " << n << " is " << cnt;
    getch( );
}

// INPUT :

Data :
1
Data:
2
Data:
3
Data:
4
Data:
5
Data:

6
Data:
0
Enter the Level to Expand :1

// OUTPUT :

No of Nodes at Level 1 is 6

```

Q.24 Implement the following member function for the Tree class:

```
void defoliate();
// removes all the leaves from this tree.
```

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
enum
{
    LEFT,
    RIGHT
} ;
class cTree;
class cNode
{
    int      data;
    cNode   * lptr,
            * rptr;
public:
    cNode( )
    {
        data = 0;
        lptr = rptr = '\0';
    }
    cNode( int  x )
    {
        data = x;
        lptr = rptr = '\0';
    }
    friend      class  cTree;
} ;
class cTree
{
    cNode   * Root;
    cNode   * Defoliate( cNode * );
public:
    cTree( )
    {
        Root = '\0';
    }
    void Insert( );
    void Defoliate( )const;
    void InOrder( cNode * );
    cNode   * GetRoot( )
    {
        return Root;
    }
} ;
void cTree::Insert( )
{
    int      x,
            dir;
    cNode   * ptr,
            * prev;
    while( 1 )
    {
        cout << "Data : ";
        cin >> x;
        if( x == 0 )
            break;
        else
        {
```

```

cNode  * NewNode  = new cNode( x );
if( !Root )
    Root = NewNode;
else
{
    for( ptr = Root; ptr; )
    {
        prev = ptr;
        if( ( ptr->data )
            < ( NewNode->data ) )
        {
            dir = RIGHT;
            ptr = ptr->rptr;
        }
        else
        {
            dir = LEFT;
            ptr = ptr->lptr;
        }
    }
    if( dir == LEFT )
        prev->lptr = NewNode;
    else
        prev->rptr = NewNode;
}
}
}

void cTree::InOrder( cNode * Curr )
{
    if( Curr )
    {
        InOrder( Curr->lptr );
        cout << Curr->data << endl;
        InOrder( Curr->rptr );
    }
}
cNode  * cTree::Defoliate( cNode * R )
{
    if( !R )
        return R;
    if( !( R->lptr || R->rptr ) )
    {
        delete R;
        return NULL;
    }
    R->lptr = Defoliate( R->lptr );
    R->rptr = Defoliate( R->rptr );
    return R;
}
void cTree::Defoliate( )const
{
    Defoliate( Root );
}
void main( )
{
    cTree  T;
    clrscr( );
    T.Insert( );
    T.InOrder( T.GetRoot( ) );
    T.Defoliate( );
    cout << endl << endl;
    T.InOrder( T.GetRoot( ) );
    getch( );
}

```

```
}
```

```
// INPUT :
```

```
Data :  
1  
Data:  
3  
Data:  
5  
Data:  
7  
Data:  
9  
Data:  
0
```

```
// OUTPUT :
```

```
1  
3  
5  
7  
9
```

```
1  
3  
5  
7
```

Q.25 There are 10 records present in a file with the following structures

```

struct
{
char itemcode[6] ;
char itemname[20];
int qty;
};

Write a program to read these records and arrange them in ascending order and
write them in a target file.
```

// Example Code :

```

#include      <iomanip.h>
#include      <iostream.h>
#include      <fstream.h>
#include      <string.h>
#include      <conio.h>
#include      <stdlib.h>
#define      MAX      3
class cItem
{
    char itcode[ 6 ];
    char itname[ 20 ];
    int qty;
public:
    void ReadData( );
    void WriteData( );
    friend void Sort( cItem *, int );
} ;
void cItem::ReadData( )
{
    cout << "ITEM CODE ";
    cin >> itcode;
    cout << "ITEM NAME ";
    cin >> itname;
    cout << "QUANTITY ";
    cin >> qty;
}
void cItem::WriteData( )
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 8 ) << itcode << setw( 20 ) << itname;
    cout.setf( ios::right, ios::adjustfield );
    cout << setw( 5 ) << qty << endl;
}
void Sort( cItem * T, int n )
{
    for( int i = 0; i < n - 1; i++ ) // < >
    {
        for( int j = 0; j < n - 1 - i; j++ ) // < >
        {
            if( strcmp( T[ j ].itcode,
                        T[ j + 1 ].itcode ) > 0 )
            {
                cItem tmp = T[ j ];
                T[ j ] = T[ j + 1 ];
                T[ j + 1 ] = tmp;
            }
        }
    }
}
```

```

}

void main( )
{
    cItem iitm[ MAX ],
            itm[ MAX ];
    fstream file;
    clrscr();
    for( int i = 0; i < MAX; i++ )                                // < >

        iitm[ i ].ReadData( );
    Sort( iitm, MAX );
    file.open( "item.dat", ios::trunc | ios::in | ios::out | ios::binary );
    file.seekg( 0, ios::beg );
    file.clear( );
    for( i = 0; i < MAX; i++ )
        file.write( ( char * )&itm[ i ], sizeof( itm ) );
    file.seekg( 0, ios::beg );
    for( i = 0; i < MAX; i++ )
    {
        file.read( ( char * )&itm[ i ], sizeof( itm ) );
        iitm[ i ].WriteData( );
    }
    getch( );
}
/*struct cItem
{
char itcode [6];
char itname [20];
int qty;
};
struct cItem* ReadData(int n)
{
struct cItem item[MAX];
for(int i=0;i<n;i++)
{
cout<<" ITEM CODE ";
cin>>item[i].itcode;
cout<<"ITEM NAME ";
cin>>item[i].itname;
cout<<"QUANTITY ";
cin>>item[i].qty;
}
return item;
}
void WriteData(struct cItem*itm,int n)
{
for(int i=0;i<n;i++)
{
cout.setf(ios::left,ios::adjustfield);
cout<<setw(8)<<item[i].itcode<<setw(20)<<item[i].itname;
cout.setf(ios::right,ios::adjustfield);
cout<<setw(5)<<item[i].qty<<endl;
}
}
void Sort(cItem *T,int n)
{
for(int i=0;i<n-1;i++)
{
for(int j=0;j<n-1-i;j++)
{
if (strcmp (T[j].itcode, T[j+1].itcode)>0)
{
cItem tmp=T[ j];
T[ j]=T[ j+1];
}
}
}

```

```

T[j+1]=tmp;
}
}
}
}
}
void main()
{
struct cItem * ReadData(int);
void WriteData(struct cItem *,int);
void Sort(struct cItem *,int);
struct cItem *iitm;
iitm=(struct cItem*)malloc(sizeof(struct cItem)*MAX);
fstream file;
clrscr();
iitm= ReadData(MAX);
Sort (iitm,MAX);
file. open("item.dat",ios::trunc|ios::in|ios::out|ios::binary);
file.seekg(0,ios::beg);
file.clear();
for(int i=0;i<MAX;i++)
file.write ((char*)&iitm[i],sizeof (iitm));
file.seekg(0,ios::beg);
/*for(i=0;i<MAX;i++)
{
file.read((char*)&iitm[i],sizeof(iitm));
iitm[i].WriteData();
} */
//getch ();
//}

```

// Input :

```

ITEM CODE 3
ITEM NAME c
QUANTITY 9
ITEM CODE 2
ITEM NAME b
QUANTITY 8
ITEM CODE 1
ITEM NAME a
QUANTITY 7

```

// output :

1	a	7
2	b	8
3	c	9

Q.26 Write a class that implements a circular queue as a linked list.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <stdio.h>
template<class xType>class    cCQueue;
template<class xType>class  cNode
{
    friend      cCQueue< xType >;
    xType           item;
    cNode *next;
public:
    cNode( )
    {
        next = '\0';
    }
    cNode( xType  x )
    {
        item = x;
        next = '\0';
    }
    void DispNode( );
} ;
template<class xType>class cCQueue
{
    cNode< xType >      * front,
                           * rear;
public:
    cCQueue( )
    {
        rear = front = '\0';
    }
    void Add( );
    void Disp( );
    void Delete( );
} ;
template<class xType>  void cCQueue< xType >::Add( )
{
    xType  x;
    char   ch = 'y';
    while( ch == 'y' || ch == 'Y' )
    {
        cout << "Item . ";
        cin >> x;
        cNode < xType > * NewNode = new cNode < xType >( x );
        if( !front )
        {
            front      = rear = NewNode;
            NewNode->next = front;
        }
        else
        {
            NewNode->next = rear->next;
            rear->next   = NewNode;
            rear         = NewNode;
        }
        cout << "Want to Add More      '[y/n] : ";
        ch = getchar( );
        fflush( stdin );
    }
}
```

```

}

template<class xType> void cCQueue< xType >::Disp( )
{
    int flag = 0;
    for( cNode < xType > * ptr = rear->next;
          !( flag && ptr == rear->next );
          ptr = ptr->next, flag )
        ptr->DispNode( );
}

template<class xType> void cNode< xType >::DispNode( )
{
    cout << item << endl;
}

template<class xType> void cCQueue< xType >::Delete( )
{
    if( front )
    {
        cNode< xType > * ptr = rear->next;
        rear->next = front->next;
        front = front->next;
        delete ptr;
    }
    else
        cout << "Queue is empty !.....Abortng";
}

void main( )
{
    cCQueue< int > Q;
    Q.Add( );
    clrscr( );
    cout << "Queue is . " << endl;
    Q.Disp( );
    Q.Delete( );
    Q.Delete( );
    Q.Delete( );
    Q.Delete( );
    cout << "Queue is :" << endl;
    Q.Disp( );
}

// INPUT :

Item . 1
Want to Add More      '[y/n] : Y
Item . 2
Want to Add More      '[y/n] : Y
Item . 3
Want to Add More      '[y/n] : Y
Item . 4
Want to Add More      '[y/n] : N

// OUTPUT :
1
2
3
4
1

```

Q.27 Write a class that implements a Bubble Sorting algorithm on a set of 25 numbers.

// Example Code :

```
#include      <iostream.h>
#include      <conio.h>
#include      <iomanip.h>
class cArray
{
    int num[ 5 ];
public:
    cArray( );
    friend      ostream & operator <<( ostream &, cArray & );
    friend      istream & operator >>( istream &,cArray & );
    void        sort( );
} ;
cArray::cArray( )
{
    for( int i = 0; i < 5; i++ )                                // < >
        num[ i ] = 0;
}
ostream & operator <<( ostream & out, cArray & arr )
{
    out << "Elements of the Array are : " << endl;
    for( int i = 0; i < 5; i++ )                                // < >
    {
        out << arr.num[ i ] << endl;
    }
}
istream & operator >>( istream & in, cArray & arr )
{
    cout << "Enter Elements of the Array :      " << endl;
    for( int i = 0; i < 5; i++ )                                // < >
    {
        in >> arr.num[ i ];
    }
}
void cArray::sort( )
{
    for( int i = 1; i < 5; i++ )                                // < >
    {
        for( int j = 0; j < 5 - i; j++ )                        // < >
        {
            if( num[ j ] > num[ j + 1 ] )
            {
                int tmp = num[ j ];
                num[ j ]      = num[ j + 1 ];
                num[ j + 1 ] = tmp;
            }
        }
    }
}

void main( )
{
    cArray      arr;
```

```
clrscr( );
cin >> arr;
arr.sort( );
cout << arr;
getch( );
}
```

INPUT

```
Enter
Elements of the Array :
2
3
4
5
0
```

OUTPUT

```
Elements
of the Array are :
0
2
3
4
5
```

Q.28 A class has two member functions as Encrypt() and Decrypt(). Implement this class to encrypt a string and decrypt it using your own method for encryption and decryption.

// Example Code :

```
#include      <iostream.h>
#include      <conio.h>
#include      <iomanip.h>
#include      <stdio.h>
class cString
{
    char *      str;
    int   len;
public:
    cString( )
    {
        str = '\0';
        len = 0;
    }
    cString( char * );
    void Encrypt( );
    void Decrypt( );
    void GetStr( );
    void DispStr( );
}
cString::cString( char * S )
{
    for( int i = 0; S[ i ]; i++ )                                // < >

        ;
    len = i;
    str = new char [len + 1];
    int j = 0;
    for( i = 0; S[ i ]; i++ )
        str[ j++ ] = S[ i ];
    str[ j ] = '\0';
}
void cString::GetStr( )
{
    cout << "Length of String :";
    cin >> len;
    str = new char [len + 1];
    gets( str );
    fflush( stdin );
}
void cString::DispStr( )
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( len ) << str << endl;
}
void cString::Encrypt( )
{
    for( int i = 0; i < len; i++ )                                // < >

        str[ i ] += ( i + 100 - ( i * 2 ) );
}
void cString::Decrypt( )
{
    for( int i = 0; i < len; i++ )                                // < >

        {
            str[ i ] -= ( i + 100 - ( i * 2 ) );
        }
}
```

```

        }
    }
void main( )
{
    cString S1 ( "Mickey Donald 1234 *%&~@" );
    clrscr( );
    S1.GetStr( );
    cout << "Input String :";
    S1.DispStr( );
    S1.Encrpt( );
    cout << "Encrypted String : ";
    S1.DispStr( );
    S1.Decrpt( );
    cout << "decrypted String : ";
    S1.DispStr( );
    getch( );
}

// INPUT :
Length
of String :4
DSDA

// OUTPUT :

Input String :DSDA
Encrypted String :
Decrypted
string:
DSDA

```

Q.29 Write a program for linked list, which will have the facility to add, modify and delete the records from linked list. The fields to be accepted from the user include

Employee No	Numeric
Name	String
Department	String

Option should be made to view the list generated. Modularize the code to perform the above actions.

// Example Code :

```
#include    <iostream.h>
#include    <conio.h>
#include    <iomanip.h>
#include    <stdio.h>
#include    <string.h>
class cNode
{
    int      eno;
    char    name[ 15 ];
    char    dept[ 10 ];
    cNode   * next;
public:
    cNode( );
    cNode( int, char *, char * );
    void MakeNode( int, char *, char * );
    void PutNext( cNode * );
    int GetEno( )
    {
        return eno;
    }
    char * GetName( )
    {
        return name;
    }
    char * GetDept( )
    {
        return dept;
    }
    cNode   * GetNext( );
    void    DispNode( );
} ;
class cList
{
    cNode   * start,
            * last;
    int      TotNodes;
public:
    cList( )
    {
        start = last = '\0';
        TotNodes = 0;
    }
    void Add( );
    void Add( int, char *, char * );
    void Modify( );
    void Modify( int );
    void Delete( );
    void Delete( int );
    void DispList( );
} ;
cNode::cNode( )
```

```

{
    eno = 0;
    strcpy( name, '\0' );
    strcpy( dept, '\0' );
    next = '\0';
}

cNode::cNode( int a, char * nm, char * dp )
{
    eno = a;
    strcpy( name, nm );
    strcpy( dept, dp );
    next = '\0';
}
void cNode::MakeNode( int a, char * nm, char * dp )
{
    eno = a;
    strcpy( name, nm );
    strcpy( dept, dp );
    next = '\0';
}
void cNode::PutNext( cNode * ptr )
{
    next = ptr;
}
cNode * cNode::GetNext( )
{
    return next;
}
void cNode::DispNode( )
{
    cout << setw( 4 ) << eno;
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 15 ) << name << setw( 10 ) << dept << endl;
}
void cList::Add( )
{
    int xeno;
    char xname[ 15 ];
    char xdept[ 10 ];
    while( 1 )
    {
        cout << "Emp. No. (0-Exit) ";
        cin >> xeno;
        if( xeno == 0 )
            break;
        cout << "Name of Employee . ";
        gets( xname );
        fflush( stdin );
        cout << "Department . ";
        gets( xdept );
        fflush( stdin );
        cNode * NewNode = new cNode;
        NewNode->MakeNode( xeno, xname, xdept );
        if( !start )
            start = last = NewNode;
        else
        {
            last->PutNext( NewNode );
            last = NewNode;
        }
    }
}

```

```

void cList::Add( int a, char * nm, char * dp )
{
    cNode * NewNode = new cNode;
    NewNode->MakeNode( a, nm, dp );
    if( !start )
        start = last = NewNode;
    else
    {
        last->PutNext( NewNode );
        last = NewNode;
    }
    TotNodes++;
}
void cList::Modify( )
{
    int xeno;
    char xname[ 15 ];
    char xdept[ 10 ];
    cout << "Employee Number to be Modified : ";
    cin >> xeno;
    for( cNode * ptr = start; // < >

        ptr && ptr->GetEno( ) != xeno;
        ptr = ptr->GetNext( ) )
    ;
    if( ptr )
    {
        cout << "Enter name      " ;
        gets( xname );
        fflush( stdin );
        cout << "Enter Department      " ;
        gets( xdept );
        fflush( stdin );
        strcpy( ptr->GetName( ), xname );
        strcpy( ptr->GetDept( ), xdept );
    }
    else
    {
        textattr( 136 );
        gotoxy( 30, 10 );
        printf( "List not existing or Record not found\n" );
    }
}
void cList::Modify( int x )
{
    char xname[ 15 ];
    char xdept[ 10 ];
    for( cNode * ptr = start; // < >

        ptr && ptr->GetEno( ) != x;
        ptr = ptr->GetNext( ) )
    ;
    if( ptr )
    {
        cout << "Enter name:" ;
        gets( xname );
        fflush( stdin );
        cout << "Enter Department:" ;
        gets( xdept );
        fflush( stdin );
        strcpy( ptr->GetName( ), xname );
        strcpy( ptr->GetDept( ), xdept );
    }
    else

```

```

    {
        textattr( 153 );
        gotoxy( 30, 10 );
        cprintf( "List not existing or Record not found\n" );
    }
}

void cList::Delete( )
{
    int      xeno;
    cNode   * prev;
    char     ch  = 'n';
    cout << "Employee Number to be Deleted :";
    cin >> xeno;
    for( cNode * ptr  = start;                                //  < >

          ptr && ptr->GetEno( ) != xeno;
          prev = ptr, ptr = ptr->GetNext( ) )
    ;
    if( ptr && ptr->GetEno( ) == xeno )
    {
        ptr->DispNode( );
        cout << "Confirm Deletion (y/n) ?";
        ch = getchar();
        fflush( stdin );
        if( ch == 'Y' || ch == 'Y' )
        {
            if( ptr == start )
                start = start->GetNext( );
            else
                prev->PutNext( ptr->GetNext( ) );
            delete ( ptr );
        }
    }
    else
    {
        textattr( 150 );
        gotoxy( 35, 10 );
        cprintf( "List not existing or record not found" );
    }
}
void cList::Delete( int  x )
{
    cNode   * prev;
    char     ch  = 'n';
    for( cNode * ptr  = start;                                //  < >

          ptr && ptr->GetEno( ) != x;
          prev = ptr, ptr = ptr->GetNext( ) )
    ;
    if( ptr && ptr->GetEno( ) == x )
    {
        ptr->DispNode( );
        cout << "Confirm Deletion (y/n)?";
        ch = getchar();
        fflush( stdin );
        if( ch == 'Y' || ch == 'Y' )
        {
            if( ptr == start )
                start = start->GetNext( );
            else
                prev->PutNext( ptr->GetNext( ) );
            delete ( ptr );
        }
    }
}

```

```

    else
    {
        textattr( 150 );
        gotoxy( 35, 10 );
        cprintf( "List not existing or Record not found" );
    }
}
void cList::DispList( )
{
    cNode * ptr;
    for( ptr = start; ptr; ptr = ptr->GetNext( ) )
        ptr->DispNode( );
}
void main( )
{
    cList L;
    clrscr( );
    L.Add( 1, "Mickey", "Computer" );
    L.Add( 2, "Ajay", "Computer" );
    L.Add( 3, "Sushil", "Computer" );
    L.Add( 4, "Donald", "Computer" );
    L.DispList( );
    L.Modify( 3 );
    L.DispList( );
    getch( );
}

```

// INPUT & OUTPUT :

```

1      Mickey           Computer
2      Ajay            Computer
3      Sushil          Computer
4      Donald          Computer
Employee Number to be Modified : 1
Enter name      ä shilpa
Enter Department      CSE
1      shilpa         CSE
2      Ajay            Computer
3      Sushil          Computer
4      Donald          Computer
Employee Number to be Deleted :3
3      Sushil          Computer
Confirm Deletion (y/n) ?y
1      shilpa         CSE
2      Ajay            Computer
4      Donald          Computer

```

Q.30 Write a class that has the functionality to write and read the data from a file line by line.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <fstream.h>
#include      <stdio.h>
#include      <string.h>

class cWrtnl
{
    char str[ 80 ];
public:
    cWrtnl( )
    {
        strcpy( str, '\0' );
    }
    void Read( );
    void Write( );
}
void cWrtnl::Read( )
{
    cout << "Line" << endl;
    gets( str );
    fflush( stdin );
}
void cWrtnl::Write( )
{
    cout << str << endl;
}
void main( )
{
    clrscr( );
    cWrtnl      S;
    int   cnt   = 0;
    char  ans   = 'Y';
    fstream  F;
    F.open( "data.dat", ios::in | ios::out | ios::app | ios::binary );
    while( ans == 'y' || ans == 'Y' )
    {
        S.Read( );
        F.write( ( char * )&S, sizeof( S ) );
        F.seekg( ( cnt * 80 ) + 1, ios::beg );
        cnt++;
        cout << "Want to Continue (y/n):";
        ans = getchar( );
        fflush( stdin );
    }
    F.seekg( 0 );
    for( int i = 0; i < cnt; i++ )                                //  < >
    {
        F.read( ( char * )&S, sizeof( S ) );
        S.Write( );
    }
    F.close( );
    getch( );
}
```

```
// INPUT :  
  
Line  
FDS  
Want  
to Continue (y/n):Y  
Line  
VFBFB  
Want to Continue (y/n):N
```

```
// OUTPUT :
```

```
FDS  
VFBFB
```

Q.31 Write a program that performs the matrix manipulations such as addition and deletion. The matrix class will have the member functions Create, Add, Delete, Display.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cMatrix
{
    int **      mat;
    int     r,
            c;
    void CreateMat( );
    void InitMat( );
public:
    cMatrix( )
    {
        mat = '\0';
        r     = c = 0;
    }
    friend     istream & operator >>( istream &,
                                         cMatrix & );
    friend     ostream & operator <<( ostream &,
                                         cMatrix & );
    cMatrix     Add( cMatrix & );
    cMatrix     operator +( cMatrix & );
    cMatrix     operator +( int );
    cMatrix     Sub( cMatrix & );
    cMatrix     operator -( cMatrix & );
    cMatrix     operator -( int );
    void       DispMat( );
}
void cMatrix::CreateMat( )
{
    mat = new int*[r];
    for( int i = 0; i < r; i++ )                                // < >
        mat[ i ] = new int [c];
}
void cMatrix::InitMat( )
{
    for( int i = 0; i < r; i++ )                                // < >
    {
        cout << "Row #" << setw( 2 ) << i + 1 << endl;
        for( int j = 0; j < c; j++ )                            // < >
        {
            cout << "Col #" << setw( 2 ) << j + 1 << ":";
            cin >> mat[ i ][ j ];
        }
    }
}
void cMatrix::DispMat( )
{
    cout.setf( ios::left, ios::adjustfield );
    for( int i = 0; i < r; i++ )                                // < >
    {
```

```

        for( int j = 0; j < c; j++ )                                // < >
            cout << setw( 5 ) << mat[ i ][ j ];
        cout << endl;
    }
}
cMatrix cMatrix::Add( cMatrix & M2 )
{
    cMatrix tmp;
    tmp.r = r;
    tmp.c = c;
    if( r == M2.r && c == M2.c )
    {
        tmp.CreateMat( );
        for( int i = 0; i < r; i++ )                                // < >
        {
            for( int j = 0; j < c; j++ )                            // < >
                tmp.mat[ i ][ j ] = mat[ i ][ j ] + M2.mat[ i ][ j ];
        }
    }
    return tmp;
}
cMatrix cMatrix::operator +( cMatrix & M2 )
{
    cMatrix tmp;
    tmp.r = r;
    tmp.c = c;
    if( r == M2.r && c == M2.c )
    {
        tmp.CreateMat( );
        for( int i = 0; i < r; i++ )                                // < >
        {
            for( int j = 0; j < c; j++ )                            // < >
                tmp.mat[ i ][ j ] = mat[ i ][ j ] + M2.mat[ i ][ j ];
        }
    }
    return tmp;
}
cMatrix cMatrix::Sub( cMatrix & M2 )
{
    cMatrix tmp;
    tmp.r = r;
    tmp.c = c;
    if( r == M2.r && c == M2.c )
    {
        tmp.CreateMat( );
        for( int i = 0; i < r; i++ )                                // < >
        {
            for( int j = 0; j < c; j++ )                            // < >
                tmp.mat[ i ][ j ] = mat[ i ][ j ] - M2.mat[ i ][ j ];
        }
    }
    return tmp;
}
cMatrix cMatrix::operator -( cMatrix & M2 )
{
    cMatrix tmp;
    tmp.r = r;

```

```

tmp.c = c;
if( r == M2.r && c == M2.c )
{
    tmp.CreateMat( );
    for( int i = 0; i < r; i++ )                                // < >

    {
        for( int j = 0; j < c; j++ )                            // < >
    }
}
return tmp;
}
cMatrix cMatrix::operator +( int x )
{
    cMatrix tmp;
    tmp.r = r;
    tmp.c = c;
    tmp.CreateMat( );
    for( int i = 0; i < r; i++ )                                // < >

    {
        for( int j = 0; j < c; j++ )                            // < >
            tmp.mat[ i ][ j ] = mat[ i ][ j ] + x;
    }
    return tmp;
}
cMatrix cMatrix::operator -( int x )
{
    cMatrix tmp;
    tmp.r = r;
    tmp.c = c;
    tmp.CreateMat( );
    for( int i = 0; i < r; i++ )                                // < >

    {
        for( int j = 0; j < c; j++ )                            // < >
            tmp.mat[ i ][ j ] = mat[ i ][ j ] - x;
    }
    return tmp;
}
istream & operator >>( istream & In, cMatrix & M )
{
    cout << "Number of Rows .";
    cin >> M.r;
    cout << "Number of Cols .";
    cin >> M.c;
    M.CreateMat( );
    M.InitMat( );
    return In;
}
ostream & operator <<( ostream & Out, cMatrix & M1 )
{
    for( int i = 0; i < M1.r; i++ )                                // < >

    {
        for( int j = 0; j < M1.c; j++ )                            // < >
    {
        cout << M1.mat[ i ][ j ] << "\t";
    }
}

```

```

        cout << endl;
    }
    return Out;
}
void main( )
{
    cMatrix M1,
            M2,
            M3;
    clrscr( );
    cin >> M1 >> M2;
    clrscr( );
    cout << M1 << M2;
    M3 = M1 + M2;
    cout << M3;
    getch( );
}

```

// INPUT & OUTPUT :

Number of Rows .3

Number of Cols .3

Row # 1

Col # 1:1

Col # 2:1

Col # 3:1

Row # 2

Col # 1:1

Col # 2:1

Col # 3:1

Row # 3

Col # 1:1

Col # 2:1

Col # 3:1

Number of Rows .3

Number of Cols .3

Row # 1

Col # 1:1

Col # 2:1

Col # 3:1

Row # 2

Col # 1:1

Col # 2:1

Col # 3:1

Row # 3

Col # 1:1

Col # 2:1

Col # 3:1

MATRIX 1

1	1	1
1	1	1
1	1	1

MATRIX 2

1	1	1
1	1	1
1	1	1

Matrix after addition

2	2	2
2	2	2
2	2	2

Q.32 Write a program that will print the largest and smallest number from and 5 row by 5 column matrix. The matrix class will have the member functions as Create, Display, Largest, Smallest.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include<conio.h> #include<string.h>
class cMatrix
{
    int mat[ 3 ][ 3 ];
    void InitMat( );
public:
    cMatrix( )
    {
    }
    friend      istream & operator >>( istream &,
                                         cMatrix & );
    friend      ostream & operator <<( ostream &,
                                         cMatrix & );
    int         Largest( );
    int         Smallest( );
    void        DispMat( );
}
int cMatrix::Largest( )
{
    int max   = mat[ 0 ][ 0 ];
    for( int i = 0; i < 3; i++ )                                // < >
    {
        for( int j = 0; j < 3; j++ )                                // < >
        {
            if( mat[ i ][ j ] >= max )
                max = mat[ i ][ j ];
        }
    }
    return max;
}
int cMatrix::Smallest( )
{
    int min   = mat[ 0 ][ 0 ];
    for( int i = 0; i < 3; i++ )                                // < >
    {
        for( int j = 0; j < 3; j++ )                                // < >
        {
            if( mat[ i ][ j ] <= min )
                min = mat[ i ][ j ];
        }
    }
    return min;
}
void cMatrix::InitMat( )
{
    for( int i = 0; i < 3; i++ )                                // < >
    {
        cout << "Row #" << setw( 2 ) << i + 1 << endl;
    }
}
```

```

        for( int j = 0; j < 3; j++ )                                // < >
    {
        cout << "Col #" << setw( 2 ) << j + 1 << ":";
        cin >> mat[ i ][ j ];
    }
}
void cMatrix::DispMat( )
{
    cout.setf( ios::left, ios::adjustfield );
    for( int i = 0; i < 3; i++ )                                // < >
    {
        for( int j = 0; j < 3; j++ )                                // < >
            cout << setw( 5 ) << mat[ i ][ j ];
        cout << endl;
    }
}
istream & operator >>( istream & In, cMatrix & M )
{
    M.InitMat( );
    return In;
}
ostream & operator <<( ostream & Out, cMatrix & M1 )
{
    for( int i = 0; i < 3; i++ )                                // < >
    {
        for( int j = 0; j < 3; j++ )                                // < >
        {
            cout << M1.mat[ i ][ j ] << "\t";
        }
        cout << endl;
    }
    return Out;
}
void main( )
{
    cMatrix M1;
//clrscr ();
    cin >> M1;
    clrscr( );
    cout << M1;
    int m = M1.Largest( );
    cout << endl << "Maximum is . " << m;
    int m1 = M1.Smallest( );
    cout << endl << "Smallest is . " << m1;
    getch( );
}

// INPUT :
Row # 1
Col # 1:1
Col # 2:2
Col # 3:3
Row # 2
Col # 1:4
Col # 2:5
Col # 3:6

```

```
Row # 3
Col # 1:3
Col # 2:8
Col # 3:3
```

```
// OUTPUT :
```

1	2	3
4	5	6
3	8	3

```
Maximum is . 8
Smallest is . 1
```

Q.33 Write a template class for sorting method. Using this class, write a test program for sorting using different data types.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
template<class cType> void Sort( cType * arr, int      n )
{
    cType   tmp;
    for( int  i  = 0; i < n - 1; i++ )                                //  < >

    {
        for( int  j  = 0; j < n - 1 - i; j++ )                          //  < >

        {
            if( arr[ j ] > arr[ j + 1 ] )
            {
                tmp          = arr[ j ];
                arr[ j ]      = arr[ j + 1 ];
                arr[ j + 1 ] = tmp;
            }
        }
    }
}
void main( )
{
    char *      arr;
    int   n;
    clrscr( );
    cout << "enter no. of elements : ";
    cin >> n;
    arr = new char [n];
    cout << "Enter Elements now \n";
    for( int  i  = 0; i < n; i++ )                                     //  < >

        cin >> arr[ i ];
    Sort( arr, n );
    cout << "sorted list is" << endl;
    for( i = 0; i < n; i++ )
        cout << arr[ i ] << endl;
    getch( );
}
// INPUT :
enter
no. of elements : 6
Enter Elements now
2
7
1
4
9
3
// OUTPUT :
sorted
list is
1
2
3
4
7
9
```

Q.34 Write a program that will convert the given decimal input in to other number systems Hexadecimal, Octal, and Binary. The class Conversion will have the methods as GetNumber, PrintNumber, ConvertToHexadecimal, ConvertToOctal, ConvertToBinary.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cNumber
{
    long      num;
    char con[ 30 ];
public:
    cNumber( )
    {
        num = 0;
    }
    void GetNum( );
    void DispNum( );

    void Hex( );
    void HeX( );
    void Oct( );
    void Bin( );
}
void cNumber::GetNum( )
{
    cout << "Enter any decimal :";
    cin >> num;
}
void cNumber::DispNum( )
{
    cout << num << " in converted form is " << con << endl;
}
void cNumber::HeX( )

{
    int i = 0;
    long num1;
    num1 = num;
    while( num1 )
    {
        if( num1 % 16 <= 9 )
            con[ i++ ] = num1 % 16 + 48;
        else
            switch( num1 % 16 )
            {
                case 10:
                    con[ i++ ] = 'A';
                    break;
                case 11:
                    con[ i++ ] = 'B';
                    break;
                case 12:
                    con[ i++ ] = 'C';
                    break;
                case 13:
                    con[ i++ ] = 'D';
                    break;
                case 14:
```

```

        con[ i++ ] = 'E';
        break;
    case 15:
        con[ i++ ] = 'F';
        break;
    }
    num1 = num1 / 16;
}
con[ i ] = '\0';
int j = 0;
for( i = 0; con[ i ]; i++ )
;
for( i--; j <= i; i--, j++ )
{
    int tmp = con[ j ];
    con[ j ] = con[ i ];
    con[ i ] = tmp;
}
}
void cNumber::Hex( )
{
    int i = 0;
    long num1;
    num1 = num;
    while( num1 )
    {
        if( num1 % 16 <= 9 )
            con[ i++ ] = num1 % 16 + 48;
        else
            switch( num1 % 16 )
            {
                case 10:
                    con[ i++ ] = 'a';
                    break;
                case 11:
                    con[ i++ ] = 'b';
                    break;
                case 12:
                    con[ i++ ] = 'c';
                    break;
                case 13:
                    con[ i++ ] = 'd';
                    break;
                case 14:
                    con[ i++ ] = 'e';
                    break;
                case 15:
                    con[ i++ ] = 'f';
                    break;
            }
        num1 = num1 / 16;
    }
    con[ i ] = '\0';
    int j = 0;
    for( i = 0; con[ i ]; i++ )
    ;
    for( i--; j <= i; i--, j++ )
    {
        int tmp = con[ j ];
        con[ j ] = con[ i ];
        con[ i ] = tmp;
    }
}
void cNumber::Oct( )

```

```

{
    int i = 0;
    long num1;
    num1 = num;
    while( num1 )
    {
        con[ i++ ] = ( num1 % 8 ) + 48;
        num1      = num1 / 8;
    }
    con[ i ] = '\0';
    int j = 0;
    for( i = 0; con[ i ]; i++ )
    ;
    for( i--; j <= i; i--, j++ )
    {
        int tmp = con[ j ];
        con[ j ] = con[ i ];
        con[ i ] = tmp;
    }
}
void cNumber::Bin( )
{
    int i = 0;
    long num1;
    num1 = num;
    while( num1 )
    {
        con[ i++ ] = ( num1 % 2 ) + 48;
        num1      = num1 / 2;
    }
    con[ i ] = '\0';
    int j = 0;
    for( i = 0; con[ i ]; i++ )
    ;
    for( i--; j <= i; i--, j++ )
    {
        int tmp = con[ j ];
        con[ j ] = con[ i ];
        con[ i ] = tmp;
    }
}
void main( )
{
    cNumber C;
    clrscr( );
    C.GetNum( );
    C.Hex( );
    CDispNum( );
    C.HeX( );
    CDispNum( );
    C.Oct( );
    CDispNum( );
    C.Bin( );
    CDispNum( );
    getch( );
}

// INPUT :
Enter
any decimal :24

// OUTPUT :

```

24 in converted HEXADECIMAL form is 18  
24 in converted OCTAL form is 30  
24 in converted BINARY form is 11000

Q.35 Write a class Time that will have the member functions to calculate the difference and addition for given two time values. It will store the time in hours(0-31), minutes (0-59), and seconds (0-59).

```
// Example Code :  
  
#include      <iostream.h>  
#include      <iomanip.h>  
#include      <conio.h>  
#include      <string.h>  
class cTime  
{  
    int hh,  
        mm,  
        ss;  
public:  
    cTime( )  
    {  
        hh = ss = mm = 0;  
    }  
    cTime( int a, int b, int c )  
    {  
        hh = a;  
        mm = b;  
        ss = c;  
    }  
    int IsValid( );  
    friend istream & operator >>( istream &, cTime & );  
    friend ostream & operator <<( ostream &, cTime & );  
    cTime operator +( cTime & );  
    cTime operator -( cTime & );  
};  
istream & operator >>( istream & In, cTime & T )  
{  
    cout << "Hours      :" ;  
    In >> T.hh;  
    cout << "Minutes   :" ;  
    In >> T.mm;  
    cout << "Seconds   :" ;  
    In >> T.ss;  
    return In;  
}  
ostream & operator <<( ostream & Out, cTime & T )  
{  
    cout.fill( '0' );  
    Out << setw( 2 ) << T.hh << ":" << setw( 2 ) << T.mm << ":" << setw( 2 )  
    << T.ss << endl;  
    return Out;  
}  
cTime cTime::operator +( cTime & T2 )  
{  
    cTime tmp;  
    tmp.hh = hh + T2.hh;  
    tmp.mm = mm + T2.mm;  
    if( tmp.mm > 59 )  
    {  
        tmp.hh++;  
        tmp.mm -= 60;  
    }  
}
```

```

        tmp.ss = ss + T2.ss;
        if( tmp.ss > 59 )
        {
            tmp.mm++;
            tmp.ss -= 60;
        }
        return tmp;
    }
cTime cTime::operator -( cTime & T2 )
{
    cTime tmp;
    if( hh >= T2.hh )
    {
        tmp.hh = hh - T2.hh;
        tmp.mm = mm - T2.mm;
        if( tmp.mm < 0 )
        {
            tmp.hh--;
            tmp.mm += 60;
        }
        tmp.ss = ss - T2.ss;
        if( tmp.ss < 0 )
        {
            tmp.mm--;
            tmp.ss += 60;
        }
    }
    return tmp;
}
int cTime::isValid( )
{
    if( hh <= 23 && hh >= 0 && mm <= 59 && mm >= 0
        && ss <= 59 && ss >= 0 )
        return 1;
    return 0;
}
void main( )
{
    cTime T1,
                T2,
                T3,
                T4;
    clrscr( );
    cin >> T1 >> T2;
    if( T1.IsValid( ) && T2.IsValid( ) )
    {
        T3 = T1 + T2;
        T4 = T1 - T2;
        cout << "time 1 - " << T1 << "time 2 - " << T2 << "Addition - "
<< T3
                << "Subtraction - " << T4;
    }
    else
        cout << "invalid Time entered :";
    getch( );
}

// INPUT :
Hours :
2

```

```
Minutes:  
34  
Seconds:  
11  
Hours:  
3  
Minutes:  
20  
Seconds:  
33  
  
// OUTPUT :  
  
time  
1 - 02: 34:11  
time 2 - 03: 20:33  
Addition - 05: 54:44  
Subtraction - 00: 00:00
```

Q.36 Write a class to handle fractions such as "1/3". Define Addition, Subtraction, Multiplication, and Division operators for these fractions.  
(Hint: use operator overloading)

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cFraction
{
    int num,
        den;
    int LCM( int, int );
public:
    cFraction( )
    {
        num = den = 1;
    }
    cFraction( int     a, int     b = 1 )
    {
        num = a;
        den = b;
    }
    friend    istream   & operator >>( istream &,
                                         cFraction & );
    friend    ostream    & operator <<( ostream &,
                                         cFraction & );
    cFraction    operator +( cFraction & );
    cFraction    operator -( cFraction & );
    cFraction    operator *( cFraction & );
    cFraction    operator /( cFraction & );
    cFraction    Simplify( );
    int          IsValid( );
}
cFraction cFraction::operator *( cFraction & F )
{
    cFraction tmp;
    tmp.num = num * F.num;
    tmp.den = den * F.den;
    return tmp.Simplify();
}
cFraction cFraction::operator /( cFraction & F )
{
    cFraction tmp;
    tmp.num = num * F.den;
    tmp.den = den * F.num;
    return tmp.Simplify();
}
cFraction cFraction::Simplify( )
{
    cFraction tmp;
    int     j = 0,
          tt,
          k = 0,
          t2;
    if( num > den )
    {
        tt = num;
        t2 = den;
    }
}
```

```

    else
    {
        tt = den;
        t2 = num;
    }
    for( int i = 2; i < tt + 1; i++ )           // < >

    {
        if( tt % i == 0 )
        {
            if( t2 % i == 0 )
            {
                tt /= i;
                t2 /= i;
            }
        }
    }
    if( num > den )
    {
        tmp.num = tt;
        tmp.den = t2;
    }
    else
    {
        tmp.den = tt;
        tmp.num = t2;
    }
    return tmp;
}

cFraction cFraction::operator +( cFraction & F )
{
    cFraction tmp;
    int lc = LCM( den, F.den );
    tmp.num = num * ( lc / den ) + F.num * ( lc / F.den );
    tmp.den = lc;
    return tmp.Simplify();
}

cFraction cFraction::operator -( cFraction & F )
{
    cFraction tmp;
    int lc = LCM( den, F.den );
    tmp.num = num * ( lc / den ) - F.num * ( lc / F.den );
    tmp.den = lc;
    return tmp.Simplify();
}
int cFraction::IsValid( )
{
    if( this->den == 0 )
        return 0;
    return 1;
}
int cFraction::LCM( int num1, int num2 )
{
    int did,
        div,
        lcm;
    if( num1 > 0 && num2 > 0 )
    {
        if( num1 > num2 )
        {
            did = num1;
            div = num2;

```

```

        }
    else
    {
        did = num2;
        div = num1;
    }
    int rem = did % div;
    while( rem )
    {
        did = div;
        div = rem;
        // hAre div is H.C.F.
        rem = did % div;
    }
    lcm = ( num1 * num2 ) / div;
}
else
    lcm = 0;
return lcm;
}
istream & operator >>( istream & In, cFraction & F )
{
    cout << "Numerator      :      ";
    In >> F.num;
    cout << "Denomenator     :      ";
    In >> F.den;
    return In;
}
ostream & operator <<( ostream & Out, cFraction & F )
{
    if( F.num == 0 )
        Out << F.num << endl;
    else if( F.den == 1 )
        cout << F.num << endl;
    else if( F.num == 1 && F.den == 1 )
        cout << 1 << endl;
    else if( F.den == 0 )
        cout << "Error: Divide by Zero" << endl;
    else
        Out << F.num << "/" << F.den << endl;
    return Out;
}
void main( )
{
    cFraction f1 ( 1, 4 ),
               f2 ( 1, 5 ),
               f3,
               f4;
    clrscr( );
    cout << "fraction 1 : 1/4" << endl;
    cout << "fraction 2 : 1/5";
    f3 = f1 + f2;
    f4 = f3 - f2;
    cout << f1 << f2 << "Add :" << f3 << f4;
    f3 = f2 * f1;
    f4 = f1 / f2;
    cout << "multiply" << f3 << f4;
    getch( );
}

// output :
fraction

```

```
1 : 1/4
fraction 2 : 1/5
Add:
9/20
1/4
multiply:
1/20
5/4
```

Q.37 Write a Date class that allows you to add, subtract, read and print simple dates of the form dd/mm/yy. Considering leap year calculations will be an added advantage.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <stdio.h>
#include      <conio.h>
class cDate
{
    int dd,
        mm,
        yy;
public:
    cDate( )
    {
        dd = mm = yy = 0;
    }
    cDate( int a, int b, int c )
    {
        dd = a;
        mm = b;
        yy = c;
    }
    friend    istream & operator >>( istream &, cDate & );
    friend    ostream & operator <<( ostream &, cDate & );
    int       operator -( cDate & );
    void      operator +( int );
}
istream & operator >>( istream & In, cDate & D )
{
    In >> D.dd >> D.mm >> D.yy;
    return In;
}
ostream & operator <<( ostream & Out, cDate & D )
{
    Out << D.dd << "/" << D.mm << "/" << D.yy << endl;
    return Out;
}
int cDate::operator -( cDate & D2 )
{
    int doyTable[]={0,0,31,59,90,120,151,182,212,243,273,304,334,365};
    int days1 = 0,
        days2 = 0;
    days1 += doyTable[ mm ] + dd;
    days1 = days1 + ( yy - 1 ) / 4;
    days1 = days1 - ( yy - 1 ) / 100;
    days1 = days1 + ( yy - 1 ) / 400;
    days1 += ( yy - 1 ) * 365;

    days2 += doyTable[ D2.mm ] + D2.dd;
    days2 = days2 + ( D2.yy - 1 ) / 4;
    days2 = days2 - ( D2.yy - 1 ) / 100;
    days2 = days2 + ( D2.yy - 1 ) / 400;
    days2 += ( D2.yy - 1 ) * 365;

    return ( days2 - days1 >= 0 )
           ?( days2 - days1 ): ( days1 - days2 );
}
void cDate::operator +( int x )
{
    int NoDays = 0;
```

```

dd += x;
switch( mm )
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        NoDays += 31;
        break;
    case 2:
    {
        if( ( ( YY ) % 4 == 0 )
            && ( ( YY ) % 100 != 0 ) )
            || ( ( YY ) % 400 == 0 ) )
            NoDays += 1;
        NoDays += 28;
    }
    break;
    case 4:
    case 6:
    case 9:
    case 11:
        NoDays += 30;
        break;
}
if( ( dd ) > NoDays )
{
    dd -= NoDays;
    mm++;
    if( mm > 12 )
    {
        mm = mm % 12;
        YY += mm / 12;
        YY++;
    }
}
void main( )
{
    cDate D1,
           D2;
    int x;
    clrscr( );
    cout << "Enter I Date (dd-mm-yy) .";
    cin >> D1;
    cout << "Enter II Date (dd-mm-yy) .";
    cin >> D2;
    cout << D1 << D2;
    cout << "Diff = " << D1 - D2;
    cout << "No. of Days to be added :";
    cin >> x;
    D1 + x;
    cout << "date after addition : " << D1;
}

// INPUT :
Enter
I Date      (dd-mm-yy) .24
07
86
Enter II Date     (dd-mm-yy) .12

```

12  
85

// OUTPUT :

24/7/86

12/12/85

Diff = 225No. of Days to be added :45

```

Q.38 Define a "string_match" base class .
class string_matcher
{
public :
// Returns TRUE if the string matches, false if not.
int match(const char string);
.....
}
Define the derived classes that match words, numbers and strings.
// Example Code :

#include      <iostream.h>
#include      <conio.h>
#include      <iomanip.h>
#include      <stdio.h>
class String_Matcher
{
    char *      str;
    int   len;
public:
    String_Matcher( )
    {
        str = '\0';
        len = 0;
    }
    String_Matcher( char * S )
    {
        int j = 0;
        for( int i = 0; S[ i ]; i++ )                                // < >

            ;
        len = i;
        str = new char [len + 1];
        for( i = 0; S[ i ]; i++ )
            ;
        str[ j++ ] = S[ i ];
        str[ j ] = '\0';
    }
    int match( const char * );
}
int String_Matcher::match( const char * string )
{
    for( int i = 0;                                         // < >

        string[ i ] == str[ i ] && str[ i ]
        && string[ i ]; i++ )
    ;
    if( string[ i ] != str[ i ] )
        return 0;
    else
        return 1;
}
void main( )
{
    String_Matcher     S1 ( "Mickey" ),S2;
    S2 = "donald";
    clrscr( );
    if( S1.match( "mickey" ) )
        cout << "same";
    else
        cout << "diff";
    getch( );
}
// OUTPUT : diff

```

Q.39 Write a base Class Number that holds a single integer value and contains one member function (print it). Define three derived classes to print the value in hex, octal, and binary.

```
// Example Code :  
  
#include      <iostream.h>  
#include      <iomanip.h>  
#include      <conio.h>  
#include      <stdio.h>  
class cNumber  
{  
protected:  
    int num;  
public:  
    cNumber( )  
    {  
        num = 0;  
    }  
    cNumber( int x )  
    {  
        num = x;  
    }  
    virtual void Print_it( )  
    {  
        cout << setw( 5 ) << num << endl;  
    }  
};  
class cHex:public cNumber  
{  
public:  
    cHex( ):cNumber( )  
    {  
    }  
    cHex( int a ):cNumber( a )  
    {  
    };  
    void Print_it( )  
    {  
        cout.setf( ios::hex, ios::basefield );  
        cout << setw( 5 ) << num << endl;  
    }  
};  
class cOct:public cNumber  
{  
public:  
    cOct( ):cNumber( )  
    {  
    }  
    cOct( int a ):cNumber( a )  
    {  
    };  
    void Print_it( )  
    {  
        cout.setf( ios::oct, ios::basefield );  
        cout << setw( 5 ) << num << endl;  
    }  
};  
class cBinary:public cNumber  
{  
public:  
    cBinary( ):cNumber( )  
    {
```

```

        }
        ;
cBinary( int a ):cNumber( a )
{
}
;
void Print_it( );
}
;
void cBinary::Print_it( )
{
    int i      = 0,
        tmp,
        j      = 0;
    char bin[ 30 ];
    while( num )
    {
        bin[ i++ ] = num % 2 + 48;
        num      = num / 2;
    }
    bin[ i ] = '\0';
    for( i = 0; bin[ i ]; i++ )
    ;
    for( i--, j = 0; i >= j; i--, j++ )
    {
        tmp      = bin[ i ];
        bin[ i ] = bin[ j ];
        bin[ j ] = tmp;
    }
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 20 ) << bin << endl;
}
void main( )
{
    cNumber * ptr;
    int n;
    cout << "Input Number :";
    cin >> n;
    cNumber N ( n );
    cHex H ( n );
    cOct O ( n );
    cBinary B ( n );
    clrscr( );
    ptr = &H;
    ptr->Print_it( );
    ptr = &O;
    ptr->Print_it( );
    ptr = &B;
    ptr->Print_it( );
    getch( );
}

// INPUT :
Input
Number:
6

// OUTPUT :
6
6
110

```

Q.40 Write a template min that returns the minimum of two values. Make sure you handle the strings correctly.

// Example Code :

```
#include      <iostream.h>
#include      <conio.h>
#include      <iomanip.h>
template<class xType>  xType Min( xType & a, xType & b )
{
    if( a > b )
        return b;
    return a;
}
char * Min( char * S1, char * S2 )
{
    for( int  l1  = 0; S1[ l1 ]; l1++ )                                //  < >
    ;
    for( int  l2  = 0; S2[ l2 ]; l2++ )                                //  < >
    ;
    if( l1 < l2 )
        return S1;
    return S2;
}
void main( )
{
    int    x  = 40,
          y  = 60;
    float a  = 4.87,
          b  = .897;
    char   p  = 'a',
          q  = 'A';
    char  str1[ 10 ]  = "Mickey";
    char  str2[ 10 ]  = "Don";
    clrscr( );
    cout << "Minimum of" << x << " " << y << " is :" << Min( x, y ) <<
endl;
    cout << "Minimum of" << a << " " << b << " is :" << Min( a, b ) <<
endl;
    cout << "Minimum of" << p << " " << q << " is :" << Min( p, q ) <<
endl;
    cout << "Minimum of" << str1 << " " << str2 << "is :" << Min( str1,
str2 )
    << endl;
    getch( );
}

// OUTPUT :
Minimum
of 40 60 is :40
Minimum of 4.87 0.897 is :0.897
Minimum of a A is :A
Minimum of Mickey Don is :Don
```

Q.41 Write a program to reverse a given string using stack technique.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <stdio.h>
void main( )
{
    clrscr( );
    void reverse( );
    cout << "Enter a String : ";
    reverse( );
    getch( );
}
void reverse( )
{
    char x;
    x = getchar( );
    if( x == '\n' )
    {
        cout << endl << "Reverse is:";
        return;
    }
    reverse( );
    cout << x;
}

// INPUT :
Enter
a String : system

// OUTPUT :
Reverse
is:
metsys
```

Q.42 Write a class called Clock that simulates the keeping of time. Use three private class members:

hours, minutes, and seconds. Your class should be able to :  
Set() that starting time.  
Increment() the time by one second.  
Display() the time.

The function should take an argument with a default value of zero to imply military time. If this value is something other than zero, display the time in standard Am and PM notation. For example, 4 minutes and 31 seconds past 7 PM should be displayed either 19:04:31 or 7:04:31 PM and 5 minutes past midnight should be displayed as either 00:05:

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class cClock

{
    int hh,
        mm,
        ss;
public:
    cClock( )
    {
        hh = mm = ss = 0;
    }
    void Set( int a = 0, int b = 0, int c = 0 );
    void Increment( );
    void Display( );
    int IsValid( );
}
int cClock::IsValid( )
{
    int flag = 0;
    if( ( hh <= 23 && hh >= 0 )
        && ( mm <= 59 && mm >= 0 )
        && ( ss <= 59 && ss >= 0 ) )
        flag = 1;
    return flag;
}
void cClock::Set( int a, int b, int c )
{
    hh = a;
    mm = b;
    ss = c;
}
void cClock::Increment( )
{
    ss++;
    if( ss >= 60 )
    {
        ss = 0;
        mm = 0;
        hh++;
    }
    if( hh > 23 )
        hh = 0;
}
void cClock::Display( )
```

```

{
    cout.fill( '0' );
    if( hh > 12 )
        cout << setw( 2 ) << hh - 12 << ":" << setw( 2 ) << mm << ":" <<
        setw( 2 ) << ss << "PM" << endl;
    else
        cout << setw( 2 ) << hh << ":" << setw( 2 ) << mm << ":" << setw(
2 )
        << ss << "AM" << endl;
}
void main( )
{
    cClock      Clk;
    clrscr( );
    Clk.Set( 23, 59, 59 );
    if( Clk.IsValid( ) )
    {
        Clk.Display( );
        Clk.Increment( );
        Clk.Display( );
        Clk.Increment( );
    }
    else
        cout << "invalid time";
    getch( );
}

/*
#include<iostream.h>
#include<iomanip.h>
#include<conio.h>
#include<string.h>
class cClock
{
int hh,mm, ss;
public:
cClass ()
{hh=mm=ss=0;}
void Set(int a=0,int b=0,int c=0);
void Increment();
void Display();
int IsValid();
};
int cClock :: IsValid()
{
int.flag=0;
if ((hh<=23 && hh>=0)&&(mm<=59 && mm>=0)&&(ss<=59 && ss>=0 ))
flag=1;
return flag;
}
void cClock::Set(int a,int b,int c)
{
hh=a;
mm=b;
ss=c;
}
void cClock::Increment()
{

```

```

ss++;
if (ss>=60)
{
ss=0;
mm++;
}
if (mm>=60)
mm= 0;
hh++;
}
if (hh>23)
hh=0;
}
void cClock :: Display()
{
cout.fill('0');
if (hh>12)
cout<<setw(2)<<hh-12<<" :" <<setw(2)<<mm<<" :" <<setw(2)<<ss<<" PM "<<endl;
else
cout<<setw(2)<<hh-12<<" :" <<setw(2)<<mm<<" :" <<setw(2)<<ss<<" AM "<<endl;
}
void main ()
{
cClock Clk;
clrscr();
Clk.Set(23,59,59);
if(Clk.IsValid())
{
Clk.Display();
Clk.Increment();
Clk.Display();
Clk.Increment();
Clk.Display();
Clk.Increment();
Clk.Display();
Clk.Increment();
Clk.Display();
Clk.Increment();
}
else
cout<<"inavlid time";
getch();
} */
}

// OUTPUT :
INITIAL
DATE - 11 : 59:59PM
Incremented date -
00:00:00AM
00:00:01AM
00:00:02AM
00:00:03AM

```

Q.43 Write a class called Date that keeps track of current date. Your class should be able to :

Set() the starting date.  
Increment() the day by 1. If no. of days overflows then corresponding changes in month and year should be considered.  
Display() the date in MM/DD/YY format.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#include      <time.h>
#include      <dos.h>
#include      <process.h>
class cDate
{
    unsigned   int yy,
                dd,
                mm;
public:
    cDate( )
    {
        dd = mm = yy = 0;
    }
    cDate( int a, int b, int c )
    {
        dd = a;
        mm = b;
        yy = c;
    }
    void      Set( );
    void      Increment( );
    friend    ostream & operator <<( ostream &,
                                         cDate & );
}
ostream & operator <<( ostream & Out, cDate & D )
{
    cout << setiosflags( ios::left );
    Out << D.dd << "/" << D.mm << ":" << D.yy << endl;
    return Out;
}
void cDate::Set( )
{
    char buff[ 9 ];
    _strdate( buff );
    cout << "\nBuffer : " << buff << endl;
    dd          = ( buff[ 3 ] - 48 ) * 10 + buff[ 4 ] - 48;
    mm          = ( buff[ 0 ] - 48 ) * 10 + buff[ 1 ] - 48;
    yy          = ( buff[ 6 ] - 48 ) * 10 + buff[ 7 ] - 48;
    buff[ 8 ] = '\0';
}
void cDate::Increment( )
{
    int NoDays = 0;
    dd++;
    switch( mm )
    {
        case 1:
        case 3:
        case 5:
        case 7:
```

```

        case 8:
        case 10:
        case 12:
            NoDays = 31;
            break;
        case 2:
            if( ( yy % 4 == 0 && yy % 100 != 0 )
                || yy % 400 == 0 )
                NoDays = 29;
            else
                NoDays = 28;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            NoDays = 30;
    }
    if( ( dd ) > NoDays )
    {
        dd -= NoDays;
        mm++;
        if( mm > 12 )
        {
            mm = mm % 12;
            yy += mm / 12;
            yy++;
        }
    }
    if( yy == 0 )
        yy += 2000;
    struct dosdate_t reset;
    reset.year = yy;
    reset.day = dd;
    reset.month = mm;
    _dos_setdate( &reset );
    system( "date" );
}
void main( )
{
    cDate D;
    clrscr( );
    D.Set( );
    cout << "The Date is : " << D;
    D.Increment( );
    D.Increment( );
    cout << D;
    D.Increment( );
    D.Increment( );
    cout << D;
    D.Set( );
    cout << D;
    getch( );
}

// INPUT :

Buffer :
04/17/08
The Date is : 17/4;
8
Current date is Thu 04/17/2008
Enter new date (mm-dd-yy): 7-24-86

```

```
// OUTPUT :  
Current  
date is Thu 07/24/1986  
Enter new date (mm-dd-yy): 07-24-07  
19 / 4;  
8  
Current date is Wed 07/24/1907
```

Q.44 Write a program that reads in string input from the user, reverse the case of the letters, and then echos the string back to the user. The class ReverseCase contains a character buffer up to 80 bytes in length as the member variable and Read(), Convert() and Print() as member functions. Define Read and print functions as inline functions.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <stdio.h>
#include      <conio.h>
class cRevCase
{
    char *      str;
    int   len;
public:
    cRevCase( )
    {
        str = '\0';
        len = 0;
    }
    cRevCase( char * S )
    {

        for( int i = 0; S[ i ]; i++ )                                // < >

            ;
        len = i;
        str = new char [len + 1];
        int j = 0;
        for( i = 0; S[ i ]; i++ )
            str[ j++ ] = S[ i ];
        str[ j ] = '\0';
    }
    inline void Read( )
    {
        cout << "Length of String : ";
        cin >> len;
        str = new char [len + 1];
        cout << "Enter String : ";
        cin >> str;
    }
    inline void Print( )
    {
        cout.setf( ios::left, ios::adjustfield );
        cout << setw( len ) << str << endl;
    }
    cRevCase Convert( );
} ;
cRevCase cRevCase::Convert( )
{
    cRevCase tmp;
    tmp.len = len;
    int j = 0;
    tmp.str = new char [tmp.len + 1];
    for( int i = 0; str[ i ]; i++ )                                // < >

    {
        if( str[ i ] >= 'a' && str[ i ] <= 'z' )
            tmp.str[ j++ ] = str[ i ] - 32;
        else if( str[ i ] >= 'A'
                  && str[ i ] <= 'Z' )

```

```

        tmp.str[ j++ ] = str[ i ] + 32;
    else
        tmp.str[ j++ ] = str[ i ];
    }
}
tmp.str[ j ] = '\0';
return tmp;
}
void main( )
{
    cRevCase S1 ( "MiCkEy123" ),
               S2 ( "Ramesh" ),
               S3,
               S4;
    clrscr( );
    S3.Read( );
    cout << "string 1 :";
    S1.Print( );
    S4 = S3.Convert( );
    S2 = S1.Convert( );
    cout << "string 2 :";
    S2.Print( );
    cout << "string 3 :";
    S3.Print( );
    cout << "string 4 :";
    S4.Print( );
    getch( );
}

// INPUT :
Length
of String : 5
Enter String      :SAGAR

// OUTPUT :

string
1 :MiCkEy123
string 2 :mIcKeYL23
string 3 :SAGAR
string 4 :sagar

```

Q.45 Create base class media. This class has two fields one for title and the other for price. Derive two specific classes called tape and book, tape has a field called length and book has got one field called pages. Add two member functions to base class, one get\_data() to initialize base class data members and second display\_data() which is virtual to display values. Override display\_data() function in both derived classes to display data specific to them.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
#include      <stdio.h>
class cMedia
{
protected:
    char title[ 10 ];
    float price;
public:
    cMedia( )
    {
        strcpy( title, '\0' );
        price = 0.0;
    }
    cMedia( char * t, float p )
    {
        strcpy( title, t );
        price = p;
    }
    void GetData( );
    virtual void DispData( );
} ;
class cBook:public cMedia
{
    int pages;
public:
    cBook( ):cMedia( )
    {
        pages = 0;
    }
    cBook( char * t, float pr, int p )
                           :cMedia( t, pr )
    {
        pages = p;
    }
    void GetData( );
    void DispData( );
} ;
class cTape:public cMedia
{
    int len;
public:
    cTape( ):cMedia( )
    {
        len = 0;
    }
    cTape( char * t, float pr, int p )
                           :cMedia( t, pr )
    {
        len = 1;
    }
    void GetData( );
}
```

```

        void DispData( );
    }

void cMedia::GetData( )
{
    cout << "Title:" ;
    gets( title );
    fflush( stdin );
    cout << "Price:" ;
    cin >> price;
}

void cMedia::DispData( )
{
    cout.setf( ios::left, ios::adjustfield );
    cout.precision( 3 );
    cout << setw( 20 ) << title << "Rs. " ;
    cout.setf( ios::right, ios::adjustfield );
    cout.setf( ios::showpoint );
    cout << setw( 10 ) << price;
}

void cBook::GetData( )
{
    cMedia::GetData( );
    cout << "Pages      : " ;

    cin >> pages;
}

void cBook::DispData( )
{
    cMedia::DispData( );
    cout << setw( 4 ) << pages << endl;
}

void cTape::GetData( )
{
    cMedia::GetData( );
    cout << "Length     . " ;
    cin >> len;
}

void cTape::DispData( )
{
    cMedia::DispData( );
    cout << setw( 4 ) << len << endl;
}

void main( )
{
    cMedia      *  ptr;
    cBook      B;
    cTape      T;
    ptr = &B;
    clrscr( );
    B.GetData( );
    ptr->DispData( );
    ptr = &T;
    T.GetData( );
    ptr->DispData( );
    getch( );
}

```

// INPUT & OUTPUT :

Title:  
khnh  
Price:

230  
Pages:  
234  
khnk                      Rs.        230.000 234

Title:  
Networking  
Price:  
442  
Length       . 120  
Networking                      Rs.        442.000 120

Q.46 An election is contested by five candidates. The candidates are numbered from 1 to 5 and voting is done by accepting the candidate number from voter. Write a program which will accept the votes and count votes for each candidate. If the number typed by voter is outside the range the vote is discarded and the program will keep track of the number of such discarded votes.

```
// Example Code :

#include    <iostream.h>
#include    <iomanip.h>
#include    <conio.h>
#include    <stdio.h>
class cElection
{
    int votes[ 5 ];
    int dis;
public:
    cElection( );
    void Cast( );
    void Disp( );
}
cElection::cElection( )
{
    for( int i = 0; i < 5; i++ )                                // < >

        votes[ i ] = 0;
    dis = 0;
}
void cElection::Cast( )
{
    int x;
    cout << "Candidate Number : (1-5) :";
    cin >> x;
    if( x <= 5 && x > 0 )
        votes[ x - 1 ]++;
    else
        dis++;
}
void cElection::Disp( )
{
    for( int i = 0; i < 5; i++ )                                // < >

        cout << i + 1 << setw( 3 ) << votes[ i ] << endl;
    cout << "Discarded votes =" << dis << endl;
}
void main( )
{
    char ch = 'Y';
    cElection E;
    clrscr();
    while( ch == 'Y' || ch == 'y' )
    {
        E.Cast();
        cout << "Want to cast more      :";
        ch = getche();
        cout << endl;
    }
    //clrscr();
    E.Disp();
    getch();
}
```

```
// INPUT :  
  
Candidate  
Number:  
(1-5) :2  
Want to cast more      :Y  
Candidate Number : (1-5) :2  
Want to cast more      :Y  
Candidate Number : (1-5) :  
4  
Want to cast more      :N
```

```
// OUTPUT :  
1  0  
2  2  
3  0  
4  1  
5  0  
Discarded votes =0
```

Q.47 Implement circular link list. Include the following functions ;  
 a. Adding nodes to the list.  
 b. Traversing the whole list  
 c. Deleting a node form the list.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <string.h>
class       cCList;
class       cNode
{
    char      name[ 15 ];
    int       age;
    float     basic;
    cNode   * next;
public:
    cNode( )
    {
        strcpy( name, '\0' );
        age   = 0;
        basic = 0.0;
        next  = '\0';
    }
    cNode( char *, int, float );
    void DispNode();
    friend      class   cCList;
}
class       cCList
{
    cNode   * start,
            * last;
public:
    cCList( )
    {
        start = last = '\0';
    }
    void Add();
    void Add( char *, int, float );
    void Search( char * );
    void Delete( char * );
    void DispList();
}
cNode::cNode( char * n, int a, float b )
{
    strcpy( name, n );
    age   = a;
    basic = b;
    next  = '\0';
}
void cNode::DispNode()
{
    cout.setf( ios::left, ios::adjustfield );
    cout << setw( 15 ) << name;
    cout.setf( ios::right, ios::adjustfield );
    cout << setw( 3 ) << age << setw( 6 ) << basic << endl;
}
void cCList::Add()
{
    char  nm[ 15 ];
    int   a;
```

```

float b;
while( 1 )
{
    cout << "Name (Quit):";
    cin >> nm;
    if( strcmp( nm, "Quit" ) == 0 )
        break;
    else
    {
        cout << "Age:";
        cin >> a;
        cout << "Basic:";
        cin >> b;
        cNode * NewNode = new cNode( nm, a, b );
        if( !start )
        {
            start = last = NewNode;
            last->next = NewNode;
        }
        else
        {
            NewNode->next = start;
            last->next = NewNode;
            last = NewNode;
        }
    }
}
void cCList::Search( char * nm )
{
    int count = 1,
        yes = 0,
        flag = 0;
    if( last->next )
    {
        for( cNode * ptr = last->next; // < >

                !( flag && ptr == last->next );
                ptr = ptr->next, flag = 1, count++ )

        {
            if( strcmp( ptr->name, nm ) == 0 )
            {
                cout << "Found at Rec # " << count << endl;
                ptr->DispNode();
                yes++;
            }
        }
        if( !yes )
            cout << "Name not Found ..... ." << endl;
    }
    else
        cout << "List does not exists ..." << endl;
}
void cCList::Delete( char * nm )
{
    int flag = 0;
    cNode * prev;
    for( cNode * ptr = start; // < >

            !( flag && ptr == start );
            prev = ptr, ptr = ptr->next, flag = 1 )

    {
        if( strcmp( ptr->name, nm ) == 0 )
            break;

```

```

        }
        if( strcmp( ptr->name, nm ) == 0 )
        {
            if( ptr == last->next )
            {
                if( ptr == last )
                {
                    ptr->next = '\0';
                    ptr      = '\0';
                }
                else
                    last->next = ptr->next;
            }
            else
            {
                if( ptr == last )
                {
                    prev->next = ptr->next;
                    last      = prev;
                }
                else
                    prev->next = ptr->next;
            }
            delete ptr;
        }
        else
            cout << endl << "Name Not Found .....Aborting" << endl;
    }
    void cCList::DispList( )
    {
        int flag  = 0;
        for( cNode * ptr  = last->next;                                //  < >
              !( flag && ptr == last->next );
              ptr = ptr->next, flag = 1 )
            ptr->DispNode( );
    }
    void main( )
    {
        cCList      L;
        char name[ 15 ];
        clrscr( );
        L.Add( );
        clrscr( );
        L.DispList( );
        cout << "Name to be Deleted :";
        cin >> name;
        cout << endl;
        L.Search( name );
        getch( );
    }

// OUTPUT :

ABC
12      1200
XYZ          24      2400
QUIT         32 13000
quit        22      3444
Name to be Deleted :QUIT

Found at Rec # 3
QUIT          32 13000

```

Q.48-49 Write a menu-driven program which will accept an array of 10 integer values and sort them with any two sorting algorithms of your choice.

// Example Code :

```
#include      <iostream.h>
#include      <iomanip.h>
#include      <conio.h>
#include      <stdio.h>
void main( )
{
    int * arr;
    int n;
    void Bubble( int *, int );
    void Quick( int *, int, int );
    void Insert( int *, int );
    void Selection( int *, int );
    clrscr( );
    cout << "Enter No. of Elements :";
    cin >> n;
    arr = new int [n];
    cout << "Enter the elements now " << endl;
    for( int i = 0; i < n; i++ )                                // < >

    {
        cout << "arr[" << i + 1 << "] :";
        cin >> arr[ i ];
    }
//clrscr();
    cout << "Select Ur Choice " << endl;
    cout << " 1      Bubble Sort " << endl;
    cout << " 2      Insertion Sort " << endl;
    cout << " 3      Selection Sort " << endl;
    cout << " 4      Quick Sort " << endl;
    cout << "Enter ur Choice      (1-4) :";
    int ch;
    cin >> ch;
    switch( ch )
    {
        case 1:
            Bubble( arr, n );
            break;
        case 2:
            Insert( arr, n );
            break;
        case 3:
            Selection( arr, n );
            break;
        case 4:
            Quick( arr, 0, n );
            break;
        default:
            cout << "I N V A L I D C H O I C E E N T E R E D";
            break;
    }
    cout << "Sorted List is " << endl;
    cout.width( 4 );
    for( i = 0; i < n; i++ )
        cout << "arr[" << i + 1 << "] :" << arr[ i ] << endl;
    getch( );
}
void Bubble( int * arr, int n )
```

```

{
    for( int i = 0; i < n - 1; i++ )                                // < >
    {
        for( int j = 0; j < n - 1 - i; j++ )                          // < >
        {
            if( arr[ j ] > arr[ j + 1 ] )
            {
                int tmp = arr[ j ];
                arr[ j ] = arr[ j + 1 ];
                arr[ j + 1 ] = tmp;
            }
        }
    }
}

void Insert( int * arr, int n )
{
    int tmp;
    for( int i = 1; i < n; i++ )                                    // < >
    {
        if( arr[ i - 1 ] > arr[ i ] )
            tmp = arr[ i ];
        for( int j = i - 1;                                            // < >
              j >= 0 && arr[ j ] > tmp; j-- )
            arr[ j + 1 ] = arr[ j ];
        arr[ j + 1 ] = tmp;
    }
}

void Selection( int * arr, int n )
{
    int GetMin( int *, int, int );
    int tmp;
    for( int i = 0; i < n; i++ )                                    // < >
    {
        int min = GetMin( arr, i, n );
        tmp = arr[ min ];
        arr[ min ] = arr[ i ];
        arr[ i ] = tmp;
    }
}

int GetMin( int * arr, int pos, int size )
{
    int tmp,
        min = pos++;
    while( pos < size )
    {
        if( arr[ pos ] < arr[ min ] )
            min = pos;
        pos++;
    }
    return min;
}

void Quick( int * arr, int lo, int hi )
{
    int Divide( int *, int, int );
    int pivotloc;
    if( lo < hi )

    {
        pivotloc = Divide( arr, lo, hi );
}

```

```

        Quick( arr, lo, pivotloc - 1 );
        Quick( arr, pivotloc + 1, hi );
    }
}
int Divide( int * arr, int      lo, int   hi )
{
    int pivotloc  = ( lo + hi ) / 2;
    int pivotval  = arr[ pivotloc ];
    arr[ pivotloc ] = arr[ 10 ];
    arr[ lo ]       = pivotloc;
    pivotloc        = lo;
    for( int i = lo + 1; i < hi; i++ ) //  < >

    {
        if( arr[ i ] <= pivotval )
        {
            int tmp  = arr[ ++pivotloc ];
            arr[ pivotloc ] = arr[ i ];
            arr[ i ]       = tmp;
        }
    }
    arr[ lo ]       = arr[ pivotloc ];
    arr[ pivotloc ] = pivotval;
    return pivotloc;
}

// INPUT :

Enter
No. of Elements :6
Enter the elements now
arr[1] :23
arr[2] :87
arr[3] :65
arr[4] :12
arr[5] :99
arr[6] :43
Select Ur Choice
1           Bubble Sort
2           Insertion Sort
3           Selection Sort
4           Quick Sort
Enter ur Choice (1-4) :1

// OUTPUT :

Sorted
List is
arr[1]      :12
arr[2]      :23
arr[3]      :43
arr[4]      :65
arr[5]      :87
arr[6]      :99

```