
Assignment Name: Implementation of Linear array (static) operations: Insert, Delete, Traverse.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>

class demo
{
	int a[10],i,j,n,item,k;
public:
	void get();
	void insert();
	void del();
	void dis();
};
void demo::get()
{
	cout<<"\nEnter n";
	cin>>n;
	cout<<"\nEnter Array Element:";
	for(i=1;i<=n;i++)
	cin>>a[i];
}
void demo::insert()
{
	cout<<"\nEnter Position:";
	cin>>k;
	cout<<"\nEnter Item:";
	cin>>item;
	j=n;
	while(j>=k)
	{
		a[j+1]=a[j];
		j--;
	}
	a[k]=item;
	n++;
}
void demo::del()
{
	cout<<"\nEnter Position:";
	cin>>k;
	j=k;
	while(j<=n-1)
	{
		a[j]=a[j+1];
		j++;
	}
	n--;
}
void demo::dis()
{
	cout<<"\n Elements are\n";
	for(i=1;i<=n;i++)
	cout<<a[i]<<"\t";
}
void main()

{
      clrscr();
      demo d;
      int ch;
      d.get();
      cout<<"\n1. Insert 2.Del 3.Dis 4. Exit\n";
      while(ch!=4)
      {
            cout<<"\n Enter choice";
            cin>>ch;
            switch(ch)
            {
                  case 1: d.insert(); break;
                  case 2: d.del(); break;
                  case 3: d.dis(); break;
                  case 4: exit(0);
            }
      }
      getch();
}
*/ Output */
```

```
Enter n 3

Enter Array Element:1 2 4

1. Insert 2.Del 3.Dis 4. Exit

 Enter choice 3

 Elements are
1       2       4
 Enter choice 1

Enter Position: 2

Enter Item: 6

 Enter choice 3

 Elements are
1       6       2       4
 Enter choice 2

Enter Position: 3

 Enter choice 3

 Elements are
1       6       4
 Enter choice 4
```

Matrix Addition using Two Dimensional Arrays in C++

```
#include <iostream.h>
#include<conio.h>

int main()
{
    int r, c, a[100][100], b[100][100], sum[100][100], i, j;

    cout << "Enter number of rows (between 1 and 100): ";
    cin >> r;

    cout << "Enter number of columns (between 1 and 100): ";
    cin >> c;

    cout << endl << "Enter elements of 1st matrix: " << endl;

    // Storing elements of first matrix entered by user.
    for(i = 0; i < r; ++i)
       for(j = 0; j < c; ++j)
       {
           cout << "Enter element a" << i + 1 << j + 1 << " : ";
           cin >> a[i][j];
       }

    // Storing elements of second matrix entered by user.
    cout << endl << "Enter elements of 2nd matrix: " << endl;
    for(i = 0; i < r; ++i)
       for(j = 0; j < c; ++j)
       {
           cout << "Enter element b" << i + 1 << j + 1 << " : ";
           cin >> b[i][j];
       }

    // Adding Two matrices
    for(i = 0; i < r; ++i)
        for(j = 0; j < c; ++j)
            sum[i][j] = a[i][j] + b[i][j];

    // Displaying the resultant sum matrix.
    cout << endl << "Sum of two matrix is: " << endl;
    for(i = 0; i < r; ++i)
        for(j = 0; j < c; ++j)
        {
            cout << sum[i][j] << "  ";
            if(j == c - 1)
                cout << endl;
        }

getch();
}
```

```
-----------------------------------------------------------------------------
Assignment Name: Implementation of Linear and Binary Search
Class: MCA I    Lab: CA Lab III                        (DS)
-----------------------------------------------------------------------------

#include<iostream.h>
#include<conio.h>
#include<process.h>

class demo
{

int a[10],i,j,n,f,temp,ele,demo,mid,low,high; public:
void get();
void sort();
void linear();
void binary();

void dis();
};

void demo::get()
{
cout<<"\n Enter n:";
cin>>n;
cout<<"\nEnter array Elements:";
for(i=1;i<=n;i++)
cin>>a[i];
}

void demo::linear()

{
int ele;
cout<<"\nEnter the element to be search";
cin>>ele;
for(i=1;i<=n;i++)
{
if(a[i]==ele)
{

cout<<"\nSuccessful search"; cout<<"\nElement is found at position "<<i;
return;
}

}
if(i>n)
{
cout<<"\nUnsuccessful search:";
cout<<"\nElement is not found ";
}
}

void demo::sort()
{
```

```
for(i=1;i<=n;i++)
{
for(j=1;j<=n-1;j++)

{
if(a[j]<a[j+1])
{
temp=a[j];
a[j]=a[j+1];
a[j+1]=temp;
}
}
}
}

void demo::binary()
{
cout<<"\nEnter element to be search ";
cin>>ele;
f=0;
low=1;
high=n;
while(low<=high)
{
mid=(low+high)/2;
if(a[mid]==ele)

{
f=1;
cout<<"\nElement is found at :"<<mid;
return;
}
else if(a[mid]<ele)
low=mid+1;
else if(a[mid]>ele)
high=mid-1;
}
if(f==0)
cout<<"\n Element is not found:";
}

void demo::dis()
{
cout<<"\n Element are \n";
for(i=1;i<=n;i++)
cout<<a[i]<<"\t";
}

void main()
{
clrscr();
demo d;

int ch;
d.get();
```

```
d.dis();
cout<<"\n 1:Linear 2:Binary 3:exit\n";
while(ch!=3)
{
cout<<"\nEnter Choice:";
cin>>ch;
switch(ch)
{
case 1: d.linear(); break;
case 2: d.sort();
d.dis();

d.binary(); break;
case 3: exit(0); break;
}
}
getch();
}

*/ Output */

Enter n:3

Enter array Elements:12 3 45

Element are
12   3    45
1:Linear 2:Binary 3:exit

Enter Choice:1

Enter the element to be search 3

Successful search

Element is found at position 2
Enter Choice:2

Element are
45   12   3
Enter element to be search 12

Element is found at :2
Enter Choice:2

Element are
45   12   3
Enter element to be search 56

Element is not found:
Enter Choice:3
```

```
-------------------------------------------------------------------------------
Assignment Name: Implementation of Stack for Integer
Class: MCA I                                           Lab: CA Lab III (DS)
-------------------------------------------------------------------------------
#include<iostream.h>
#include<conio.h>
#include<process.h>
class stack
{
	int s[10],n,top,ele,i;
public:
	stack()
	{
		top=-1;

	}
	void push();
	void dis();
	int pop();
	int peep();
	void change();
};
void stack::push()
{
	if(top>=2)
	 cout<<"\nStack is overflow:";
	else
	{

	 cout<<"\nEnter element:";
	 cin>>ele;
	 top++;
	 s[top]=ele;
```

```
      }
}

void stack::dis()
{
      cout<<"\nElements in stack are:\n";
      for(i=top;i>=0;i--)
       cout<<s[i]<<"\t";
}
int stack::pop()
{
      if(top==-1)
      {
            cout<<"\nUnderflow";
            return 0;
      }
      else
            return (s[top--]);
}

int stack::peep()

{
      cout<<"\nEnter position:";
      cin>>i;
      if((top-i+1)<0)
      {
            cout<<"\nUnderflow";
            return 0;
      }
      else
      return (s[top-i+1]);

}
```

```
void stack::change()
{
	cout<<"\nEnter position ";
	cin>>i;
	if((top-i+1)<0)
	{
		cout<<"\nUnderflow";
	}
	else
	{
		int n;
		cout<<"\nEnter element:";
		cin>>n;
		s[top-i+1]=n;
	}
}

void main()
{
	clrscr();
	stack s;
	int ch;
	cout<<"\n1. Push  2.Display  3.Pop  4.Peep  5.Change 6.Exit\n";
	while(ch!=6)
	{
		cout<<"\nEnter ch :";
		cin>>ch;
		switch(ch)
		{
			case 1: s.push(); break;
			case 2: s.dis(); break;
```

```
                case 3: int n=s.pop();
                     if(n>0)
                     cout<<"\nPop ele is "<<n;
                     break;
                case 4: int m=s.peep();
                     if(m>0)
                     cout<<"\nPeep ele is "<<m;
                     break;
                case 5: s.change(); break;
                case 6: exit(0);
           }
     }
     getch();
}
*/ Output */
1. Push 2.Display 3.Pop 4.Peep 5.Change 6.Exit
Enter ch :1
Enter element:10
Enter ch :1
Enter element:20
Enter ch :1
Enter element:30
Enter ch :1
Stack is overflow:
Enter ch :2
Elements in stack are:

30      20      10
Enter ch :3
Pop ele is 30
Enter ch :2
Elements in stack are:
```

```
20      10
Enter ch
:4
Enter position:1

Peep ele is 20

Enter ch :2

Elements in stack are:


20      10
Enter ch
:5
Enter position 1

Enter element:80

Enter ch :2

Elements in stack are:

80      10
Enter ch : 6
```

Assignment Name: Implementation of Infix to Postfix Expression

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

#include<string.h>

class convert

{

char infix[20],postfix[20],s[20];

int i,p,top;

public:

convert()

{

top=-1;

i=p=0;


cout<<"\nEnter infix Expression:";

cin>>infix;

strcat(infix,")");

s[++top]='(';

}

int precedance(char);

void post();

void display();

};

int convert::precedance(char ch)

{

switch(ch)


{

case '^':return 3;

case '*':return 2;
case '/':return 2;

case '+':return 1;

case '-':return 1;

default: return 0;

}

}

void convert::post()

{

char ch;

while(top!=-1)

{

ch=infix[i++];

if((ch>='A'&&ch<='Z')||(ch>='a'&&ch<='z')||(ch>='1'&&ch<='9'))

postfix[p++]=ch;

else if(ch=='(')

s[++top]=ch;

else if(ch=='+'||ch=='-'||ch=='*'||ch=='/'||ch=='^')

{

while(precedance(ch)<=precedance(s[top]))

postfix[p++]=s[top--];

s[++top]=ch;


}

else if(ch==')')

{

while(s[top]!='(')

postfix[p++]=s[top--];

top--;

}

else

cout<<"\nWrong string";

}

postfix[p]='\0';
```

```
}

void convert::display()

{

cout<<"\nPostfix Expression is :"<<postfix;

}

void main()

{

clrscr();

convert c;

c.post();

c.display();

getch();

}


*/ Output */

Enter infix Expression:(a*b-(c+d/e^f)*h)

Postfix Expression is :ab*cdef^/+h*-

Enter infix Expression:a+2*5

Postfix Expression is :a25*+
------------------------------------------------------------------------------
Assignment Name: Evaluation of Postfix Expression using Stack
Class: MCA I                                              CA LAB III (DS)
-------------------------------------------------------------------------------
#include<iostream.h>

#include<stdlib.h>
#include<conio.h>
#include<math.h>
#include<ctype.h>
#include<stdio.h>
const int MAX = 50 ;
class postfix
{
     private :
          int stack[MAX] ;
          int top, nn ;
          char s[20] ;

     public :
          postfix( ) ;
          void setexpr () ;
          void push ( int item ) ;
          int pop( ) ;
          void calculate( ) ;
          void show( ) ;
} ;
postfix :: postfix( )
{
     top = -1 ;
}
void postfix :: setexpr ( )

{
     gets(s);
}
void postfix :: push ( int item )
{
     if ( top == MAX - 1 )
     cout << endl << "Stack is full" ;
     else
     {
          top++ ;
          stack[top] = item ;
     }

}
int postfix :: pop( )
{
     if ( top == -1 )
     {
          cout << endl << "Stack is empty" ;
          return NULL ;
     }
          int data = stack[top] ;
          top-- ;
          return data ;
}
void postfix :: calculate( )

{
     int n1, n2, n3 ;
```

```
      int i=0;
      char ch;
      while (s[i]!='\0')

           int n=0;
           ch = s[i];
           if ( ch == ' ' || ch == '\t' )
           {
                 i++ ;
                 continue ;
           }
           if ( isdigit ( ch ) )
           {
                 while(isdigit(s[i]))
                 {
                       nn = s[i] - '0'
                       ; n= n*10+nn;
                       i++;

                 }
                 push (n) ;

           }
           else
           {
                 n1 = pop( ) ;
                 n2 = pop( ) ;
                 switch ( ch )
                 {
                       case '+' : n3 = n2 + n1 ;
                       break ;
                       case '-' : n3 = n2 - n1 ;
                       break ;
                       case '/' : n3 = n2 / n1 ;
                       break ;

                       case '*' : n3 = n2 * n1 ;
                       break ;
                       case '%' : n3 = n2 % n1 ;
                       break ;
                       case '^' : n3 = pow ( n2 , n1 ) ;
                       break ;
                       default : cout << "Unknown operator" ;
                       exit ( 1 ) ;
                 }
                       push ( n3 ) ;
           }
           i++ ;

      }
}
void postfix :: show( )
{
      nn = pop ( ) ;
      cout << "\nResult is: " << nn ;
}
void main( )
{
      clrscr();
      cout << "\nEnter postfix expression to be evaluated : \n" ;
      postfix q ;
      q.setexpr() ;

      q.calculate() ;
```

```
     q.show() ;
     getch();
}
/* OUTPUT

Enter postfix expression to be evaluated:
122 140 * 150 340 + -

Result is: 16590
```

```
------------------------------------------------------------------------
Assignment Name: Implementation of linear queue for integer
Class: MCA I     Lab: CA Lab III (DS)
-----------------------------------------------------------------------
#include<iostream.h>

#include<conio.h>
#include<process.h>
class queue
{
int f,r,q[10],n,i;
public:
queue()
{
f=r=0;
}
void insert();
void del();

void dis();
};

void queue::insert()
{
if(r==3)
cout<<"\nOverflow";
else
{
cout<<"\nEnter n";
cin>>n;
if(f==0)
f=1;

r++;
q[r]=n;
}
}

void queue::del()
{
if(f==0)
{
cout<<"\nUnderflow";
return;
}

else
{
int n;
n=q[f];
if(f==r)
f=r=0;
else
f++;
cout<<"\nDeleted element is "<<n;
}
}

void queue::dis()

{
if(f==0)
cout<<"\nUnderflow";
else
{
cout<<"\nElements in queue are:";
for(i=f;i<=r;i++)
cout<<q[i]<<"\t";
}
}

void main()
{
clrscr();
queue q;
int ch;
cout<<"\n 1.insert 2.display 3.delete 4. exit \n";
while(ch!=4)
{
cout<<"\nEnter ch:";
cin>>ch;
switch(ch)

{
case 1: q.insert(); break;
case 2: q.dis(); break;
case 3: q.del(); break;
case 4:exit(0);
}
}
getch();
}

*/ Output */


1.insert 2.display 3.delete 4. exit

Enter ch:3

Underflow
Enter ch:1

Enter n10

Enter ch:1

Enter n20

Enter ch:1

Enter n30

Enter ch:1

Overflow
Enter ch:2
Elements in queue
are:10                      20
Enter ch:3

Deleted element is 10
Enter ch:2
Elements in queue
are:20                      30
Enter ch:4
```

```
---------------------------------------------------------------------------
- Assignment Name: Implementation of Circular Queue for integer
Class: MCA I                                              Lab: CA Lab III (DS)
----------------------------------------------------------------------------
```

```
#include<iostream.h>
#include<conio.h>
class queue
{
	int a[5],r,f;
public:

	queue()
	{
		f=r=-1;
	}
	void push();
	void pop();
	void show();
};

void queue::push()
{
	int item;

	if(f==0 &&r==4 || f==r+1)
	{
		cout<<"\n Overflow";
	}
	else
	{
		if(r==4)
```

```
          r=-1;

          r++;

          cout<<"\nEnter item :";

          cin>>item;

          a[r]=item;


          if(f==-1)

          {

               f=0;

          }

     }

}


void queue::pop()

{

     if(f==-1)

     {

          cout<<"\n Underflow";

     }

     else

     {

          cout<<"\nDeleted element is :"<<a[f];

          if(f==r)

          {

               f=-1;

               r=-1;

          }
```

```
            else
            {
                  if(f==4)
                   f=0;
                  else
                   f++;
            }
      }
}

void queue::show()
{
      if(f==-1)
      {
            cout<<"\nEmpty :";
      }
      else if(f<=r)
      {
            for(int i=f;i<r;i++)
            {
                  cout<<"\n"<<a[i];
            }
      }
      else
      {
            for(int i=f;i<=4;i++)
            {
                  cout<<"\n"<<a[i];
            }
            for(int j=0;j<=r;j++)
            {
                  cout<<"\n"<<a[i];
```

```
           }

      }

}

void main()

{

      queue s;

      int ch;

      clrscr();


      do

      {cout<<"\n 1: Push 2: Pop 3:show 4:exit ";
           cout<<"\nEnter choice";

           cin>>ch;


           switch(ch)

           {

                case 1: s.push(); break;

                case 2: s.pop(); break;

                case 3: s.show(); break;

                default: cout<<"\n Wrong Choice";

           }

      }while(ch<=3);

}
```

```
*/ Output */


1: Push 2: Pop 3:show 4:exit
```

```
Enter choice1


 Overflow


 1: Push 2: Pop 3:show
4:exit Enter choice3


10

20

30

40

50


 1: Push 2: Pop 3:show
4:exit Enter choice2


Deleted element is :10


 1: Push 2: Pop 3:show
4:exit Enter choice2


Deleted element is :20


 1: Push 2: Pop 3:show
4:exit Enter choice3


30

40

50


 1: Push 2: Pop 3:show
4:exit Enter choice1


Enter item :44



 1: Push 2: Pop 3:show
4:exit Enter choice1


Enter item :55



 1: Push 2: Pop 3:show
4:exit Enter choice1


 Overflow
 1: Push 2: Pop 3:show
4:exit Enter choice3


30

40

50

44

55


 1: Push 2: Pop 3:show
4:exit Enter choice 4
```

- **Assignment Name: Perform Bubble Sort for Integer**

Class: MCA I **Lab: CA Lab III (DS)**

```
#include<iostream.h>

#include<conio.h>

class demo

{

      int a[10],i,last,exch,j,n,temp;

public:

      void get();

      void asc_sort();

      void dec_sort();

      void disp();

};


void demo::get()

{

      cout<<"\n Enter the array size:";

      cin>>n;

      cout<<"\nEnter the array element:";

      for(i=1;i<=n;i++)

      cin>>a[i];

}


void demo::asc_sort()

{

      last=n;

      for(i=1;i<=n-1;i++)

      {

            exch=0;

            for(j=1;j<=last-1;j++)

            {
                if(a[j]>a[j+1]) //change a[j]<a[j+1] for descending
                sort {

                     temp=a[j];

                     a[j]=a[j+1];

                     a[j+1]=temp;

                     exch=exch+1;

                }


           }

           if(exch==0)

                return;

           else

                last=last-1;


      }

}


void demo::dec_sort()

{

      last=n;

      for(i=1;i<=n-1;i++)

      {

           exch=0;

           for(j=1;j<=last-1;j++)

           {

                if(a[j]<a[j+1])
                {
                        temp=a[j];
                        a[j]=a[j+1];
                        a[j+1]=temp;
                        exch=exch+1;
                }
            }
            if(exch==0)
                return;
            else
                last=last-1;
        }
}

void demo::disp()
{
        cout<<"\nThe array element are :";
        for(i=1;i<=n;i++)
            cout<<a[i]<<"\t";
}

void main()
{

        clrscr();
        demo d;
        d.get();
        d.disp();
        d.asc_sort();
        cout<<"\nAfter Ascending Sort:";
        d.disp();
        d.dec_sort();
```

```
      cout<<"\nAfter Descending Sort:";

      d.disp();

      getch();

}


*/ Output */



 Enter the array size: 3


Enter the array element: 12 3 45


The array element are:  12 3       45

After Ascending Sort:

The array element are:  3  12      45

After Descending Sort:

The array element are:  45 12      3
```

- Assignment Name: Implementation of Selection Sort

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>


class demo

{


      int a[10],i,
min_index,j,n,temp,max_index; public:

      void get();

      void asc_sort();

      void dsc_sort();

      void disp();

};


void demo::get()

{

      cout<<"\nEnter the array size:";

      cin>>n;

      cout<<"\nEnter the array element:";

      for(i=1;i<=n;i++)

      cin>>a[i];

}


void demo::asc_sort()

{

      for(i=1;i<=n-1;i++)

      {

            min_index=i;

            for(j=i+1;j<=n;j++)

            {
                if(a[j]<a[min_index]) // change a[j]>a[min_index] for
                descend min_index=j;

            }


            if(min_index!=i)

            {

                temp=a[min_index];

                a[min_index]=a[i];

                a[i]=temp;

            }

        }

}



void demo::dsc_sort()

{

        for(i=1;i<=n;i++)

        {

            max_index=i;

            for(j=i+1;j<=n;j++)

            {

                if(a[j]>a[max_index])

                min_index=j;

            }


            if(max_index!=i)
            {

                  temp=a[max_index];

                  a[max_index]=a[i];

                  a[i]=temp;

            }

      }

}


void demo::disp()

{

      cout<<"\n The array element are ";

      for(i=1;i<=n;i++)

      cout<<a[i]<<"\t";

}


void main()

{

      clrscr();

      demo d;

      d.get();

      d.disp();

      d.asc_sort();

      cout<<"\nAfter ascending sort :";

      d.disp();

      d.dsc_sort();

      cout<<"\n After Descending sort :";

      d.disp();

      getch();

}


*/ Output */


Enter the array size:4
Enter the array element:12 3 -45 -6


 The array element are  12         3        -45      -6

After ascending sort:

 The array element are  -45        -6       3        12

 After Descending sort:

 The array element are  12         3        -6       -45
```

- Assignment Name: Implementation of Insertion Sort

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

#include<math.h>

class insert

{

      int n,a[10],temp,ptr,q,i,j,k,key;

      public:

      void get();

      void sort();

      void display();

};

void insert::get()

{

      cout<<"\nEnter Range: ";

      cin>>n;

      for(i=1;i<=n;i++)

       a[i]=random(1000);

      cout<<"\nElements are: ";

      for(i=1;i<=n;i++)

       cout<<a[i]<<"\t";

}

void insert::sort()

{

      a[0]=-9999;

      for(i=2;i<=n;i++)

      {

            temp=a[i];

            ptr=i-1;
            while(temp<a[ptr])

            {

                  a[ptr+1]=a[ptr];

                  ptr--;

            }

            a[ptr+1]=temp;

      }

}

void insert::display()

{

      cout<<"\nSorted Element using Insertion Sort:
      "; for(i=1;i<=n;i++)

       cout<<a[i]<<"\t";

}

void main()

{

      clrscr();

      insert h;

      h.get();

      h.sort();

      h.display();

      getch();

}

*/ Output */

Enter Range: 5

Elements are: 10         3        335      33       355

Sorted Element using Insertion Sort: 3    10       33       335    355
```

- Assignment Name: Implementation of Radix Sort

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>
#include<conio.h>
#define max 100

class radixsort
{
      int i;
      public:
      void output(int *, int);
      void radix_sort(int *,int);
};

void radixsort::radix_sort(int a[],int n)
{
      int bucket[10][5],buck[10],b[10];
      int i,j,k,l,num,div,large,passes;
      div=1;
      num=0;
      large=a[0];
      for(i=1;i<n;i++)
      {
            if(a[i]>large)
            large=a[i];
      }
      while(large>0)
      {
            num++;
            large=large/10;
      }
```

```
     for(passes=0;passes<num;passes++)

     {

          for(k=0;k<10;k++)

          buck[k]=0;

               for(i=0;i<n;i++)

               {

                    l=(a[i]/div)%10;

                    bucket[l][buck[l]++]=a[i];

               }

               i=0;

          for(k=0;k<10;k++)

          {

               for(j=0;j<buck[k];j++)

               a[i++]=bucket[k][j];

          }

          div*=10;

     }

}


void radixsort::output(int a[],int n)

{

     for(i=0;i<=n-1;i++)

     {

     cout<<endl<<a[i];

     }

}
```

```
void main()
{
      int a[max],n,i;

      radixsort r;
      clrscr();
      cout<<"\nHow many elements in Array:";
      cin>>n;
      cout<<endl;
      cout<<"\nEnter Elements :"<<endl;
      for(i=0;i<=n-1;i++)
      {
            cin>>a[i];
      }
      r.radix_sort(a,n);
      cout<<"\n Sorted Array:";

      r.output(a,n);
      getch();
}
```

```
*/ Output */

How many elements in Array:  5

Enter Elements :335      260      33      10      3
Sorted Array :3        10          33          260        335
```

- Assignment Name: Implementation of Quick sort for integer

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

#include<string.h>


class demo

{

      int x[20],n;

      public:

      void get();

      void asort(int,int);

      int partition(int,int);

      void disp();

};



void demo::get()

{

      cout<<"\nEnter the array size:";

      cin>>n;

      cout<<"\nEnter the array element:";

      for(int i=1;i<=n;i++)

      cin>>x[i];

      asort(1,n);

}


void demo::asort(int p,int q)

{

      if(p<q)

      {
           int j=partition(p,q);

           asort(p,j-1);

           asort(j+1,q);

      }

}


int demo::partition(int lb, int ub)

{

      int a,left,right,temp;

      a=x[lb];

      left=lb+1;

      right=ub;

      do

      {

           while(x[left]<a) //change x[left]>a for
           descending left++;

           while(x[right]>a) //change x[right]<a for descending

           right--;

           if(left<right)

           {

                temp=x[left];

                x[left]=x[right];

                x[right]=temp;

           }

      }while(left<=right);
```

```
      x[lb]=x[right];
      x[right]=a;
      return(right);
}

void demo::disp()
{
      cout<<"\nThe array element are: ";
      for(int i=1;i<=n;i++)
      cout<<x[i]<<"\t";
}

void main()
{
      clrscr();
      demo d;
      d.get();
      cout<<"\nAfter Ascending sort";
      d.disp();
      getch();
}

*/ Output */
```

```
Enter the array size: 5

Enter the array element: 12 3 -45 -67 8

After Ascending sort
The array element are: -67      -45     3       8       12
```

- Assignment Name: Implementation of Merge sort

Class: MCA I **Lab: CA Lab III (DS)**

```
#include<iostream.h>

#include<conio.h>

#include<stdio.h>


class merge

{

      int a[10],n;

public:


      void read();

      void merge_sort(int l,int h);

      void merge1(int l,int m, int h);

      void disp();

};



void merge::read()

{

      cout<<"\n How many Elements you want to
      store:\n"; cin>>n;

      cout<<"\n Enter elements \n";

      for(int i=1;i<=n;i++)

            cin>>a[i];

      merge_sort(1,n);


}


void merge::merge_sort(int l,int h)

{
      int mid;

      if(l<h)

      {

            mid=int((l+h)/2);

            merge_sort(l,mid);

            merge_sort(mid+1,h);

            merge1(l,mid,h);

      }

}


void merge::merge1(int low,int mid,int high)

{

      int b[10];

      int i=low;

      int k=low;

      int j=mid+1;


      while((i<=mid)&&(j<=high))

      {


            if(a[i]<=a[j])//Change a[i]>=a[j] for
            descending {


                  b[k]=a[i];

                  i++;

                  k++;

            }

            else
```

```
          {
                b[k]=a[j];
                j++;
                k++;
          }
    }
          if(i>mid)
          {
                while(j<=high)
                {
                      b[k]=a[j];
                      j++;
                      k++;
                }
          }
          else
          {
                while(i<=mid)
                {
                      b[k]=a[i];
                      i++;
                      k++;
                }

          }

    for(int k1=low;k1<=high;k1++)
          a[k1]=b[k1];
}

void merge::disp()
{
    for(int i=1;i<=n;i++)
```

```
      cout<<a[i]<<"\t";
}



void main()
{
      clrscr();
      merge m;
      m.read();
      cout<<"\nAfter Sorting\n";
      m.disp();
      getch();
}
```

```
*/ Output */

How many Elements you want to store:
5

Enter elements
12 -34 5 67 -8



After Sorting
-34     -8      5       12      67
```

- Assignment Name: Implementation of Max Heap Tree using Insert

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

class heap

{

      int n,a[10],q,i,j,k,key;

public:

      void get();

      void create();

      void display();

};

void heap::get()

{

      cout<<"\nEnter Range:";

      cin>>n;

      cout<<"\nEnter the element:";

      for(i=1;i<=n;i++)

       cin>>a[i];

}

void heap::create()

{

      for(q=2;q<=n;q++)

      {

       i=q;

       key=a[q];

       j=i/2;

       while(i>1 && key>a[j]) //change

       {

            a[i]=a[j];

            i=j;
            j=i/2;


            if(j<1)

             j=1;

        }

      a[i]=key;

      }

}

void heap::display()

{

      cout<<"\nHeap Tree:";

      for(i=1;i<=n;i++)

       cout<<a[i]<<"\t";

}

void main()

{

      clrscr();

      heap h;

      h.get();

      h.create();

      h.display();

      getch();

}

*/ Output */

Enter Range:7

Enter the element:80 45 70 40 35 50  90

Heap Tree:90      45      80      40      35      50      70
```

```
---------------------------------------------------------------------------
- Assignment Name: Implementation of Min Heap Tree using Insert
Class: MCA I                                                 Lab: CA Lab III (DS)
-----------------------------------------------------------------------------
#include<iostream.h>

#include<conio.h>
class heap
{
      int n,a[10],q,i,j,k,key;
public:
      void get();
      void create();
      void display();
};
void heap::get()
{
      cout<<"\nEnter Range:";

      cin>>n;
      cout<<"\nEnter the element:";
      for(i=1;i<=n;i++)
       cin>>a[i];
}
void heap::create()
{
      for(q=2;q<=n;q++)
      {
       i=q;
       key=a[q];
       j=i/2;
       while(i>1 && key<a[j]) //change

       {
            a[i]=a[j];
            i=j;
            j=i/2;


            if(j<1)
             j=1;
        }
      a[i]=key;
      }
}
void heap::display()
{
      cout<<"\nHeap Tree";
      for(i=1;i<=n;i++)
       cout<<a[i]<<"\t";
}
void main()
{
      clrscr();
      heap h;
      h.get();
      h.create();
      h.display();
      getch();
}
*/ Output */
Enter Range:7
Enter the element:80 45 70  40 35 50 90
Heap Tree35      40      50       80      45      70     90
```

- Assignment Name: Implementation of Max heap using Heapify/Adjust

Class: MCA –I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>


class heap

{

  int i,j,item,a[10],n;

  public:

   void get();

   void show();

   void adjust(int [],int i,int j);

   void heapify(int [],int);

 };

void heap::get()

{

 cout<<"enter the size of array";

 cin>>n;

 for(i=1;i<=n;i++)

 cin>>a[i];

 heapify(a,n);

}

void heap::show()

{

  cout<<"\nThe element is=>\n";

  for(i=1;i<=n;i++)

  cout<<a[i]<<"\t";

}

void heap::adjust(int a[],int i,int n)

{

  j=2*i;
  item=a[i];
  while(j<=n)
  {
    if((j<n)&&(a[j]<a[j+1]))
    j++;
    if(item>=a[j])
    break;
    a[j/2]=a[j];
    j=2*j;
   }
   a[j/2]=item;
}
void heap::heapify(int a[],int n)
{
  for(i=n/2;i>=1;i--)
  adjust(a,i,n);
}
void main()
{
 clrscr();
 heap h;
 h.get();
 h.show();
 getch();
}
```

```
/*

output==>


enter the size of array


7

element are=>

10      3        335     33     355      217    536

ele after max heap=>

536     355      335     33     3        217    10

*/
```

```
-----------------------------------------------------------------------------
- Assignment Name: Implementation of Min Heap using Heapify / Adjust
Class: MCA -I                                          Lab: CA Lab III (DS)
------------------------------------------------------------------------------
#include<iostream.h>

#include<conio.h>


class heap

{

   int i,j,item, a[1000],n;

   public:

   void get();

   void show();

   void adjust(int [],int i,int j);

   void heapify(int [],int);

 };

void heap::get()

{

 cout<<"enter the size of array";

 cin>>n;

 for(i=1;i<=n;i++)

 cin>>a[i];

 heapify(a,n);

}

void heap::show()

{

  cout<<"\nthe element is=>\n";

  for(i=1;i<=n;i++)

  cout<<a[i]<<"\t";

}

void heap::adjust(int a[],int i,int n)

{

  j=2*i;
  item=a[i];
  while(j<=n)
  {
    if((j<n)&&(a[j]>a[j+1]))
    j++;
    if(item<=a[j])
    break;
    a[j/2]=a[j];
    j=2*j;
   }
   a[j/2]=item;
}
void heap::heapify(int a[],int n)
{
  for(i=n/2;i>=1;i--)
  adjust(a,i,n);
}
void main()
{
 clrscr();
 heap h;
 h.get();
 h.show();
 getch();
}
```

```
/*

output==>


enter the size of array

7


element are=>

10      3        335     33      355     217    536

element are=>

3       10       217     33      355     335    536


*/
```

- Assignment Name: Implementation of Heap Sort using Adjust/Heapify

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

#include<stdlib.h>

class Heap

{

     int b[50],n;

     public:

           void get();

           void heapsort(int a[],int n);

           void heapify(int a[],int n);

           void adjust(int a[],int i,int n);

           void disp();

};

void Heap::get()

{

      cout<<"\n Enter How many node wants to
      store:"; cin>>n;

      for(int i=1;i<=n;i++)

      cin>>b[i];

      heapsort(b,n);

}

void Heap::disp()

{

      for(int i=1;i<=n;i++)

      cout<<"\t"<<b[i];

      cout<<"\n";

}

void Heap::heapsort(int a[],int n)
```

```
{
      heapify(a,n);
      cout<<"\n MAX Heap Tree is :\n";
      disp();
      for(int i=n;i>=2;i--)
      {
        int t=a[i];
        a[i]=a[1];
        a[1]=t;
        adjust(a,1,i-1);
      }
}
void Heap::heapify(int a[],int n)
{
   int i;
   for(i=n/2;i>=1;i--)
   {
     adjust(a,i,n);
   }
}
void Heap::adjust(int a[],int i, int n)
{
   int j=2*i;
   int item=a[i];
   while(j<=n)
   {
      if((j<n) && (a[j]<a[j+1]))
      j=j+1;
      if(item>=a[j])
       return;
      else
      {
       a[j/2]=a[j];
       j=2*j;
      }
   }
   a[j/2]=item;
}

void main()
{
   clrscr();
   Heap h;
   h.get();
   cout<<"\n Sorted Heap Tree is :\n";
   h.disp();
   getch();
}
/* OUTPUT

Enter How many node wants to store:7

12  5  67  43 77 23 7



MAX Heap Tree is :

        77      43      67      12      5       23      7

Sorted Heap Tree is :
```

```
5        7        12       23       43       67       77
```

Assignment Name: Perform Insert, Display, delete, search, sum operation on Singly Linked List

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
class node
{
      int info,item,s;
      node *link;
public:
      void insert();
      void dis();
      void del();
      void search();
      void sum();
};
node *move,*start=NULL,*temp;

void node::insert()
{
      cout<<"\nEnter the item:";
      cin>>item;
      node *node1=new node;
      node1->link=NULL;
      node1->info=item;
      if(start==NULL)
            start=node1;
      else
      {
            move=start;
            while(move->link!=NULL)
            move=move->link;
            move->link=node1;
      }
}

void node::dis()
{
      node *x;
      x=start;
      cout<<"\n Elements in LL are:";
            while(x!=NULL)
            {
                  cout<<"\t"<<x->info;
                  x=x->link;
            }
}

void node::sum()
{
      node *x;
      x=start;
      s=0;
      while(x!=NULL)
      {
            s=s+x->info;
            x=x->link;
      }
      cout<<"\nSum of node is"<<s;
}

void node::del()
{
      temp=start;
      if(temp!=NULL)
      {
            temp=temp->link;
            cout<<"\nDeleted node is"<<start->info;
            start=temp;
      }
      else
            cout<<"\n List is empty:";
}

void node::search()
{
      int c=0,f=0,d;
      cout<<"\nEnter item";
      cin>>item;
      temp=start;
      while(temp!=NULL)
      {
            c++;
            if(temp->info==item)
            {
                  f=1;
                  d=c;
                  break;
            }
            temp=temp->link;
        }
        if(f==1)
            cout<<"\nElement is found at position "<<d;
        else
            cout<<"\nElement is not found";
}

void main()
{
        clrscr();
        node n;
        int ch;
        cout<<"\n1.Insert  2.Display 3. Delete 4.Search 5.Sum 6.Exit\n";

        do
        {
            cout<<"\nEnter choice";
            cin>>ch;
            switch(ch)
            {
                case 1: n.insert(); break;
                case 2: n.dis(); break;
                case 3: n.del(); break;
                case 4: n.search(); break;
                case 5: n.sum(); break;
                case 6: exit(0);
            }
      }while(ch!=6);
      getch();
}
```

```
*/ Output */


1.Insert  2.Display 3. Delete 4.Search 5.Sum 6.Exit

Enter choice1
Enter the item:10
Enter choice1
Enter the item:20
Enter choice1
Enter the item:30
Enter choice2
 Elements in LL are:     10      20     30
Enter choice3
Deleted node is10
Enter choice2
 Elements in LL are:     20      30
Enter choice5

Sum of node is50
Enter choice4

Enter item30

Element is found at position 2
Enter choice4

Enter item19
```

```
Element is not found

Enter choice 6
```

Assignment Name: Program to Implementation of Dynamic Stack using Linked List

Class: MCA I   Lab: CA Lab III (DS)

```
#include<conio.h>
#include<iostream.h>
#include<process.h>
class stack
{
	int info, ele;
	stack *node,*link,*top;
public:
	stack()
	{
		top=NULL;
	}
	void insert();
	void del();
	void dis();
};

void stack::insert()
{
	node=new stack;
	cout<<"\nEnter Info:";
	cin>>ele;
	node->info=ele;
	node->link=NULL;
	if(top==NULL)
	{
		top=node;
	}
```

```
      else

      {

            node->link=top;

            top=node;

      }

}

void stack::del()

{

      if(top==NULL)


      {

            cout<<"\n Underflow";

      }

      else

      {


            cout<<"\nDeleted Element is :"<<top-
            >info; top=top->link;


      }

}


void stack::dis()

{

      stack *move;


      move=top;

      while(move!=NULL)

      {

            cout<<"\t"<<move->info;

            move=move->link;
      }
}

void main()
{

      clrscr();
      int ch;
      stack s;
      cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";
      while(ch!=4)
      {
            cout<<"\nEnter Choice";
            cin>>ch;
            switch(ch)
            {
                  case 1: s.insert(); break;
                  case 2: s.dis(); break;

                  case 3: s.del(); break;
                  case 4:exit(0);
            }
      }
getch();
}

*/ Output */

1.Insert 2.Show 3.Delete 4.Exit
Enter Choice1

Enter Info:23

Enter Choice1
Enter Info:55


Enter Choice1


Enter Info:66


Enter Choice1


Enter Info:77


Enter Choice2
        77      66      55      23
Enter Choice3


Deleted Element is :77
Enter Choice2
        66      55      23
Enter Choice
```

- Assignment Name: Implementation of Dynamic Queue using Link List

Class: MCA I   Lab: CA Lab III (DS)

```
#include<conio.h>

#include<iostream.h>

#include<process.h>

class queue

{

      int info, ele,c;

      queue *node,*link,*start,*move;

public:

      queue()

      {

      start=NULL;

      c=0;

      }

      void insert();

      void del();

      void dis();

};


void queue::insert()

{

      node=new queue;

      if(c<3)

      {

            cout<<"\nEnter Info:";

            cin>>ele;

            node->info=ele;

            node->link=NULL;

            if(start==NULL)

            {
```

```
                        start=node;

                        c++;

                        return;

                  }

                  else

                  {

                        move=start;

                        while(move->link!=NULL)

                        move=move->link;

                        move->link=node;

                        c++;

                  }

            }

            else

                  cout<<"\n Overflow";

      }

      void queue::del()

      {

            move=start;

            if(move!=NULL)

            {

                  move=move->link;

                  cout<<"\nDeleted Element is :"<<start-
                  >info; start=move;

            }

            else

                  cout<<"\nUnderflow";
```

```
}
void queue::dis()
{
      move=start;
      if(move==NULL)
      {
            cout<<"\n Queue is empty ";
            return;
      }
      else
      {
            while(move!=NULL)
            {
                  cout<<move->info<<"\t";
                  move=move->link;
            }
      }
}
void main()
{
      clrscr();
      int ch;
      queue s;
      cout<<"\n1.Insert 2.Show 3.Delete 4.Exit";
      while(ch!=4)
      {
            cout<<"\nEnter Choice";
            cin>>ch;
            switch(ch)
            {
                  case 1: s.insert();break;
                  case 2: s.dis();break;
```

```
                case 3: s.del();break;

                case 4:exit(0);

            }

        }
getch();

}


*/ Output */



1.Insert 2.Show 3.Delete 4.Exit


Enter Choice2


Queue is empty


Enter Choice1


Enter Info:10


Enter Choice1


Enter Info:20


Enter Choice1


Enter Info:30
Enter Choice1


 Overflow

Enter Choice2

10      20      30

Enter Choice3

Deleted Element is :10

Enter Choice2

20      30
```

```
-----------------------------------------------------------------------------
- Assignment Name: Implementation of Priority Queue

Class: MCA I                                                  Lab: CA Lab III (DS)

-------------------------------------------------------------------------------

#include <iostream.h>


#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<conio.h>

/*


 * Node Declaration
 */
struct node

{

     int priority;

     int info;

     struct node *link;


};

/*


 * Class Priority
 Queue */
class Priority_Queue

{

    private:

        node *front;

    public:

        Priority_Queue()

        {

            front = NULL;

        }


        /*


         * Insert into Priority Queue
         */
        void insert(int item, int priority)
```

```
{

    node *tmp, *q;

    tmp = new node;

    tmp->info = item;

    tmp->priority = priority;

    if (front == NULL || priority < front->priority)

    {

        tmp->link = front;

        front = tmp;

    }

    else

    {

        q = front;

        while (q->link != NULL && q->link->priority <= priority)

            q=q->link;

        tmp->link = q->link;

        q->link = tmp;

    }

}

/*

 * Delete from Priority Queue
 */

void del()

{

    node *tmp;

    if(front == NULL)
```

```
                cout<<"Queue Underflow\n";

            else

            {

                tmp = front;

                cout<<"Deleted item is: "<<tmp->info<<endl;


                front = front->link;

                free(tmp);

            }

        }

        /*


         * Print Priority
         Queue */
        void display()

        {

            node *ptr;

            ptr = front;

            if (front == NULL)


                cout<<"Queue is empty\n";

            else


            {
                cout<<"Queue is
                :\n"; cout<<"Priority Item\n";
                while(ptr != NULL) {




                    cout<<ptr->priority<<"                    "<<ptr->info<<endl;

                    ptr = ptr->link;

                }

            }

        }

};

/*


 * Main
 */
int main()
```

```
{
      clrscr();
    int choice, item, priority;
    Priority_Queue pq;
    do
    {
      cout<<"1.Insert\n";
      cout<<"2.Delete\n";
      cout<<"3.Display\n";
      cout<<"4.Quit\n";
      cout<<"Enter your choice : ";
      cin>>choice;
      switch(choice)
      {
      case 1:
          cout<<"Input the item value to be added in the queue :
          "; cin>>item;
          cout<<"Enter its priority : ";
          cin>>priority;
          pq.insert(item, priority);
          break;
      case 2:
          pq.del();
          break;
      case 3:
          pq.display();
          break;
      case 4:
          break;
      default :
          cout<<"Wrong choice\n";
      }
    }
    while(choice != 4);
    getch();
    return 0;
}

/*
4                 15
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Deleted item is: 12
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Deleted item is: 10
1.Insert
2.Delete
3.Display
4.Quit
Enter your choice : 2
Deleted item is: 15
1.Insert
```

```
2.Delete

3.Display

4.Quit

Enter your choice : 4


*/
```

--
- Assignment Name: Implementation of Doubly Link List

Class: MCA I   Lab: CA Lab III (DS)

--

```
#include<iostream.h>

#include<conio.h>

#include<process.h>

class node

{

      int info,c,j;

      node *left,*right;

public:

      void insert();

      void display();

      void del();

};


node *start=NULL,*temp=NULL,*move=NULL, *temp1=NULL;

void node::insert()

{

      int item;

      node *p=new node;

      cout<<"\nEnter element:";

      cin>>item;

      p->info=item;

      p->left=NULL;

      p->right=NULL;

      if(start==NULL)

      {

            start=p;

            return;

      }

      else
      {
            temp=start;
            while(temp->right!=NULL)
            temp=temp->right;
            temp->right=p;
            p->left=temp;
      }
}

void node::display()
{
      move=start;
      if(move==NULL)
      {
            cout<<"\n LL Empty:";
            return;
      }
      else
      {
            cout<<"\n node in DLL are :";
            while(move!=NULL)
            {
                  cout<<move->info<<"\t";
                  move=move->right;
            }
      }
```

```
}

void node::del()
{
      if(start==NULL)
      {
            cout<<"\n LL Empty:";
            return;
      }
      temp=start;
      start=temp->right;
      start->left=NULL;
      temp->right=NULL;
      cout<<"\n deleted element is"<<temp->info;
}


void main()
{
      clrscr();
      node n;
      int ch;
      cout<<"\n1. Insert 2. Display 3.Delete 4. Exit";
      while(ch!=4)
      {
            cout<<"\nEnter choice";
            cin>>ch;
```

```
          switch(ch)

          {

               case 1: n.insert(); break;

               case 2: n.display(); break;

               case 3: n.del(); break;

               case 4: exit(0);

          }

     }

     getch();

}
```

```
*/ Output */




1. Insert 2. Display 3.Delete
4. Exit Enter choice2


 LL Empty:
Enter choice1



Enter element:10



Enter choice1



Enter element:20



Enter choice1



Enter element:30
Enter choice2

 node in DLL are :10     20     30

Enter choice3

 deleted element is10
Enter choice2

  node in DLL are :20    30
 Enter choice3

 deleted element is20
Enter choice3

 deleted element is30
Enter choice2

 LL E
mpty:
Enter
choic
e3

 LL E
mpty
:
Ente
r
choi
ce
```

Assignment Name: Implementation of Circular Link List

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>

#include<process.h>

class node

{

      int info,c,i;

      node *link;

public:

      node()

      {

            c=0;

      }

      void insert();

      void display();

      void del();

};


node *start=NULL, *temp=NULL,*move=NULL, *temp1=NULL;

void node::insert()

{

      int item;

      node*p=new node;

      cout<<"\nEnter Element:";

      cin>>item;

      p->info=item;

      p->link=NULL;

      if(start==NULL)

      {
            start=p;

            p->link=start;

            c++;

      }

      else

      {

            temp=start;

            while(temp->link!=start)

            temp=temp->link;


            temp->link=p;

            p->link=start;

            c++;

      }

}


void node::display()

{

      if(start==NULL)

      {

            cout<<"\n LL empty";

            return;

      }


            node *temp;

            temp=start;

            move=start->link;

            cout<<temp->info;

            while(move!=start)
```

```
          {
               cout<<"->"<<move->info;
               move=move->link;
          }
     cout<<"\n Number of nodes in CLL are :"<<c;
}

void node::del()
{
     int pos;
     cout<<"\nEnter Position:";
     cin>>pos;
     if(c==1)
     {
          start=NULL;
     }
     if(start==NULL)
     {
          cout<<"\n LL Empty:";
          return;
     }

     if(pos>c||pos<1)
     {
          cout<<"\nInvalid Position";
          return;
     }

     if(pos==1)
     {
          temp=start;
          while(temp->link!=start)
```

```
            temp=temp->link;
            temp1=start;
            start=start->link;
            temp->link=start;
            cout<<"\nDeleted Element is "<<temp1->info;
            delete(temp1);
            c--;
        }
        else
        {

            temp=start;
            i=1;
            while(i<pos-1)
            {
                temp=temp->link;
                i++;
            }

            temp1=temp->link;
            temp->link=temp1->link;
            cout<<"\nDeleted element is"<<temp1->info;
            delete(temp1);
            c--;

        }
}



void main()
```

```
{
      clrscr();
      node n;
      int ch;
      cout<<"\n 1.Insert 2.Display 3.Delete 4.Exit";
      while(ch!=4)
      {
            cout<<"\n Enter Choice";
            cin>>ch;
            switch(ch)
            {
                  case 1: n.insert(); break;
                  case 2: n.display(); break;
                  case 3: n.del(); break;
                  case 4: exit(0);
            }
      }getch();
}
```

*/ Output */

```
 1.Insert 2.Display 3.Delete 4.Exit
 Enter Choice1

Enter Element:10

 Enter Choice1

Enter Element:20

 Enter Choice2
10->20
```

```
 Number of nodes in CLL are :2

 Enter Choice3


Enter Position:2


Deleted element is20

 Enter Choice2

10

 Number of nodes in CLL are :1


 Enter Choice3


Enter Position:1


 LL Empty:
 Enter Choice2


 LL empty Enter
 Choice 4
```

```
---------------------------------------------------------------------------

Assignment Name: Implementation of Binary search Tree Insertion, Deletion,

                 Search & Recursive Tree Traversal

Class: MCA I                                        Lab: CA Lab III (DS)

---------------------------------------------------------------------------
```

```
# include<conio.h>
# include<process.h>
# include<iostream.h>
# include<alloc.h>

struct node

{

      int ele;

      node *left;

      node *right;

};


typedef struct node *nodeptr;


class bstree

{

      public:

            void insert(int,nodeptr &);

            void del(int,nodeptr &);

            int deletemin(nodeptr &);

            void find(int,nodeptr &);

            void preorder(nodeptr);

            void inorder(nodeptr);

            void postorder(nodeptr);


};


void bstree::insert(int x,nodeptr &p)

{

      if (p==NULL)
```

```
    {
        p = new node;
        p->ele=x;
        p->left=NULL;
        p->right=NULL;
    }
    else
    {
        if (x < p->ele)
            insert(x,p->left);
        else if (x>p->ele)
            insert(x,p->right);
        else
            cout<<"Element already Exits !\n";
    }
}

void bstree:: del(int x,nodeptr &p)
{
    nodeptr d;
    if (p==NULL)
        cout<<"Element not found\n";
    else if (x < p->ele)
        del(x,p->left);
    else if (x > p->ele)
```

```
            del(x,p->right);
      else if ((p->left == NULL) && (p->right ==NULL))
      {
            d=p;
            free(d);
            p=NULL;
      }
       else if (p->left == NULL)
      {
            d=p;
            free(d);
            p=p->right;
      }
      else if (p->right ==NULL)
      {
            d=p;
            p=p->left;
            free(d);
      }
      else
      p->ele=deletemin(p->right);
}

int bstree::deletemin(nodeptr &p)
{
      int c;
      if (p->left == NULL)
      {
            c=p->ele;
            p=p->right;
            return c;
      }
      else
            c=deletemin(p->left);
            return c;
}




void bstree::find(int x,nodeptr &p)
{
      if (p==NULL)
            cout<<"Element not found !\n";
      else
      {
            if (x <p->ele)
                  find(x,p->left);
            else if ( x> p->ele)
                  find(x,p->right);
            else
                  cout<<"Element Found !\n";
      }
}

void bstree::preorder(nodeptr p)

{
      if (p!=NULL)
      {
            cout<<p->ele<<"-->";
            preorder(p->left);
            preorder(p->right);
      }
}
void bstree::inorder(nodeptr p)
{
      if (p!=NULL)
      {
            inorder(p->left);
            cout<<p->ele<<"-->";
            inorder(p->right);
      }
}

void bstree::postorder(nodeptr p)
{
      if (p!=NULL)
      {
            postorder(p->left);
            postorder(p->right);
            cout<<p->ele<<"-->";
      }
}



void main()
{
clrscr();
int ch,x;
bstree bst;
char c='y';
nodeptr root;
```

```
root=NULL;

do

{

      clrscr();


      cout<<"\n-------------------------";
      cout<<"\nBinary Search Tree"; cout<<"\n-----------
      --------------\n"; cout<<"\n1.Insertion 2.Deletion
      3.Find 4.Preorder"; cout<<"\n5.Inorder 6.Postorder
      7.Exit "; cout<<"Enter your choice :\n"; cin>>ch;





      switch(ch)

      {

      case 1:

            cout<<"1.Insertion\n";


            cout<<"Enter the new element to get inserted
            :\n"; cin>>x;


            bst.insert(x,root);

            cout<<"Inorder traversal is :";

            bst.inorder(root);

            break;


      case 2:

            cout<<"2.Deletion\n";


            cout<<"Enter the element to get deleted
            :\n"; cin>>x;
```

```
          bst.del(x,root);

          bst.inorder(root);

          break;


     case 3:


          cout<<"3.Find\n";

          cout<<"Enter the element to be searched :\n";

          cin>>x;

          bst.find(x,root);

          break;


     case 4:

          cout<<"4.Preorder\n";

          if (root==NULL)

               cout<<"Tree is empty\n";

          else

          {


               cout<<"Preorder traversal (Recursive) is
               :\n"; bst.preorder(root);

          }

          break;

     case 5:

          cout<<"5.Inorder\n";

          if (root==NULL)

               cout<<"Tree is empty";

          else

          {


               cout<<"Inorder traversal (Recursive) is
               :\n"; bst.inorder(root);

          }

          break;


     case 6:
            cout<<"6.Postorder\n";

            if (root==NULL)

                  cout<<"Tree is empty";

            else

            {


                  cout<<"Postorder traversal (Recursive) is
                  :\n"; bst.postorder(root);

            }

            break;


      case 7:

            exit(0);

      }

      cout<<"\nContinue (y/n) ? ";

      cin>>c;

      }while (c=='y' || c == 'Y');

      getch();


}
```

Assignment Name: Implementation of Binary search Tree Insertion, Deletion and

Non Recursive Tree Traversals

Class: MCA I   Lab: CA Lab III (DS)

```
# include<conio.h>
# include<process.h>
# include<iostream.h>

# include<alloc.h>
struct node

{


      int ele;
      node *left;
      node *right;

};



typedef struct node *nodeptr;


class stack

{

      private:

            struct snode

            {

                  nodeptr ele;

                  snode *next;

            };

            snode *top;

      public:

            stack()

            {

                  top=NULL;


            }

            void push(nodeptr p)

            {

                  snode *temp;
```

```
     temp = new snode;
     temp->ele = p;
     temp->next = top;
     top=temp;
}

void pop()
{

     if (top != NULL)
     {
     nodeptr t;
     snode *temp;
     temp = top;
     top=temp->next;
     delete temp;
     }
}

nodeptr topele()
{
     if (top !=NULL)

          return top->ele;
     else
          return NULL;
}
```

```
          int isempty()
          {
          return ((top == NULL) ? 1 : 0);
          }

};



class bstree
{
     public:
          void insert(int,nodeptr &);
          void del(int,nodeptr &);
          int deletemin(nodeptr &);
          void preordernr(nodeptr);
          void inordernr(nodeptr);
          void postordernr(nodeptr);



};

void bstree::insert(int x,nodeptr &p)
{
     if (p==NULL)
     {
          p = new node;
          p->ele=x;
          p->left=NULL;
          p->right=NULL;
     }
     else
     {
          if (x < p->ele)
```

```
                insert(x,p->left);
            else if (x>p->ele)
                insert(x,p->right);
            else
                cout<<"Element already Exits !\n";
        }
}

void bstree:: del(int x,nodeptr &p)
{
        nodeptr d;
        if (p==NULL)
            cout<<"Element not found\n";
        else if (x < p->ele)
            del(x,p->left);
        else if (x > p->ele)
            del(x,p->right);
        else if ((p->left == NULL) && (p->right ==NULL))
        {
            d=p;
            free(d);
            p=NULL;
        }
         else if (p->left == NULL)
        {
            d=p;
            free(d);
            p=p->right;
      }
      else if (p->right ==NULL)
      {
            d=p;
            p=p->left;
            free(d);
      }
      else
      p->ele=deletemin(p->right);
}

int bstree::deletemin(nodeptr &p)
{
      int c;
      if (p->left == NULL)
      {
            c=p->ele;
            p=p->right;
            return c;
      }
      else
            c=deletemin(p->left);
            return c;
}

void bstree::preordernr(nodeptr p)
{
      stack s;
      while (1)
      {
      if  (p != NULL)
```

```
	{
		cout<<p->ele<<"-->";
		s.push(p);
		p=p->left;
	}
	else
	if (s.isempty())
	{
		  //cout<<"Stack is empty";
		return;

	}
	else
	{
		nodeptr t;
		t=s.topele();
		p=t->right;
		s.pop();
	}
	}
}

void bstree::inordernr(nodeptr p)
{

	stack s;
	while (1)
	{
	if  (p != NULL)
	{
```

```
          s.push(p);

          p=p->left;

      }

      else

      {

      if (s.isempty())

      {

           //   cout<<"Stack is
          empty"; return;
      }

      else

      {

          p=s.topele();

          cout<<p->ele<<"-->";

      }

      s.pop();

      p=p->right;

      }

      }

}


void bstree::postordernr(nodeptr p)

{

      stack s;

      while (1)

      {

      if  (p != NULL)

      {

          s.push(p);

          p=p->left;


      }

      else
```

```
        {

                if (s.isempty())

                {

                          //  cout<<"Stack is
                        empty"; return;
                }

                else

                if (s.topele()->right == NULL)

                {

                        p=s.topele();

                        s.pop();

                        cout<<p->ele<<"-->";

                        if (p==s.topele()->right)

                        {

                                cout<<s.topele()->ele<<"-->";

                                s.pop();

                        }

                }

                if (!s.isempty())

                        p=s.topele()->right;

                else

                        p=NULL;

        }

        }

}


void main()
```

```
{

clrscr();

int ch,x;

bstree bst;

char c='y';

nodeptr root;

root=NULL;

do

{

      clrscr();

      cout<<"\n-------------------------";

      cout<<"\nBinary Search Tree";

      cout<<"\n-------------------------\n";

      cout<<"\n1.Insertion 2.Deletion 3.Preorder 4.Inorder 5.Postorder 6.Exit\n
";

      cout<<"Enter your choice :\n";

      cin>>ch;


      switch(ch)

      {

      case 1:

            cout<<"    1.Insertion\n";


            cout<<"Enter the new element to get inserted
            :\n"; cin>>x;


            bst.insert(x,root);

            cout<<"Inorder traversal is :";

            bst.inordernr(root);

            break;


      case 2:

            cout<<" 2.Deletion \n";

            cout<<"Enter the element to get deleted :\n";
```

```
        cin>>x;

        bst.del(x,root);

        bst.inordernr(root);

        break;


    case 3:

        cout<<"3.Preorder\n";

        if (root==NULL)

             cout<<"Tree is empty\n";

        else

        {

             cout<<"\nPreorder traversal (Non-Recursive) is
             :\n"; bst.preordernr(root);

        }

        break;

    case 4:

        cout<<"4.Inorder\n";

        if (root==NULL)

             cout<<"Tree is empty";

        else

        {

             cout<<"Inorder traversal (Non-Recursive) is
             :\n"; bst.inordernr(root);

        }

        break;


    case 5:
            cout<<"5.Postorder\n";

            if (root==NULL)

                  cout<<"Tree is empty";

            else

            {


                  cout<<"Postorder traversal (Non-Recursive) is
                  :\n"; bst.postordernr(root);



            }

            break;

      case 6:

            exit(0);

      }

      cout<<"\nContinue (y/n) ? ";

      cin>>c;

      }while (c=='y' || c == 'Y');

      getch();

}
```

- Assignment Name: Implementation of DFS

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>


class dfstree

{

      int a[20][20], visited[20],n,i,j;

public:

      void dfs(int);

      void get();

};


void dfstree::get()

{

      cout<<"\nEnter the number of node";

      cin>>n;

      for(i=0;i<n;i++)

       visited[i]=0;

      cout<<"\nEnter the adjancy matrix:";

      for(i=0;i<n;i++)

      {

       for(j=0;j<n;j++)

        {

          cin>>a[i][j];

        }

      }

      dfs(0);

}

void dfstree::dfs(int v)

{
      int k;

      visited[v]=1;

      cout<<"\t"<<v+1;

      for(k=1;k<n;k++)

      if(a[v][k]==1)

      if(visited[k]==0)

       dfs(k);

}


void main()

{

      clrscr();

      dfstree d;

      d.get();

      getch();

}
```

```
*/ Output */

Enter the number of node5


Enter the adjancy matrix:

0 1 1 0 0

1 0 0 1 1

1 0 0 1 0

0 1 1 0 1

0 1 0 1 0


1       2       4       3       5
```

- Assignment Name: Implementation of BFS

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>


class bfstree

{

      int
reach[20],a[20][20],q[20],n,i,j,f,r,index; public:

      bfstree()

      {

       f=r=0;

       index=1;

      }

      void get();

      void bfs();

};


void bfstree::get()

{

      cout<<"\nEnter number of vertices:";

      cin>>n;

      cout<<"\nEnter Adjacency matrix:";

      for(i=1;i<=n;i++)

      for(j=1;j<=n;j++)

      {

       reach[i]=0;

       cin>>a[i][j];

      }

}
```

```
void bfstree::bfs()

{

      reach[1]=1;

      f++;

      r++;

      q[r]=index;

      cout<<"\nBFS is ";

      while(f<=r)

      {

       index=q[f];

       f++;

       cout<<index<<"\t";

        for(j=1;j<=n;j++)

        {

          if(a[index][j]==1 && reach[j]!=1)

          {

            reach[j]=1;

            r++;

            q[r]=j;

          }

        }

       }

}


void main()

{

      clrscr();
      bfstree b;

      b.get();

      b.dbfs();

      getch();

}


*/ Output */


Enter number of vertices:6


Enter Adjacency matrix:

0 1 1 0 0 0

1 0 0 1 0 0

1 0 0 0 0 1

0 1 0 0 1 1

0 0 0 1 0 0

0 0 1 1 0 0


BFS is 1         2        3        4        6       5
```

- Assignment Name: Implementation of All Pair Shortest Path

Class: MCA I   Lab: CA Lab III (DS)

```
#include<iostream.h>

#include<conio.h>


class path

{

      int a[5][5],i,j,k,n,s,d;

public:

      void insert();

      void display();

};


void path::insert()

{

      cout<<"\nEnter the no. of vertices";

      cin>>n;

      cout<<"\nEnter the matrix:";

      for(i=1;i<=n;i++)

       for(j=1;j<=n;j++)

        {

          cin>>a[i][j];

          if(a[i][j]==-1)

            a[i][j]=9999;

        }


      for(i=1;i<=n;i++)

       for(j=1;j<=n;j++)

        for(k=1;k<=n;k++)

         if(a[i][j]<(a[i][k]+a[k][j]))

           a[i][j]=a[i][j];
        else

         a[i][j]=(a[i][k]+a[k][j]);

}


void path::display()

{

     for(i=1;i<=n;i++)

     {

      for(j=1;j<=n;j++)


       cout<<"\t"<<a[i][j];

       cout<<"\n";

     }

cout<<"\nEnter the source vertex:";

cin>>s;


cout<<"\nEnter the destination
vertex:"; cin>>d;


cout<<"\nPath from Source "<<s<<" to destination "<<d<<" is ";

cout<<a[s][d];

}


void main()

{


     clrscr();

     path p;

     p.insert();

     cout<<"\n Shortest path is \n";

     p.display();
     getch();

}


*/ Output */


Enter the no. of vertices 3


Enter the matrix:0 4 11

                 6 0 2

                 3 -1 0


  Shortest path is

         0       4      6

         5       0      2

         3       7      0


Enter the source vertex:3


Enter the destination vertex:2


Path from Source 3 to destination 2 is 7
```

- Assignment Name: Implementation of Prims Algorithm

Class: MCA I   Lab: CA Lab III (DS)

```
#include<conio.h>

#include<iostream.h>


int g[100][100],tree[100],n;
class spanning
{
public:
void get()
{
      cout<<"Enter the number of nodes ";
      cin>>n;
      cout<<"\nEnter the graph \n";
      for(int i=1;i<=n;i++)

      {
            for(int j=1;j<=n;j++)
            {
                  cin>>g[i][j];
            }
      }
}
void dis()
{
      cout<<"\nThe graph is \n";
      for(int i=1;i<=n;i++)
      {
            cout<<"\n";

            for(int j=1;j<=n;j++)
            {
                  cout<<g[i][j]<<" ";
            }

      }

}

void prims()

{

      int total=0,v1,v2;

      for(int l=1;l<=n;l++)

      {

            tree[l]=0;

      }


      tree[1]=1;

      cout<<"\nv1 v2 min_dist";


      for(int k=2;k<=n;k++)

      {

            int min_dist=30000;

            for(int i=1;i<=n;i++)

            {

            for(int j=1;j<=n;j++)

            {

            if(g[i][j]&&((tree[i] && !tree[j])||(!tree[i]&&tree[j])))

            {

                                    if(g[i][j]<min_dist)

                                    {

                                            min_dist=g[i][j];

                                            v1=i;
```

```
                                v2=j;
                           }
               }
               }
               }
               cout<<"\n"<<v1<<"   "<<v2<<"   "<<min_dist;
               tree[v1]=1;
               tree[v2]=1;
               total=total+min_dist;
          }
          cout<<"\n cost of spanning tree is "<<total;
     }
     };

     void main()
     {
          spanning s;
          clrscr();
          s.get();
          s.dis();
          s.prims();
          getch();
     }

     /* Output */
```

```
Enter the number of nodes 6


Enter the graph

0 3 0 0 2 0

3 0 2 4 0 0

0 2 0 0 5 2

0 4 0 0 1 0

2 0 5 1 0 6

0 0 2 0 6 0


The graph is


0 3 0 0 2 0

3 0 2 4 0 0

0 2 0 0 5 2

0 4 0 0 1 0

2 0 5 1 0 6

0 0 2 0 6 0


v1 v2 min_dist

1  5    2

4  5    1

1  2    3

2  3    2

3  6    2
```

```
cost of spanning tree is 10
```