SPCA110 / SPIT104

POSTGRADUATE COURSE

MASTER OF COMPUTER APPLICATIONS / M.Sc. INFORMATION TECHNOLOGY

FIRST YEAR

SECOND SEMESTER / FIRST SEMESTER

PRACTICAL - III / PRACTICAL - I

DATA STRUCTURES USING C++ LAB

INSTITUTE OF DISTANCE EDUCATION
UNIVERSITY OF MADRAS

M.C.A. / M.Sc., (IT)
FIRST YEAR
SEMESTER - II / SEMESTER - I

PRACTICAL - III / PRACTICAL - I
DATA STRUCTURE USING C++ LAB

WELCOME

Warm Greetings.

It is with a great pleasure to welcome you as a student of Institute of Distance Education, University of Madras. It is a proud moment for the Institute of Distance education as you are entering into a cafeteria system of learning process as envisaged by the University Grants Commission. Yes, we have framed and introduced Choice Based Credit System(CBCS) in Semester pattern from the academic year 2018-19. You are free to choose courses, as per the Regulations, to attain the target of total number of credits set for each course and also each degree programme. What is a credit? To earn one credit in a semester you have to spend 30 hours of learning process. Each course has a weightage in terms of credits. Credits are assigned by taking into account of its level of subject content. For instance, if one particular course or paper has 4 credits then you have to spend 120 hours of self-learning in a semester. You are advised to plan the strategy to devote hours of self-study in the learning process. You will be assessed periodically by means of tests, assignments and quizzes either in class room or laboratory or field work. In the case of PG (UG), Continuous Internal Assessment for 20(25) percentage and End Semester University Examination for 80 (75) percentage of the maximum score for a course / paper. The theory paper in the end semester examination will bring out your various skills: namely basic knowledge about subject, memory recall, application, analysis, comprehension and descriptive writing. We will always have in mind while training you in conducting experiments, analyzing the performance during laboratory work, and observing the outcomes to bring out the truth from the experiment, and we measure these skills in the end semester examination. You will be guided by well experienced faculty.

I invite you to join the CBCS in Semester System to gain rich knowledge leisurely at your will and wish. Choose the right courses at right times so as to erect your flag of success. We always encourage and enlighten to excel and empower. We are the cross bearers to make you a torch bearer to have a bright future.

With best wishes from mind and heart,

DIRECTOR

M.C.A. / M.Sc., (IT)
FIRST YEAR
SEMESTER - II / SEMESTER - I

PRACTICAL - III / PRACTICAL - I
DATA STRUCTURE USING C++ LAB

COURSE WRITER

Mrs. V. PARIMALA RANGANAYAKI
Asst. Prof. in Computer Science
D.B Jain College
Thoraipakkam,
Chennai - 600 097.

EDITING

Dr. (Mrs.) S. SASIKALA, M.C.A., M.Phil., Ph.D.,
Asst. Prof. in Computer Science
Institute of Distance Education
University of Madras
Chennai - 600 005.

© UNIVERSITY OF MADRAS, CHENNAI 600 005.

M.C.A. / M.Sc., (IT)

FIRST YEAR

SEMESTER - II / SEMESTER - I

PRACTICAL - III / PRACTICAL - I

DATA STRUCTURE USING C++ LAB

SYLLABUS

1. Implementation of Arrays (Single and Multi-Dimensional)
2. Polynomial Object and necessary overloaded operators.
3. Singly Linked Lists.
4. Circular Linked Lists.
5. Doubly Linked Lists.
6. General Lists.
7. Implementation of Stack (using Arrays and Pointers)
8. Implementation of Queue (Using Arrays and Pointers)
9. Implementation of Circular Queue (using Arrays and Pointers)
10. Evaluation of Expressions.
11. Binary Tree implementations and Traversals.
12. Binary Search Trees.
13. Shortest path (Dijkstra's algorithm).
14. Search methods in graphs (DFS and BFS) using recursion.

M.C.A. / M.Sc., (IT)

FIRST YEAR

SEMESTER - II / SEMESTER - I

PRACTICAL - III / PRACTICAL - I

DATA STRUCTURE USING C++ LAB

SCHEME OF LESSONS

Instructions:

1. Click the Start button then select run popup then give cmd command.
2. Make a directory in command prompt using md command (eg. Md sample)
3. Change the directory using cd command (eg. Cd sample).
4. Set the path in your directory(eg. Sample/>path=z:\tcc\bin)
5. Then give the tc press enter key for run the turbo c software(where tc executable file for turbo c).

INSTITUTE OF DISTANCE EDUCATION
UNIVERSITY OF MADRAS

CHENNAI - 600 005

RECORD OF PRACTICALS

M.C.A. (First Year)

20____ - 20____

Practical - IV

DATA STRUCTURES USING C++ LAB

NAME :

ENROLMENT NUMBER :

GROUP NO. :

INSTITUTE OF DISTANCE EDUCATION
UNIVERSITY OF MADRAS

CHENNAI - 600 005.

Certified that this is the Bonafide Record of work done by ____________________ with Enrolment Number _______________________ of First Year M.C.A. Degree Course in the Institute of Distance Education, University of Madras during the year ____________ in respect of **DATA STRUCTURES USING C++**

Date: **Co-ordinator**

Submitted for First Year M.C.A. Degree Course Practical Examination held on ____________________ at ____

Date: **Examiners**

1. **Name:**

Signature:

2. **Name:**

Signature:

Lesson 1

SIMPLE PROGRAMS

a. write program to find roots of the equation

```
#include<iostream.h>
#include<stdio.h>
#include<conio.h>
#include<math.h>
class equation
{
private:
float a;
float b;
float c;
public:
inline void getinfo(float a,float b, float c);
inline void display();
inline void equal(float a,float b);
inline void imag();
inline void real(float a,float b,float det);
};
inline void equation::getinfo(float aa,float bb,float cc)
{
a=aa;
b=bb;
c=cc;
}
inline void equation::display()
{
cout<<endl;
cout<<“a=”<<a<<“\n”;
cout<<“b=”<<b<<“\n”;
cout<<“c=”<<c<<endl;
}
```

```
inline void equation::equal(float a,float b)
{
float x;
x=-b/(2*a);
cout<<“Roots are equal=”<<x<<endl;
}
inline void equation::imag()
{
cout<<“Roots are imaginary\n”;
}
inline void equation::real(float a,float b,float det)
{
float x1,x2,temp;
temp=sqrt(det);
x1=(-b+temp)/(2*a);
x2=(-b-temp)/(2*a);
cout<<“Roots are real\n”;
cout<<“x1=”<<x1<<endl;
cout<<“x2=”<<x2<<endl;
}
void main(void)
{
class equation eq;
float aa,bb,cc;
clrscr();
cout<<“Enter three numbers\n”;
cin>>aa>>bb>>cc;
eq.getinfo(aa,bb,cc);
eq.display();
if(aa==0)
{
float temp;
temp=cc/bb;
cout<<“linear roots+”<<temp<<endl;
}
else
{
float det;
det=bb*bb-4*aa*cc;
if(det==0)
eq.equal(aa,bb);
else if(det<0)
eq.imag();
else
eq.real(aa,bb,det);
}
getch();
}
```

Ouput

```
Enter three numbers
1
-4
4

a=1
b=-4
c=4
Roots are equal=2
```

b. Write program to demonstrate String Functions.

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<stdio.h>
class string
{
char str[80];
public:
```

```
string()
{
strcpy (str,” “ );
}
friend istream&operator>>(istream&,string&);
friend ostream&operator<<(ostream&,string&);
string operator+(string a);
int operator==(string a);
int operator!=(string a);
int operator<(string a);
int operator>(string a);
};
istream&operator>>(istream&in,string&s)
{
gets(s.str);
return in;
}
ostream&operator<<(ostream&out,string&s)
{
puts(s.str);
return out;
}
string string::operator+(string b)
{
string c;
strcpy(c.str,str);
strcat(c.str,b.str);
return (c);
}
int string::operator==(string b)
{
if(strcmp(str,b.str)==0)
return 1;
else
```

```
return 0;
}
int string::operator!=(string b)
{
if(strcmp(str,b.str)==0)
return 0;
else
return 1;
}
int string::operator<(string b)
{
if(strcmp(str,b.str)<0)
return 1;
else
return 0;
}
int string::operator>(string b)
{
if(strcmp  (str,b.str)>0)
return 1;
else
return 0;
}
void main()
{
string s1,s2,s3,s4;
clrscr();
cout<<“\nEnter the First String:”;
cin>>s1;
cout<<“\nEnter the Second String:”;
cin>>s2;
cout<<“String 1 is . . .”;
cout<<s1<<endl;
cout<<“String 2 is . . .”;
```

```
cout<<s2<<endl;
s3=s1+s2;
cout<<“Concatenated String is . . .\n”;
cout<<s3<<endl;
if(s1==s2)
cout<<“Strings are equal . . .\n”;
if(s1!=s2)
cout<<“Strings are not equal . . .\n”;
if(s1<s2)
cout<<“String 1 is less than String 2 . . .\n”;
if(s1>s2)
cout<<“String 1 is greater than String 2 . . .\n”;
s4=s1;
cout<<“Copy of String 1 is . . .\n”;
cout<<s4;
getch();
}
```

<u>Output</u>

```
Enter the First String:hello

Enter the Second String:madam
String 1 is . . .hello

String 2 is . . .madam

Concatenated String is . . .
hellomadam

Strings are not equal . . .
String 1 is less than String 2 . . .
Copy of String 1 is . . .
hello
```

Lesson 2

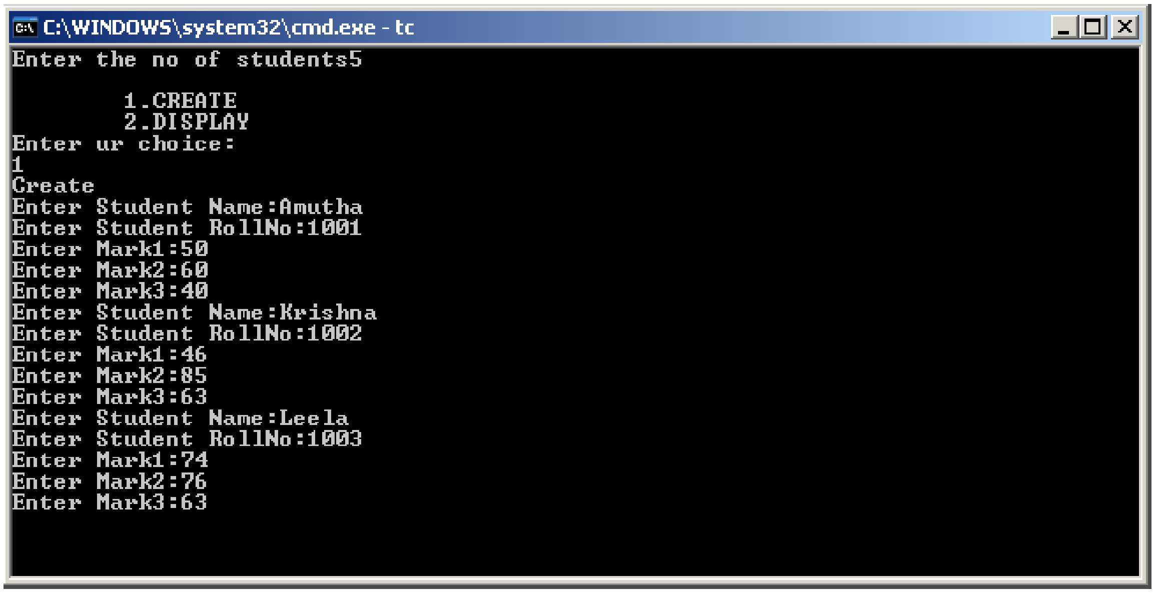
ARRAY'S

a. To write a C++ program for creating student marklist using array

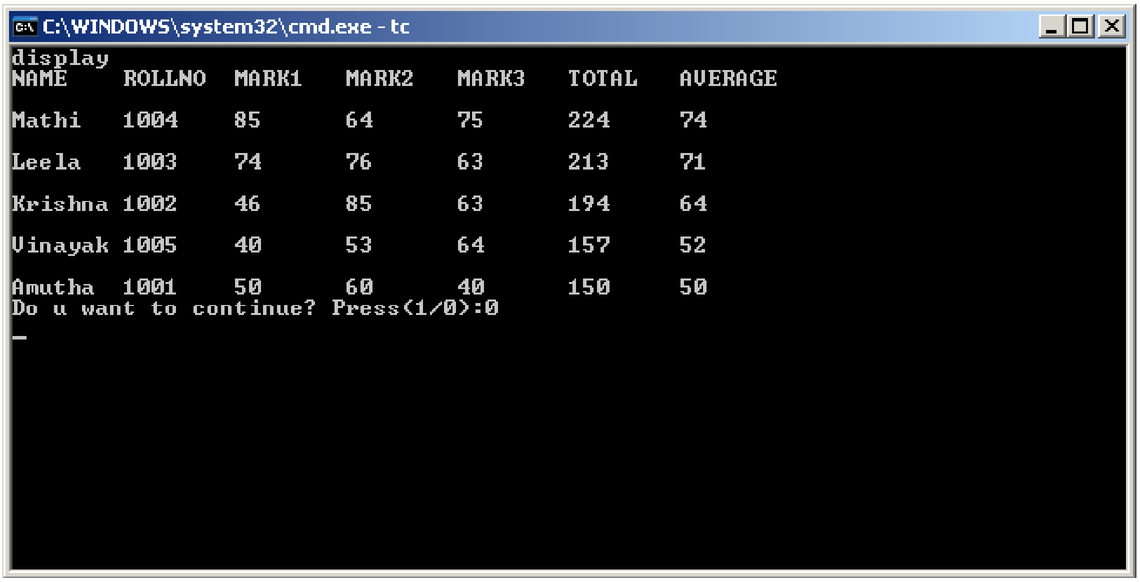
```
#include<iostream.h>
#include<string.h>
#include<conio.h>
void create(int);
void display(int);
void cal(int);

struct student
{
char name[10],res[5];
int sno,m1,m2,m3,tot;
float avg;
}s[10];
void main()
{
int n,ch,t;
clrscr();
cout<<“Enter the no of students”;
cin>>n;

do
{
cout<<“\n\t1.CREATE\n\t2.DISPLAY\nEnter ur choice:\n”;
cin>>ch;
switch(ch)
{
case 1:
create(n);
break;
case 2:
```

```
display(n);
break;
default:
cout<<“WRONG CHOICE”;
break;
}
cout<<“Do u want to continue? Press(1/0):”;
cin>>t;
}while(t!=0);
}
void create(int n)
{
cout<<“Create\n”;
for(int i=0;i<n;i++)
{
cout<<“Enter Student Name:”;
cin>>s[i].name;
cout<<“Enter Student RollNo:”;
cin>>s[i].sno;
cout<<“Enter Mark1:”;
cin>>s[i].m1;
cout<<“Enter Mark2:”;
cin>>s[i].m2;
cout<<“Enter Mark3:”;
cin>>s[i].m3;
}
}
void display(int n)
{clrscr();
cal(n);
cout<<“display”;
cout<<“\nNAME\tROLLNO\tMARK1\tMARK2\tMARK3\tTOTAL\tAVERAGE\n”;
for(int i=0;i<n;i++)
cout<<“\n”<<s[i].name<<“\t”<<s[i].sno<<“\t”<<s[i].m1<<“\t”<<s[i].m2<<“\t”<<s[i].m3<<“\t”<<s[i].tot<<“\t”<<s[i].avg<<“\n”;
}
void cal(int n)
{
for(int i=0;i<n;i++)
{
```

```
s[i].tot=s[i].m1+s[i].m2+s[i].m3;
s[i].avg=s[i].tot/3;
}
for(i=0;i<n;i++)
{
for(int j=i;j<n;j++)
{
if(s[i].tot<=s[j+1].tot)
{
char temp7[10];
int temp1,temp2,temp3,temp4,temp5;
float temp6;
temp5=s[i].tot; s[i].tot=s[j+1].tot; s[j+1].tot=temp5;
strcpy(temp7,s[i].name);strcpy(s[i].name,s[j+1].name);strcpy(s[j+1].name,temp7);
temp1=s[i].sno; s[i].sno=s[j+1].sno; s[j+1].sno=temp1;
temp2=s[i].m1; s[i].m1=s[j+1].m1; s[j+1].m1=temp2;
temp3=s[i].m2; s[i].m2=s[j+1].m2; s[j+1].m2=temp3;
temp4=s[i].m3; s[i].m3=s[j+1].m3; s[j+1].m3=temp4;
temp6=s[i].avg; s[i].avg=s[j+1].avg; s[j+1].avg=temp6;
}
}
}
}
```



```
Enter the no of students5

        1.CREATE
        2.DISPLAY
Enter ur choice:
1
Create
Enter Student Name:Amutha
Enter Student RollNo:1001
Enter Mark1:50
Enter Mark2:60
Enter Mark3:40
Enter Student Name:Krishna
Enter Student RollNo:1002
Enter Mark1:46
Enter Mark2:85
Enter Mark3:63
Enter Student Name:Leela
Enter Student RollNo:1003
Enter Mark1:74
Enter Mark2:76
Enter Mark3:63
```



```
display
NAME    ROLLNO   MARK1    MARK2    MARK3    TOTAL    AVERAGE

Mathi   1004     85       64       75       224      74

Leela   1003     74       76       63       213      71

Krishna 1002     46       85       63       194      64

Vinayak 1005     40       53       64       157      52

Amutha  1001     50       60       40       150      50
Do u want to continue? Press(1/0):0
_
```

b. Write a c++ program for create a inventory billing system using Multiple array.

```
#include<iostream.h>
#include<conio.h>
struct inventory
{
 int pid,qty;
 char pname[10];
 float price,amt;
 }s[5][10];
 void Line()
 {
 int L;
  for(L=0;L<=70;L++)
 cout<<"-";
 return;
 }
 void main()
{
 int i,j,k,flr,n=0,m=0,o;float tot[10]={0.0},cashtotal=0.0;
  clrscr();
  cout<<"enter the number of sales counter";
 cin>>m;
 for(i=1;i<=m;i++)
 {
 cout<<“\n\tcounter”<<i;
cout<<“\n\Enter the no of items”;
 cin>>n;
cout<<“\n ID:\t name:\t qty:\tUnitPrice:\n”;
 for(j=1;j<=n;j++)
 {
 cin>>s[i][j].pid;
 cin>>s[i][j].pname;
 cin>>s[i][j].qty;
 cin>>s[i][j].price;
s[i][j].amt=s[i][j].qty*s[i][j].price;
tot[i]=tot[i]+s[i][j].amt;
 }
  }
 for(i=1;i<=m;i++)
 cashtotal+=tot[i];
Line();
 cout<<“\n\n”;
 cout<<“counter\tProduct\tName\tQty\t\tPrice\t\tAmount\n”;
 cout<<“  id   \t id    \t\t       Rs.P \t         Rs.P.\n”;
 Line();
 for(i=1;i<=m;i++){
 cout<<“\n”<<i;
 for(j=1;j<=n;j++){
 cout<<“\t”<<s[i][j].pid<<“\t”;
 cout<<s[i][j].pname<<“\t”;
 cout<<s[i][j].qty<<“\t\t”;
 cout<<s[i][j].price<<“\t\t”;
 cout<<s[i][j].amt<<“\n”;
 }
Line();
 cout<<“\n\t\t\t\t\t”<<“Total Amount:”<<tot[i];
 cout<<“\n\n”;
 Line();
 }
 cout<<“\n\t\t\t”<<“Total Sales Amount:”<<cashtotal;
 getch();
 }
```

Output

```
enter the number of sales counter2
                              counter1
                         Enter the no of items3

              ID:    name:         qty:                  UnitPrice:
              1      Lux           3          50
              2      Power         2          45
              3      Liril         4          50

                              counter2
                         Enter the no of items3

              ID:    name:         qty:   UnitPrice:
              1      Maa           3          25
              2      Slice         4          30
              3      Pepsi         2          35

——————————————————————————————————————————————————————————————————————
counter  Product     Name          Qty        Price                Amount
  id        id                            Rs.      P            Rs.       P.
——————————————————————————————————————————————————————————————————————
 1          1        Lux           3          50                         150
            2        Power         2          45                          90
            3        Liril         4          50                         200
——————————————————————————————————————————————————————————————————————
                     Total Amount:440
——————————————————————————————————————————————————————————————————————
 2          1        Maa           3          25                          75
            2        Slice         4          30                         120
            3        Pepsi         2          35                          70
——————————————————————————————————————————————————————————————————————
                     Total Amount:265
——————————————————————————————————————————————————————————————————————
             Total Sales Amount:705
```

Lesson 3

POLYNOMIAL OBJECT

a. Write a C++ program to implement the Polynomial Object using overloaded operator

```
#include <iostream.h>
#include <time.h>
#include <stdlib.h>
#include <conio.h>
void line(int lines);

class Polynomial
{
        private:
                int power;
                int* coeffs;
        public:
                Polynomial() {power=0;coeffs=new int[1];}
                Polynomial(int deg) {power=deg;coeffs=new int[deg+1];}
                Polynomial(const Polynomial& X);

                void GetCoeffs(istream& in);
                int Coeff(int deg);
                void Show(ostream& out);
                int Degree() {return power;}
                //operators
                friend Polynomial operator +(Polynomial& X, Polynomial& B);
                friend Polynomial operator -(Polynomial& X, Polynomial& B);
               };

void main()
{
int ch;
clrscr();
do
```

```
{
cout<<“\n\t========================”;
cout<<“\n\t1.addition of Polynomial\n\t2.subtraction of Polynomial\n\t3.Exit”;
cout<<“\n\t========================”;
count<< "Enter your choicein";
cin>>ch;
switch(ch)
{
case 1:
     {
    Polynomial X(3);
    Polynomial Y(3);
     clrscr();
    cout<<“Enter the value for first Poly\n”;
    X.GetCoeffs(cin);
    cout<<“Enter the value for Second Poly\n”;
    Y.GetCoeffs(cin);
    cout<<“\n\tshow the First Polynomial\n\t”;
    X.Show(cout);
    line(2);
    cout<<“\n\tshow the second Polynomial\n\t”;
    Y.Show(cout);
    line(2);
    Polynomial Z(3);
    Z=X+Y;
    cout<<“\n\tShow the Add Polynomial\t”;
    Z.Show(cout);
    break;
        }
    case 2:
    {
    Polynomial X(3);
    Polynomial Y(3);
     clrscr();
    cout<<“Enter the value for first Poly\n”;
```

```
          X.GetCoeffs(cin);
          cout<<“Enter the value for Second Poly\n”;
          Y.GetCoeffs(cin);
          cout<<“\n\tshow the First Polynomial\n\t”;
          X.Show(cout);
          line(2);
          cout<<“\n\tshow the second Polynomial\n\t”;
          Y.Show(cout);
          line(2);
          Polynomial Z(3);
          Z=X-Y;
          cout<<“\n\tview the Subtraction of two Polynomial\t”;
          Z.Show(cout);
          break;
          }
 case 3:
 default:
 exit(0);

}
}while(ch!=0);
}

Polynomial operator+(Polynomial& X, Polynomial& Y)
                                        {
          Polynomial Z;
          if (X.power==Y.power)
          {
                    Z=X;
                    for (int i=Y.power; i>=0; i—)
                    {
                                   Z.coeffs[i]=X.coeffs[i]+Y.coeffs[i];
                    }
                    return Z;
          }
```

```
        else if(X.power<Y.power)
        {
                Z=Y;
                for (int i=X.power; i>=0; i—)
                {
                            Z.coeffs[i]=Y.coeffs[i];
                }
                return Z;
        }
               else if(X.power>Y.power)
        {
                Z=X;
                for (int i=X.power; i>=0; i—)
                {
                            Z.coeffs[i]=X.coeffs[i];
                }
               return Z;
}
return 0;
}
Polynomial operator-(Polynomial& X, Polynomial& Y)
                              {
        Polynomial Z;
        if (X.power==Y.power)
        {
                Z=X;
                for (int i=Y.power; i>=0; i—)
                {
                            Z.coeffs[i]=X.coeffs[i]-Y.coeffs[i];
                }
                return Z;
        }
        else if(X.power<Y.power)
        {
                Z=Y;
        for (int i=X.power; i>=0; i—)
        {
               Z.coeffs[i]=Y.coeffs[i];
        }
        return Z;
    }
        else if(X.power>Y.power)
    {
        Z=X;
        for (int i=X.power; i>=0; i—)
        {
               Z.coeffs[i]=X.coeffs[i];
        }
        return Z;
}
return 0;
}
int Polynomial::Coeff(int deg)
{

    return coeffs[deg];
}

void line(int lines)
{
    for (int i=0; i<lines; i++)
        cout << endl;
}

void Polynomial::GetCoeffs(istream& in)
{
    for (int i=power; i>=0; i—)
    {
        in >> coeffs[i];
    }
    in.ignore();
}

void Polynomial::Show(ostream& out)
{
    for (int i=power; i>=0; i—)
    {
        if (coeffs[i]>=0)
        {
               if (i!=power)
                out << “ + “;
               out << coeffs[i];

        }
        else
        {
               if (coeffs[i]<0)
                out << “ - “;
                out << 0-coeffs[i];
        }
        if (i>1)
               out << “x^” << i;
        else if (i==1)
               out << “x”;

    }
}

Polynomial::Polynomial(const Polynomial& X)
{
    coeffs=new int[X.power+1];
    power=X.power;
    for (int i=X.power; i>=0; i—)
    {
        coeffs[i]=X.coeffs[i];
        }

}
```

Output

```
Enter the value for first Poly
3
3
3
3
Enter the value for Second Poly
1
1
1
1

     show the First Polynomial
     3x^3 + 3x^2 + 3x + 3

     show the second Polynomial
     1x^3 + 1x^2 + 1x + 1



     Show the Add Polynomial 4x^3 + 4x^2 + 4x + 4
       ========================
     1.addition of Polynomial
     2.subtraction of Polynomial
     3.Exit
       ========================
      Enter your choice 2
Enter the value for first Poly
4
4
4
4
```

```
Enter the value for Second Poly
2
2
2
2

     show the First Polynomial
    4x^3 + 4x^2 + 4x + 4

     show the second Polynomial
    2x^3 + 2x^2 + 2x + 2


    view the Subtraction of two Polynomial  2x^3 + 2x^2 + 2x + 2
      ========================
    1.addition of Polynomial
    2.subtraction of Polynomial
     3.Exit
      ========================
      Enter your choice 3
```

b. Pointer Method:

```
//      ADDITION & MULTIPLICATION OF TWO POLYNOMIAL
        #include<iostream.h>
        #include<conio.h>
        #include<alloc.h>
        #define NEXT(poly)poly->nxtpoly
        typedef struct POLY
        {
        int coef,power;
        struct POLY *nxtpoly;
        }POLY;
        POLY *insertpoly(int coef, int power,POLY *first)
        {
        POLY *NEW,*current,*prod;
```

```
NEW=(POLY*)malloc(sizeof(POLY));
if(!NEW)
{
cout<<“Error:out of memory!”;
return(first);
}
NEW->coef=coef;
NEW->power=power;
NEW->nxtpoly=NULL;
if(!first)
return(NEW);
prod=first;
for(current=first;current;current=NEXT(current))
prod=(POLY*)current;
NEXT(prod)=NEW;
return(first);
}
POLY *createpoly()
{
int coef,power;
POLY*poly=NULL;
cout<<“\n Enter the coeff. power<coef.power>0->end:”;
while(1)
{
cin>>coef>>power;
cout<<coef<<“x^”<<power;
if(coef==0)
break;
else
poly=insertpoly(coef,power,poly);
if(power==0)
break;
}
return(poly);
```

```
}
POLY*freepoly(POLY *poly)
{
POLY *current;
for(current=poly;current;current=NEXT(current))
free(current);
return(NULL);
}
void displaypoly(POLY *poly)
{
POLY *current;
for(current=poly;current;current=NEXT(current))
if(current->coef!=0)
cout<<current->coef<<“X^”<<current->power<<“+”;
cout<<“=0”;
cout<<“\n”;
}
POLY *polyadd(POLY *poly1,POLY *poly2)
{
POLY *p1,*p2,*poly=NULL;
p1=poly1;
p2=poly2;
while(p1&&p2)
{
if(p1->power>p2->power)
{
poly=insertpoly(p1->coef,p1->power,poly);
p1=NEXT(p1);
}
else
if(p2->power>p1->power)
{
poly=insertpoly(p2->coef,p2->power,poly);
p2=NEXT(p2);
}
else
{
poly=insertpoly(p1->coef+p2->coef,p1->power,poly);
p1=NEXT(p1);
p2=NEXT(p2);
}
}
while(p1)
{
poly=insertpoly(p1->coef,p1->power,poly);
p1=NEXT(p1);
}
while(p2)
{
poly=insertpoly(p2->coef,p2->power,poly);
p2=NEXT(p2);
}
return(poly);
}
POLY *polymul(POLY *poly1, POLY *poly2)
{
POLY *p1,*pnew,*pmul,*prod;
prod=NULL;
for(;poly2;poly2=NEXT(poly2))
{
for(p1=poly1,pmul=NULL;p1;p1=NEXT(p1))
pmul=insertpoly(p1->coef*poly2->coef,p1->
        power+poly2->power,pmul);
pnew=polyadd(prod,pmul);
prod=freepoly(prod);
pmul=freepoly(pmul);
prod=pnew;
}
return(prod);
}
```

```
int main()
{
int choice;
clrscr();
POLY *poly1,*poly2,*poly;
poly=poly1=poly2=NULL;
cout<<“Polynomial manipulation program:\n\n”;
cout<<“Enter the first polynomial:\n”;
poly1=createpoly();
cout<<“\npoly1\t”;
displaypoly(poly1);
cout<<“Enter the second polynomial:\n”;
poly2=createpoly();
cout<<“\npoly2\t”;
displaypoly(poly2);
while(1)
{
cout<<“\nSelect polynomial operations\n1.ADD\n
            2.MUL\n3.QUIT”;
cout<<“\n Enter the choice:\t”;
cin>>choice;
cout<<“Entered the choice:\t”<<choice;
switch(choice)
{
case 1:
poly=polyadd(poly1,poly2);
break;
case 2:
poly=polymul(poly1,poly2);
break;
case 3:
default:
return(0);
}
cout<<“\nresult:”;
displaypoly(poly);
poly=freepoly(poly);
}
}
```

OUTPUT

```
    Polynomial manipulation program:
    Enter the first polynomial:
    Enter the coeff. power<coef.power>0->end:3 1 5 0
    poly1  3X^1+5X^0+=0
    Enter the second polynomial:
    Enter the coeff. power<coef.power>0->end:4 1 15 0
    poly2  4X^1+15X^0+=0

    Select polynomial operations
    1.ADD
    2.MUL
    3.QUIT

    Enter the choice:      1
    Entered the choice:     1
    result:7X^1+20X^0+=0
    Select polynomial operations
    1.ADD
    2.MUL
    3.QUIT
    Enter the choice:      2
    Entered the choice:     2
    result:12X^2+65X^1+75X^0+=0
    Select polynomial operations
1.ADD
2.MUL
3.QUIT
Enter the choice:      3
Entered the choice:     3
```

Lesson 4

LINKED LIST

a. Write a C++ programming to implementation the single linked lists.

```
#include<iostream.h>
#include<conio.h>
#include<alloc.h>
#include<stdlib.h>
struct link
{
 int info;
 struct link *next;
};
class Slink
{
 private:
  int i,number;
  link start,*previous,*new1;
 public:
  void insertion(link *);
  void create(link *);
  void display(link *);
  void delet(link *);
};
void Slink::create(link *node)
{
 start.next=NULL;
 node=&start;
 i=0;
 cout<<"\nInput Choice n for break:";
 char ch=getche();
 while(ch!='n')
 {
  node->next=(struct link *)malloc(sizeof(struct link));
  node=node->next;
  cout<<“\nInput the node:”<<(i+1)<<“:”;
  cin>>node->info;
  node->next=NULL;
  cout<<“\n Input Choice n For Break:”;
  ch=getche();
  i++;
 }
}
void Slink::insertion(link *node)
{
 node=start.next;
 previous=&start;
 new1=(struct link*)malloc(sizeof(struct link));
 new1->next=node;
 previous->next=new1;
 cout<<“\nInput the first node value:”;
 cin>>new1->info;
}
void Slink::display(link *node)
{
 node=start.next;
 cout<<“\nAfter inserting a node list is as follows:\n”;
 cout<<“Address\t\tValue”;
 while(node)
 {
  cout<<“\n”<<node;
  cout<<“”<<node->info;
  cout<<“\t”<<node->info;
  node=node->next;
 }
}
void Slink::delet(link *node)
{
node=start.next;
```

```
previous=&start;
if(node==NULL)
 cout<<“\nUnder Flow\n”;
else
 {
  previous->next=node->next;
cout<<node->info<<“is deleted”;
  free(node);
 }
}

void main()
{
char choice;
clrscr();
Slink S;
link *node=(link *)malloc(sizeof(link));
do
{
cout<<“\nCreate\tInsertion\tDisplay\tdElete\teXit\n”;
cout<<“\nSelect UR Choice:”;
choice=getch();
cout<<choice;
switch(choice)
{
case ‘C’:
S.create(node);
break;
case ‘I’:
S.insertion(node);
break;
case ‘D’:
S.display(node);
break;
case ‘E’:
```

```
S.delet(node);
break;
case ‘X’:
exit(0);
cout<<“\n”;
break;
}
}while(choice!=’X’);
getch();
}
```

Output

```
Create  Insertion      Display dElete  eXit

Select UR Choice:C
Input Choice n for break:
Input the node:1:2

 Input Choice n For Break:3
Input the node:2:3

 Input Choice n For Break:4
Input the node:3:4

 Input Choice n For Break:n
Create  Insertion      Display dElete  eXit

Select UR Choice:D
After inserting a node list is as follows:
Address        Value
0x8fde0fb82    2
0x8fde0fc03    3
0x8fde0fc84    4
Create  Insertion      Display dElete  eXit
```

```
Select UR Choice:I
Input the first node value:1

Create  Insertion      Display dElete  eXit

Select UR Choice:D
After inserting a node list is as follows:
Address        Value
0x8fde0fd01    1
0x8fde0fb82    2
0x8fde0fc03    3
0x8fde0fc84    4
Create  Insertion      Display dElete  eXit

Select UR Choice:E
Create  Insertion      Display dElete  eXit

Select UR Choice:D
After inserting a node list is as follows:
Address        Value
0x8fde0fb82    2
0x8fde0fc03    3
0x8fde0fc84    4
Create  Insertion      Display dElete  eXit

Select UR Choice:X
```

b. Write a C++ programming to Insert & Delete a desired node from Single Linked List.

Source code:

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

```
#include<stdlib.h>
struct link
{
 int info;
 struct link *next;
};
class Slink
{
 private:
  int i,number;
  link start,*previous,*new1;
 public:
  void insertion(link *);
  void create(link *);
  void display(link *);
  void delet(link *);
};
void Slink::create(link *node)
{
 start.next=NULL;
 node=&start;
 i=0;
 cout<<“\nInput Choice n for break:”;
 char ch=getche();
 while(ch!=’n’)
 {
  node->next=(struct link *)malloc(sizeof(struct link));
  node=node->next;
  cout<<“\nInput the node:”<<(i+1)<<“:”;
  cin>>node->info;
  node->next=NULL;
  cout<<"\n Input Choice n For Break:";
  ch=getche();
  i++;
 }
}
void Slink::insertion(link *node)
{
 node=start.next;
 previous=&start;
 int node_number=0;
 int insert_node;
 cout<<"\n\tInput node number you want to Insert";
 cin>>insert_node;
 while(node)
 {
 if((node_number+1)==insert_node)
 {
 new1=(struct link*)malloc(sizeof(struct link));
 new1->next=node;
 previous->next=new1;
 cout<<"\nInput the node value:";
 cin>>new1->info;
 break;
 }
 else
 {
 node=node->next;
 previous=previous->next;
 }
 node_number++;
   }
 }
void Slink::display(link *node)
{
 node=start.next;
 cout<<“\n Node list is as follows:\n”;
 while(node)
 {
  cout<<“\t”<<node->info;
  node=node->next;
 }
}
void Slink::delet(link *node)
{
node=start.next;
previous=&start;
int node_number=1;
int delete_node;
cout<<“\n Input Information of a node you want to delete”;
cin>>delete_node;
while(node)
{
if(node->info==delete_node)
{
cout<<“\n\tPosition of the Information in the list”<<node_number;
previous->next=node->next;
delete(node);
break;
}
```

```
else
{
node=node->next;
previous=previous->next;
}
}
 }

void main()
{
int choice;
clrscr();
Slink S;
link *node=(link *)malloc(sizeof(link));
do
{
cout<<"\n1:Create\t2:Insertion\t3:Display\t4:Delete\t5:eXit\n";
cout<<"\nSelect UR Choice:";
cin>>choice;
switch(choice)
{
case 1:
S.create(node);
break;
case 2:
S.insertion(node);
break;
case 3:
S.display(node);
break;
case 4:
S.delet(node);
break;
case 5:
exit(0);
cout<<"\n";
break;
}
}while(choice!=5);
getch();
}
```

Output

```
1:Create      2:Insertion    3:Display      4:Delete      5:eXit

Select UR Choice:1

Input Choice n for break:
Input the node:1:10

Input Choice n For Break:
Input the node:2:20

Input Choice n For Break:n
1:Create      2:Insertion    3:Display      4:Delete      5:eXit

Select UR Choice:3

Node list is as follows:
     10     20
1:Create      2:Insertion    3:Display      4:Delete      5:eXit
Select UR Choice:2
```

```
    Input node number you want to Insert2

Input the node value:15

1:Create     2:Insertion   3:Display    4:Delete     5:eXit

Select UR Choice:2

    Input node number you want to Insert3

Input the node value:25

1:Create     2:Insertion   3:Display    4:Delete     5:eXit

Select UR Choice:3

Node list is as follows:
     10     15     25     20
1:Create     2:Insertion   3:Display    4:Delete     5:eXit


Select UR Choice:4


 Input Information of a node you want to delete25


    Position of the Information in the list1
1:Create     2:Insertion   3:Display    4:Delete     5:eXit


Select UR Choice:3


Node list is as follows:
     10     15     20
1:Create     2:Insertion   3:Display    4:Delete     5:eXit


Select UR Choice:5
```

c. Write a C++ programming to implementation of Circular Linked Lists

```
#include<iostream.h>
#include<conio.h>
#include<malloc.h>
struct node
{
int info;
struct node *next;
};

class CircularLinkedList
{
struct node *parent,*endnode;
public :
void create();
void insert();
void delet();
void view();
};
void CircularLinkedList::create()
{
struct node *pos,*n;
int value,s;
parent=pos=NULL;
s=sizeof(struct node);
cout<<“\n enter -99 to stop”;
cin>>value;
while(value!=-99)
{
n=(struct node *)malloc(s);
n->info=value;
n->next=NULL;
if(parent==NULL)
parent=n;
```

```
else
pos->next=n;
pos=n;
cout<<“\n enter -99 to stop”;
cin>>value;
}
endnode=n;
endnode->next=parent;
}

void CircularLinkedList::view()
{
struct node *start=parent;
cout<<“\n parent->”;
do
{
cout<<start->info<<“->”;
start=start->next;
}while(start!=parent);
cout<<“parent”;
}
void CircularLinkedList::insert()
{
struct node *temp,*start=parent;
int value,pos,i=2;
cout<<“\n enter the value & position to insert :”;
cin>>value>>pos;
temp=(struct node *)malloc(sizeof(struct node));
temp->info=value;
temp->next=NULL;
if(pos==1)
{
temp->next=parent;
endnode->next=temp;
```

```
parent=temp;
}
else
{
while(start->next!=parent && i<pos)
{
start=start->next;
i++;
}
temp->next=start->next;
start->next=temp;
if(start==endnode)
endnode=temp;
}}
void CircularLinkedList::delet()
{
struct node *start=parent;
int pos,i=2;
cout<<"\n enter the position to delete :";
cin>>pos;
if(pos==1)
{
parent=parent->next;
endnode->next=parent;
}
else
{
while(start->next!=NULL && i<pos)
{
        start=start->next;
        i++;
        }
        if(start->next==endnode)
        {
```

```
start->next=parent;
endnode=start;
}
else
{
start->next=start->next->next;
}}
}

void main()
{
clrscr();
cout<<"\n\t\t OUT PUT\n";
CircularLinkedList CLL;
CLL.create();
CLL.view();
CLL.insert();
CLL.view();
CLL.delet();
CLL.view();
getch();
}
```

Output

```
          OUT PUT

enter -99 to stop90

enter -99 to stop91

enter -99 to stop92

enter -99 to stop93

enter -99 to stop-99

parent->90->91->92->93->parent
```

```
 enter the value & position to insert :95
3

 parent->90->91->95->92->93->parent
 enter the position to delete :4

 parent->90->91->95->93->parent
```

d. Write a C++ Program for Creation, Insertion and Deletion in Doubly linked list method

```
#include<iostream.h>
        #include<conio.h>
        #include<malloc.h>
        struct node
        {
        int info;
        struct node *next,*back;
        };

        class list
        {
        struct node *root,*end;
        public :
        void createinfo();
        void insert();
        void delet();
        void display();
        };
        void list::createinfo()
        {
        struct node *p,*n;
        int t, s;

        root=p=NULL;
```

```
s=sizeof(struct node);
cout<<“\n enter -999 to stop”;
cin>>t;
while(t!=-999)
{
n=(struct node *)malloc(s);
n->info=t;
n->next=NULL;
n->back=NULL;
if(root==NULL)
root=n;
else
{
p->next=n;
n->back=p;
}

p=n;
cout<<“\n enter -999 to stop”;
cin>>t;
}
end=n;
}

void list::display()
{
struct node *x=root;
cout<<“\n start->”;
while(x!=NULL)
{
cout<<x->info<<“->”;
x=x->next;
```

```
}
cout<<“end”;
x=end;
cout<<“\n back”;
while(x!=NULL)
{
cout<<x->info<<“->”;
x=x->back;
}
cout<<“end”;
}

void list::insert()
{
struct node *temp,*ex=root;
int value,pos,i=2;
cout<<“\n enter the value & position to insert :”;
cin>>value>>pos;
temp=(struct node *)malloc(sizeof(struct node));
temp->info=value;
temp->next=NULL;
temp->back=NULL;
if(pos==1)
{
temp->next=root;
root->back=temp;
root=temp;
}
else
{
while(ex->next!=NULL && i<pos)
{
ex=ex->next;
```

```
i++;
}
temp->next=ex->next;
temp->back=ex;
if(ex->next!=NULL)
ex->next->back=temp;
ex->next=temp;
if(temp->next==NULL)
end=temp;
}}

void list::delet()
{
struct node *ex=root;
int p,i=2;
cout<<"\n enter the position to delete :";
cin>>p;
if(p==1)
{
root=root->next;
root->back=NULL;
}
else
{
while(ex->next!=NULL && i<p)
{
ex=ex->next;
i++;
}
if(ex->next->next!=NULL)
{
ex->next=ex->next->next;
ex->next->back=ex;
```

```
        }
        else
        {
        ex->next=NULL;
        end=ex;
        }}}

void main()
        {
        clrscr();
        cout<<“\n\t\t OUT PUT\n”;
        list one;
        one.createinfo();
        one.display();
        one.insert();
        one.display();
        one.delet();
        one.display();
        getch();
        }
```

OUT PUT

```
enter -999 to stop1

enter -999 to stop2

enter -999 to stop3

enter -999 to stop4

enter -999 to stop-999

start->1->2->3->4->end
back4->3->2->1->end
```

```
enter the value & position to insert :100 3

 start->1->2->100->3->4->end
back4->3->100->2->1->end
enter the position to delete :2

 start->1->100->3->4->end
back4->3->100->1->end
```

e. Write a C++ programming to implement sorting techniques using General lists.

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

template <class Etype>
class Sorting
{
 Etype *Array;
 int Size;
 public:
        void GetData();
        void Display();
        void Swap(Etype &X,Etype &Y);
        void Bubble();
        void Insertion();
        void Selection();
        void QuickCall();
        void Quick(int,int);
};

template <class Etype>
void Sorting<Etype>::GetData()
{
```

```
cout<<endl<<“Enter the size of the array: “;
cin>>Size;
 if(Size<=0)
 {
  cout<<“Enter valid Size”<<endl;
  GetData();
 }
 else
 {
  Array=new Etype[Size];
  cout<<“Enter the values: “;
  for(int i=0;i<Size;i++)
          cin>>Array[i];
 }
}

template <class Etype>
void Sorting<Etype>::Display()
{
for(int i=0;i<Size;i++)
  cout<<setw(5)<<Array[i];
cout<<endl;
}
template <class Etype>
void Sorting<Etype>::Bubble()
{
 for(int i=0;i<Size-1;i++)
 {
          for(int j=i; j<Size;j++)
          {
           if(Array[i]>Array[j])
            Swap(Array[i],Array[j]);
          }
 }
```

```
}


template <class Etype>
void Sorting<Etype>::Swap(Etype& x,Etype &y)
{
 Etype temp;
 temp=x;
 x=y;
 y=temp;
}



template <class Etype>
void Sorting<Etype>::Insertion()
{
 for(int i=0;i<Size;i++)
 {
            Etype temp= Array[i];
     for(int j=i-1;j>=0 && Array[j]>temp; j—)
      Array[j+1]=Array[j];
     Array[j+1]=temp;
  }
}


template <class Etype>
void Sorting<Etype>::QuickCall()
{
  Quick(0,Size-1);
}

template <class Etype>
void Sorting<Etype>::Quick(int low,int high)
```

```
{
 if(low<high)
 {
  int i_ptr=low+1;
     int j_ptr=high;
     Etype x=Array[low];
  while(1)
  {
   while(Array[i_ptr]<x)
      i_ptr++;
   while(Array[j_ptr]>x)
      j_ptr—;
   if(i_ptr<j_ptr)
   {
      Swap(Array[i_ptr],Array[j_ptr]);
      i_ptr++;
      j_ptr++;
   }
   else
      break;
  }
  Array[low]=Array[j_ptr];
  Array[j_ptr]=x;
     Quick(low,j_ptr-1);
     Quick(j_ptr+1,high);
 }
}

template <class Etype>
void Sorting<Etype>::Selection()
{
 for(int i=Size-1;i>0;i—)
 {
 Etype max=Array[0];
 Etype index=0;
      for(int j=1;j<=i;j++)
  {
   if(Array[j]>max)
    {
            max=Array[j];
            index=j;
    }
  }
 Array[index]=Array[i];
 Array[i]=max;
 }
}

void main()
{
clrscr();
Sorting<int> S;
int choice;
do
{
clrscr();
cout<<“  SORTING METHODS “<<endl;
cout<<“1. Bubble Sort”<<endl;
cout<<“2. Insertion Sort”<<endl;
cout<<“3. Selection Sort”<<endl;
cout<<“4. Quick Sort”<<endl;
cout<<“  Place your choice ! : “;
cin>>choice;
switch(choice)
{
case 1:
cout<<endl<<“\t BUBBLE SORT”;
cout<<endl<<“\t ——— ——”;
```

```
S.GetData();
cout<<“Initial  list: “;
S.Display();
S.Bubble();
cout<<“Sorted   list: “;
S.Display();
break;
case 2:
cout<<endl<<“\t INSERTION SORT”;
cout<<endl<<“\t ———— ——”;
S.GetData();
cout<<“Initial  list: “;
S.Display();
S.Insertion();
cout<<“Sorted   list: “;
S.Display();
break;
case 3:
cout<<endl<<“\t SELECTION SORT”;
cout<<endl<<“\t ———— ——”;
S.GetData();
cout<<“Initial  list: “;
S.Display();
S.Selection();
cout<<“Sorted   list: “;
S.Display();
break;
case 4:
cout<<endl<<“\t QUICK SORT”;
cout<<endl<<“\t —— ——”;
S.GetData();
cout<<“Initial  list: “;
S.Display();
S.QuickCall();
```

```
cout<<“Sorted   list: “;
S.Display();
getch();
break;
}
getch();
}while(choice<=4);}
```

Output

```
  SORTING METHODS
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
  Place your choice ! : 1

          BUBBLE SORT
          ——— ——
Enter the size of the array: 10
Enter the values: 5 3 6 2 7 100 80 40 9 25
Initial  list:     5    3    6    2    7  100   80   40    9   25
Sorted   list:     2    3    5    6    7    9   25   40   80  100

  SORTING METHODS
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
  Place your choice ! : 2

          INSERTION SORT
          ———— ——
Enter the size of the array: 5
Enter the values: 2 1 5 10 45
```

```
Initial  list:     2    1    5   10   45
Sorted   list:     1    2    5   10   45

  SORTING METHODS
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
  Place your choice ! : 3

         SELECTION SORT
         ———————— ————
Enter the size of the array: 7
Enter the values: 4 2 5 6 1 8 9
Initial  list:     4    2    5    6    1    8    9
Sorted   list:     1    2    4    5    6    8    9


 SORTING METHODS
1. Bubble Sort
2. Insertion Sort
3. Selection Sort
4. Quick Sort
  Place your choice ! : 4

                                   QUICK SORT
                                   ——— ————
                         Enter the size of the array: 5
                         Enter the values:        6    3    1    2    7
                         Initial  list:           6    3    1    2    7
                         Sorted   list:           1    2    3    6    7
```

Lesson 5

IMPLEMENTATION OF STACK

a. Write a C++ programming to implementation of stack using arrays

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#define n 100
int top=-1;
int flag=0;
class  stacks
        {
private:char stack[100];
int str1;
public:
void push(char*,char);
int pop(char*);
void display(char*);
        };
//Definition of the push function
void stacks::push(char s[],char d)
{
if(top==(n-1))
flag=0;
else
        {       flag=1; ++top;
        s[top]=d;
        }
}
//Definition of the pop function
int stacks:: pop(char s[])
{
int pop;
if(top==-1)
```

```
{
pop=0;
flag=0;
}
else
{flag=1;
pop=s[top];
—top;
}
return(pop);
}
//Display Function
void stacks::display(char s[])
{
if(top==-1)
        {
cout<<“Stackis empty”;
        }
else
        {
for(int i=top;i>=0;—i)
cout<<“\n\t”<<s[i];
        }
}
void main()
{
clrscr();
stacks sarray;
char stack[n];
char data;
int choice;
int q=0;
int top=-1;
do{
cout<<“\nPush=1  Pop=2  Quit=3”;
cout<<“\n\t’Select Your choice from 1 2 3:’”;
 cin>>choice;
switch(choice)
{
case 1:
cout<<“\n Insert the character”;
cin>>data;
sarray.push(stack,data);
if(flag)
{
cout<<“\nAfter inserting”;
sarray.display(stack);
if(top==(n-1))
cout<<“\nStack is full”;
}
else
cout<<“\nStack overflow after pushing”;
break;
case 2: data=sarray.pop(stack);
if(flag)
        {
cout<<“\nData is popped”<<data;
cout<<“\nRest data in stack is as follows”;
sarray.display(stack);
        }
else
cout<<“\nStack underflow”;
break;
case 3: q=3;
}
        }while(!q);
}
```

Output

```
Push=1  Pop=2  Quit=3
```

```
        ‘Select Your choice from 1 2 3:’1

 Insert the charactera

After inserting
         a
Push=1  Pop=2  Quit=3
        ‘Select Your choice from 1 2 3:’1

 Insert the characterb

After inserting
        b
        a
Push=1  Pop=2  Quit=3
        ‘Select Your choice from 1 2 3:’1


 Insert the characterc


After inserting
        c
        b
        a
Push=1  Pop=2  Quit=3
        ‘Select Your choice from 1 2 3:’2


Data is poppedc
Rest data in stack is as follows
        b
        a
Push=1  Pop=2  Quit=3
        ‘Select Your choice from 1 2 3:’3
```

b. Write a C++ program to implement operations of a Stack using pointers

```
//          PUSH POP OPERATIONS OF STACK USING POINTERS


         #include<iostream.h>
```

```
#include<malloc.h>
#include<conio.h>
struct link
{
int info;
link *next;
};
class stack_link
{
private:link*start;
public:void display(link *);
link*push(link*);
link*pop(link*);
int main_menu();
};
void stack_link::display(link*rec)
{
while(rec!=NULL)
{
cout<<“\n”<<rec->info;
rec=rec->next;
}
}
link*stack_link::push(link*rec)
{
link*new_rec;
cout<<“\n Input the new value for next location of the stack:”;
new_rec=(link*)malloc(sizeof(link));
new_rec->next=rec;
cin>>new_rec->info;
new_rec->next=rec;
rec=new_rec;
return(rec);
}
link*stack_link::pop(link*rec)
```

```
{
link*temp;
if(rec==NULL)
{cout<<“\nStack is empty”;}
else
{
temp=rec->next;
free(rec);
rec=temp;
cout<<“\nAfter pop operation the stack is as follows:\n”;
display(rec);
if(rec==NULL)
cout<<“\n Stack is empty”;
}
return(rec);
}
int stack_link::main_menu()
{
int choice;
do
{
cout<<“\n 1<-push”;
cout<<“\n 2<-pop”;
cout<<“\m 3<-quit”;
cout<<“\n Input your choice:”;
cin>>choice;
if(choice<1||choice>3)
cout<<“\n incorrect choice-> try once again”;
}
while(choice<1||choice>3);
return(choice);
}
void main()
{
clrscr();
stack_link stack;
link*start;
int choice;
start=NULL;
do
{
choice=stack.main_menu();
switch(choice)
{
case 1:
start=stack.push(start);
cout<<"\n After push operation stack is as follows:";
stack.display(start);
break;
case 2:
start=stack.pop(start);
break;
default:cout<<"\n End of session";
}
}
while(choice!=3);
getch();
}
```

OUTPUT :

```
1<-push
2<-pop
3<-quit
Input your choice:1

Input the new value for next location of the stack:23

After push operation stack is as follows:m
23
```

```
1<-push
2<-pop
3<-quit
Input your choice:1

Input the new value for next location of the stack:45

After push operation stack is as follows:
45
23
1<-push
2<-pop
3<-quit
Input your choice:1

Input the new value for next location of the stack:56

After push operation stack is as follows:
56
45
23
1<-push
2<-pop
3<-quit
Input your choice:2

After pop operation the stack is as follows:

45
23
1<-push
2<-pop
3<-quit
Input your choice:3

End of session
```

Lesson 6

IMPLEMENTATION OF QUEUE

a) Implementation of Queue Using Arrays

```
#include<iostream.h>
#include<conio.h>
#include<string.h>
#include<ctype.h>
#include<process.h>
#define n 20
class quearray
{
public:
int rear,front;
char  ch;
char q[n];
public:
quearray()
{
rear=front=-1;
}
void insert();
void delqueue();
void display();
};
void quearray :: insert()
{
if((front==0)&&(rear==n-1))
{
cout<<“\n overflow”;
rear=1;
return;
}
else
```

```
if(front<0)
{
front=0;
rear=0;
cout<<“\n Insert the element”;
cin>>ch;
q[rear]=ch;
}
else
if(rear==n-1)
{
rear=0;
cout<<“\nInsert the element:”;
cin>>ch;
q[rear]=ch;
}
else
{
rear++;
cout<<“\nInsert the element:”;
cin>>ch;
q[rear]=ch;
}
}
void quearray :: delqueue()
{
if(front<0)
{
cout<<“\n underflow”;
return;
}
ch=q[front];
q[front]=NULL;
cout<<“Elementis Deleted:”<<ch;
if(front==rear)
```

```
{
front=-1;
rear=-1;
}
else
if(front==n-1)
{
front=0;
}
else
{
front++;
}
}
void quearray :: display()
{
if(front<0)
return;
if(rear>=front)
{
for(int i=front;i<=rear;i++)
{
cout<<q[i];
}
}
else
{
for(int i=front;i<=n;i++)
{
cout<<q[i];
}

for(i=0;i<=rear;i++)
{
cout<<q[i];
}
}
}
void main()
{
quearray Q;
int k=0;
int s;
clrscr();
do
{
cout<<“\n \t 1. INSERT\t 2. DELETE\t 3. QUIT\n”;
cout<<“\t Select the choice [1, 2, 3]:”;
cin>>s;
switch(s)
{
case 1:
Q.insert();
cout<<“\n queue after inserting:”;
Q.display();
break;
case 2:
Q.delqueue();
cout<<“\n queue after deleteion :”;
Q.display();
break;
case 3:
k=4;
}
}while(!k);
getch();
}
```

Output

```
        1. INSERT       2. DELETE       3. QUIT
```

```
          Select the choice [1, 2, 3]:1
 Insert the elementA
 queue after inserting:A
          1. INSERT       2. DELETE       3. QUIT
          Select the choice [1, 2, 3]:1

Insert the element:B

 queue after inserting:AB
          1. INSERT       2. DELETE       3. QUIT
          Select the choice [1, 2, 3]:1

Insert the element:C

 queue after inserting:ABC
          1. INSERT       2. DELETE       3. QUIT
          Select the choice [1, 2, 3]:2
Elementis Deleted:A
 queue after deleteion :BC
          1. INSERT       2. DELETE       3. QUIT
          Select the choice [1, 2, 3]:3
```

b. Write a C ++ program to implement operations of a queue using pointers

```
//Queue using pointers

#include<iostream.h>
#include<conio.h>
#include<malloc.h>
struct node
          {
int data;
node *link;
          };
struct queue
          {
node *first;
node *last;
node *link_next;
          };
struct Q
{
queue q;
node *qlink;
public:
void Initialise()
 {
q.first=NULL;
q.last=NULL;
 }
void insertqueue();
void deletequeue();
void display();
};
void Q::insertqueue()
{
 qlink=new(node);
cout<<“Enter the node”;
cin>>qlink->data;
cout<<“Insert the node :”<<qlink->data;
qlink->link=NULL;
if((q.last)==NULL)
q.first=qlink;
else
q.last->link=qlink;
q.last=qlink;
}
void Q::deletequeue()
         {
if(q.first==NULL)
         {
cout<<“\tQueue is Empty”;
q.last=NULL;
```

```
        }
else
        {
qlink=q.first;
cout<<“\tDelete the node:”<<q.first->data;
q.first=q.first->link;
free(qlink);
}
}
void Q:: display()
{
if(q.first==NULL)
cout<<“Queue is Empty”;
else
        {
cout<<“\nfirst”;
for(qlink=q.first;qlink!=NULL;qlink=qlink->link)
cout<<“==>”<<qlink->data;
cout<<“<==last\n”;
        }
}
void main()
{
char choice;
Q mainqueue;
clrscr();
mainqueue.Initialise();
cout<<“\t\tQUEUE USING POINTER”;
cout<<“\n\tInsert\tDelete\tView\tExit”;
do {
cout<<“\nEnter your choice:\t”;
cin>>choice;
switch(choice)
{
case ‘I’: mainqueue.insertqueue();
break;
case ‘D’: mainqueue.deletequeue();
break;
case ‘V’: mainqueue.display();
break;
case ‘E’:
break;
default:
cout<<“Invalid choice”;
}
}while(choice!=’E’);
getch();
}
```

Output:

```
            QUEUE USING POINTER
      Insert  Delete  View    Exit
Enter your choice:     I
Enter the node100
Insert the node :100
Enter your choice:     I
Enter the node200
Insert the node :200
Enter your choice:     I
Enter the node300
Insert the node :300
Enter your choice:     V

first==>100==>200==>300<==last

Enter your choice:     D
     Delete the node:100
Enter your choice:     V

first==>200==>300<==last

Enter your choice:     E
```

Lesson 7

CIRCULAR QUEUE

a. Write a C++ programming to the circular queue implementation.

```
#include<iostream.h>
#include<conio.h>
#include<process.h>
int j=0;

class circularqueue
{
int a[5],i,f,r,n;
public:
        void insert();
        void delet();
        void display();
circularqueue()
{
f=0;r=-1;n=5;a[i]=0;
}};

void main()
{
  clrscr();
 int choice;
 circularqueue obj;
 do
 {
 cout<<“\nenter the choice:”;
 cout<<“1.insert”<<endl;
 cout<<“2.delete”<<endl;
 cout<<“3.Display.\n”;
 cout<<“4.Exit.\n”;
  cin>>choice;

  switch(choice)
 {
```

```
case 1 :obj.insert(); break;
case 2 :obj.delet() ; break;
case 3 :obj.display(); break;
case 4 : exit(0); break;

default:
cout<<“Enter the correct choice.\n”;
} }while(choice<5);}

void circularqueue::insert()
{
int in;
if(j<5)
{
if(r==n-1 && f==0)
{
 cout<<“Queue is full.\n\n”;
}
else
{
cout<<“Enter the no=”;
cin>>in;
r=(r+1)%n;
a[r]=in;
j++;
}}
else
{
cout<<“\n QUEUE IF FULL “;
}}

void circularqueue::delet()
{
int temp;
        if(r<=-1)
        {
        cout<<“Queue is Empty.\n”;
```

```
          }
          else
          {
          temp=a[f];
          cout<<“The Deleted Value Is=”<<temp;
          a[f]=0;
                if(f!=r)
                 {
                 f=(f+1)%n;
                 j—;
                 }

else
          {
          f=0;r=-1;
          }} }

void circularqueue::display()
 {
   for(i=0;i<n;i++)
   cout<<a[i]<<“\t”;
  }
```

Output:

```
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=100

enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=200
```

```
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=300

enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=400

enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=500

enter the choice:1.insert
2.delete
3.Display.
4.Exit.
3
100     200     300     400     500
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
2
The Deleted Value Is=100
enter the choice:1.insert
2.delete
3.Display.
```

```
4.Exit.
2
The Deleted Value Is=200
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
3
0     0     300    400    500
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
3
350    0     300    400    500
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
1
Enter the no=450

enter the choice:1.insert
2.delete
3.Display.
4.Exit.
3
350    450    300    400    500
enter the choice:1.insert
2.delete
3.Display.
4.Exit.
4
```

Lesson 8

EVALUATIONS OF EXPRESSION

a. Write a program toconvert the Infix to Postfix expression

```
//      INFIX TO POSTFIX USING STACK

        #include<iostream.h>
        #include<conio.h>
        #include<ctype.h>
        #include<string.h>
        #include<math.h>
        class ex1
        {
        char str[50];
        public:
        void input();
        int strprt(char);
        int expprt(char);
        void convert();
        };
        void ex1::input()
        {
        cout<<“enter the Infix Expression end with $ sign:”;
        cin>>str;
        }
        int ex1::strprt(char c)
        {
        int pr;
        switch(c)
        {
        case ‘#’:
        pr=-1;
        break;
        case ‘(‘:
        case ‘)’:
```

```
pr=0;
break;
case ‘*’:
case ‘/’:
pr=2;
break;
case ‘+’:
case ‘-’:
pr=1;
break;
}
return(pr);
}
int ex1::expprt(char c)
{
int pr;
switch(c)
{
case ‘(‘:
pr=4;
break;
case ‘)’:
pr=0;
break;
case’*’:
case’/’:
pr=2;
break;
case ‘+’:
case ‘-’:
pr=1;
break;
}
return(pr);
}
```

```
void ex1::convert()
{
int i=0,top=0;
char stk[50], item;
while(str[i]!=’$’)
{
item=str[i];
if(isalpha(item))
cout<<item;
else
{
if(item==’)’)
{
while(stk[top]!=’(‘)
{
cout<<stk[top];
—top;
}
—top;
}
else
{
while((strprt(stk[top]))>expprt(item))
{
cout<<stk[top];
—top;
}
++top;
stk[top]=item;
}
}
i++;
}
while(top>=1)
{
cout<<stk[top];
—top;
}
}
void main()
{
ex1 ob;
clrscr();
ob.input();
ob.convert();
getch();
}
```

OUTPUT :

```
Enter the Infix Expression end with $ sign:(A+B)*(A-B)$

AB+AB-*
```

b. Write a program to convert the Infix to Prefix expression

```
#include <iostream.h>
#include <string.h>
#include <ctype.h>
#include<conio.h>
const int M = 50 ;
class inprefix
{
        private :
                char target[M], stack[M] ;
                char *s, *t ;
                int top, k ;
        public :
                inprefix( ) ;
                void expr ( char *str ) ;
                void push ( char c ) ;
                char pop( ) ;
                void convert( ) ;
                int priority ( char c ) ;
                void view( ) ;
} ;
inprefix :: inprefix( )
{
        top = -1 ;
        strcpy ( target, “” ) ;
        strcpy ( stack, “” ) ;
        k = 0 ;
}
void inprefix :: expr ( char *str )
{
        s = str ;
        strrev ( s ) ;
        k = strlen ( s ) ;
        * ( target + k ) = ‘\0’ ;
        t = target + ( k - 1 ) ;
}
void inprefix :: push ( char c )
{
        if ( top == M - 1 )
                cout << “\nStack is full\n” ;
        else
        {
                top++ ;
                stack[top] = c ;
        }
}
char inprefix :: pop( )
{
        if ( top == -1 )
        {
                cout << “Stack is empty\n” ;
                return -1 ;
        }
        else
        {
                char item = stack[top] ;
                top— ;
                return item ;
        }
}
void inprefix :: convert( )
{
        char opr ;

        while ( *s )
        {
                if ( *s == ‘ ‘ || *s == ‘\t’ )
                {
                        s++ ;
                        continue ;
                }

                if ( isdigit ( *s ) || isalpha ( *s ) )
                {
                        while ( isdigit ( *s ) || isalpha ( *s ) )
                        {
                                *t = *s ;
                                s++ ;
                                t— ;
                        }
                }

                if ( *s == ‘)’ )
                {
                        push ( *s ) ;
                        s++ ;
```

```
    }

    if ( *s == ‘*’ || *s == ‘+’ || *s == ‘/’ ||
                *s == ‘%’ || *s == ‘-’ || *s == ‘$’ )
    {
        if ( top != -1 )
        {
            opr = pop( ) ;

            while ( priority ( opr ) > priority ( *s ) )
            {
                *t = opr ;
                t— ;
                opr = pop( ) ;
            }
            push ( opr ) ;
            push ( *s ) ;
        }
        else
            push ( *s ) ;
        s++ ;
    }

    if ( *s == ‘(‘ )
    {
        opr = pop( ) ;
        while ( ( opr ) != ‘)’ )
        {
            *t = opr ;
            t— ;
            opr =  pop ( ) ;
        }
        s++ ;
    }
}
```

```
        while ( top != -1 )
        {
                opr = pop( ) ;
                *t = opr ;
                t— ;
        }
        t++ ;
}
int inprefix :: priority ( char c )
{
        if ( c == ‘$’ )
                return 3 ;
        if ( c == ‘*’ || c == ‘/’ || c == ‘%’ )
                return 2 ;
        else
        {
                if ( c == ‘+’ || c == ‘-’ )
                        return 1 ;
                else
                        return 0 ;
        }
}
void inprefix :: view( )
{
        while ( *t )
        {
                cout << “ “ << *t ;
                t++ ;
        }
}
```

```
void main( )
{
        char expr[M] ;
        inprefix pfix ;
clrscr();
        cout << “\nEnter an expression in infix form: “ ;
        cin.getline ( expr, M ) ;

        pfix.expr( expr ) ;
        pfix.convert( ) ;

        cout << “The Prefix expression is: “ ;
        pfix.view( ) ;
}
```

output

```
Enter an expression in infix form: (a+b)*(a-b)
The Prefix expression is:  * + a b - a b
```

C. Write a program to evaluate the given expressions.

Source code:

```
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<string.h>
#include<stdio.h>

#define M 30
#define symbol 10
```

```
#define str 20

typedef struct prepost
{
  int top;
  int s[M];
}Fix;

void init(Fix*);
void push(Fix*,int);
int pop(Fix*);
void eval(Fix*,char,int,int);
int gettype(char);
void main()
{
        char str1[M];
        int item1,item2,item,l,i,pr;
        Fix stk;
        fflush(stdin);
        int k;
         clrscr();
        do{
        cout<<“\n\t\tEvaluation of Expression.”;
        cout<<“\n\t1:PreFix\t2:PostFix3:Exit \n\t”;
        cin>>k;
        switch(k)
        {
        case 1:
```

```
        {
        init(&stk);
        cout<<“ ENTER THE PREFIX EXPRESSION “;
        gets(str1);
        l=strlen(str1);

        for(i=l;i>=0;i—)
        {
              if(str1[i]==’ ‘ || str1[i]==’\0')
              continue;
              switch(gettype(str1[i]))
              {
                   case symbol : item=str1[i]-’0';
                   push(&stk,item);
                   break;
                   case str : item1=pop(&stk);
                   item2=pop(&stk);
                   eval(&stk,str1[i],item1,item2);
              }
        }
cout<<“\n\tResult of prefix evaluation is:”;
 cout<<stk.s[0];
 getch();
 }
  break;
case 2:
        {
        init(&stk);
```

```
            cout<<“ ENTER THE Postfix EXPRESSION “;
            gets(str1);
            l=strlen(str1);

            for(i=0;i<=l;i++)
            {
                  if(str1[i]==’ ‘ || str1[i]==’\0')
                  continue;
                  switch(gettype(str1[i]))
                  {
                        case symbol : item=str1[i]-’0';
                        push(&stk,item);
                        break;
                        case str : item2=pop(&stk);
                        item1=pop(&stk);
                        eval(&stk,str1[i],item1,item2);
                  }
            }
cout<<“\n\tResult of postfix evaluation is:”;
 cout<<stk.s[0];
 getch();
 }
 break;
 case 3: k=3;
 default:cout<<“\n End”;
 }
 }while (k!=3);
}
```

```
void init(Fix *stk )
{
  stk->top=-1;
}

void push(Fix *st,int num)
{
         st->top++;
          st->s[st->top]=num;
}

int pop(Fix *st)
{
         int item;
          item=st->s[st->top];
         st->top—;
         return item;
}

void eval(Fix *st,char opr,int item1,int item2)
{
  int res;
           switch(opr)
          {
                 case ‘+’: res=item1+item2;
                 break;
                 case ‘-’: res=item1-item2;
```

```
            break;
            case ‘*’: res=item1*item2;
            break;
            case ‘/’: res=item1/item2;
            break;
            case ‘%’: res=item1%item2;
            break;
            case ‘^’: res=pow(item1,item2);
            break;
        }
  push(st,res);
}

int gettype(char c)
{
            switch(c)
            {
                  case ‘+’:
                  case ‘-’:
                  case ‘*’:
                  case ‘/’:
                  case ‘^’:
                  case ‘%’: return str;
                  default : return symbol;
             }
}
```

Output

```
          Evaluation of Expression.
     1:PreFix     2:PostFix3:Exit
     1
ENTER THE PREFIX EXPRESSION *+23-25


     Result of prefix evaluation is:-15
          Evaluation of Expression.
     1:PreFix     2:PostFix3:Exit
     2
ENTER THE Postfix EXPRESSION 23+25-*


     Result of postfix evaluation is:-15
          Evaluation of Expression.
     1:PreFix     2:PostFix3:Exit
     3

End
```

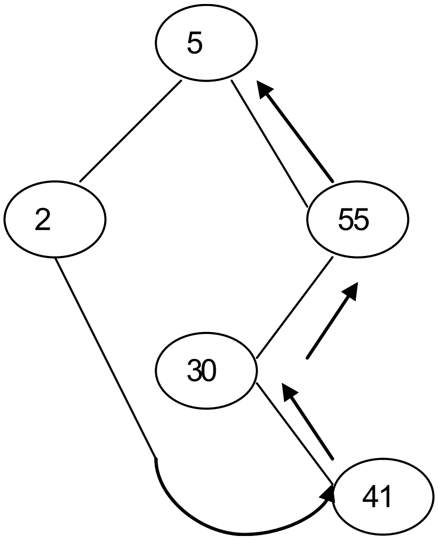
Lesson 9

TREE TRAVERSALS

a. To write a c++ program for Binary implementation and traversals using recursion.

Traverse the given tree using Inorder, Preorder and Postorder traversals.

Example : Postorder Traversals



Postorder : 2 41 30 55 5

```
#include<iostream.h>
#include<alloc.h>
#include<stdlib.h>
#include<conio.h>
typedef struct tree *node;
node create(int ,node NODE);
void inorder(node NODE);
void preorder(node NODE);
void postorder(node NODE);
struct tree
{
        int nodevalue;
        struct tree *right,*left;
```

```
}*root;

void main()
{
        node NODE= NULL;
        int nodevalue,choice,i=0,num;
        clrscr();
        cout<<"\t\t\t**** BINARY TREE TRAVERSAL****\nEnter the number of Nodes to
form a Tree:";
        cin>>num;
        cout<<"\n Enter the values for nodes\n";
        for(i=1;i<=num;i++)
        {
                cin>>nodevalue;
                NODE=create(nodevalue,NODE);

        }

cout<<"\n\t\t\t1.INORDER\n\t\t\t2.PREORDER\n\t\t\t3.POSTOTRDER\n\t\t\t4.EXIT\n";
        do
        {
                cout<<"\nEnter your choice:";
                cin>>choice;
                switch  (choice)
                {
                        case 1:
                        cout<<"Inorder Traversal\n";
                        inorder(NODE);
                        break;
                        case 2:
                        cout<<"Preoroder Traversal\n";
                        preorder(NODE);
                        break;
                        case 3:
```

```
                        cout<<“Postorder Traversal\n”;
                        postorder(NODE);
                        break;
                        default:
                        cout<<“Exit”;
                        exit(0);
                    }

    }
    while(choice!=4);
    getch();
    }
    node create(int X, node NODE)
    {

     struct tree *new1;
            new1=(tree*)malloc(sizeof(struct tree));
            if(new1==NULL)
            cout<<“No Nodes are here\n”;
            else
            {
                if(NODE==NULL)
                {
                        new1->nodevalue=X;
                        new1->left=NULL;
                        new1->right=NULL;
                        NODE=new1;
                }
                else
                {
                        if(X<NODE->nodevalue)
                        NODE->left=create(X,NODE->left);
                        else
                        NODE->right=create(X,NODE->right);
                }
            }
            return NODE;
    }
    void inorder(node NODE)
    {
            if(NODE!=NULL)
            {
                inorder(NODE->left);
                cout<<“\t”<<NODE->nodevalue;
                inorder(NODE->right);
            }
    }
    void preorder(node NODE)
    {
            if(NODE!=NULL)
            {
                cout<<“\t”<<NODE->nodevalue;
                preorder(NODE->left);
                preorder(NODE->right);
            }
    }
    void postorder(node NODE)
    {
            if(NODE!=NULL)
            {
                postorder(NODE->left);
                postorder(NODE->right);
                cout<<“\t”<<NODE->nodevalue;
            }
    }
```

OUTPUT

```
                    **** BINARY TREE TRAVERSAL****
Enter the number of Nodes to form a Tree:5

 Enter the values for nodes
5
55
2
30
41

                1.INORDER
                2.PREORDER
                3.POSTOTRDER
               4.EXIT

Enter your choice:1
Inorder Traversal
     2     5     30     41     55
Enter your choice:2
Preoroder Traversal
     5     2     55     30     41
Enter your choice:3
Postorder Traversal
     2     41     30     55     5
Enter your choice:4
```

b. Write a C++ program to implementation Binary Search Trees.

```
#include<iostream.h>
#include<conio.h>
enum boolean { false=0,true=1};

struct node
```

```
{
int ele;
node *left,*right;
}*root=NULL;

class BST
{
node *par,*temp,*temp1;
node* newnode(int);
public:
void insert(int,node*,int);
boolean search(int,node*);
node* deletemin(node**);
void del(int,node**);
void display(node*);
};

node* BST::newnode(int x)
{
node *nod=new node;
nod->ele=x;
nod->left=nod->right=NULL;
return(nod);
}

void BST::insert(int x,node *cur,int pos)
{
if(cur == NULL)
{
 cur=newnode(x);
 if(pos==1)
 par->left=cur;
 else if (pos==2)
 par->right=cur;
if(root==NULL)
root=cur;
}
else
        {
        par=cur;

        if(x < cur->ele)
                insert(x,cur->left,1);

        else if(x > cur->ele)
                insert(x,cur->right,2);
        }}

boolean BST::search(int x,node *cur)
{
if(root == NULL)
        return false;
else if(x == cur->ele)
        return true;
else if(x < cur->ele && cur->left != NULL)
        return (search(x,cur->left));
else if(x > cur->ele && cur->right != NULL)
        return (search(x,cur->right));
return false;
}
node* BST::deletemin(node **cur)
{
if((*cur)->left==NULL)
        {
        temp=(*cur);
        (*cur)=(*cur)->right;
        }
```

```
else
        temp=deletemin((&(*cur)->left));
return(temp);
}

void BST::del(int x,node **cur)
{
if((*cur)!=NULL)
        {
        if(x<(*cur)->ele)
                del(x,(&(*cur)->left));
        else if(x>(*cur)->ele)
                del(x,(&(*cur)->right));

        else    // x == (*cur)->ele
                {
                if( ((*cur)->left==NULL) && ((*cur)->right==NULL) )
                        {
                        delete((*cur));
                        (*cur)=NULL;    //must
                        }
                else if((*cur)->left==NULL)
                        (*cur)=(*cur)->right;

                else if((*cur)->right==NULL)
                        (*cur)=(*cur)->left;
                else
                        {
                        temp1=(*cur)->left;
                        (*cur)=deletemin((&(*cur)->right));
                        (*cur)->left=temp1;
                        }
                }
        }
```

```
}

void BST::display(node *cur)
{
if(cur != NULL)
        {
        display(cur->left);
        cout<<“\t”<<cur->ele;
        display(cur->right);
        }
}

void main()
{
int no,x,p;
BST bst;
clrscr();
do
{
        cout<<“\n1:Insert\t2:Delete\t3:Search\t4:Display\t5:Exit\nSelect your option :”;
        cin>>no;

        switch(no)
        {
                case 1:
                        cout<<“\nEnter the no. to be inserted:”;
                        cin>>x;
                        bst.insert(x,root,0);
                        break;
                case 2:
                        cout<<“\nEnter the element to be deleted : “;
                        cin>>x;
                        bst.del(x,&root);
                        break;
```

```
            case 3:
                    cout<<“\nEnter the element to be searched :”;
                    cin>>x;
                    p=bst.search(x,root);
                    if(p==true)
                            cout<<“The element is in the BST\n”;
                    else
                            cout<<“The element is not in the BST\n”;
                    break;
            case 4:
                    cout<<“\nThe elements in the list are\n”;
                    bst.display(root);
    }

}while(no<5);

}
```

Output:

```
1:Insert     2:Delete     3:Search     4:Display     5:Exit
Select your option :1
Enter the no. to be inserted:3

1:Insert     2:Delete     3:Search     4:Display     5:Exit
Select your option :1
Enter the no. to be inserted:4

1:Insert     2:Delete     3:Search     4:Display     5:Exit
Select your option :1
Enter the no. to be inserted:7

1:Insert     2:Delete     3:Search     4:Display     5:Exit
Select your option :1
```

```
Enter the no. to be inserted:6

1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :4

The elements in the list are
      3     4     6     7
1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :2
Enter the element to be deleted : 3

1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :4
The elements in the list are
      4     6     7

1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :3
Enter the element to be searched :7

The element is in the BST
1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :3
Enter the element to be searched :10

The element is not in the BST Possition
1:Insert      2:Delete      3:Search      4:Display     5:Exit
Select your option :5
```

c. Write a C++ program for Breadth & Depth First Traversal in undirected Graphs

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

```
#define n 50
int adjmat[n][n];
class  BFSDFS
{
private:
        int Point[n];
        int pathvisit[n];
public:
        BFSDFS()
        {

        }
        ~BFSDFS()
        {}
        void create(int);
        void depthfirst(int);
        void breathfirst(int);
        void DFS(int,int);
};
void BFSDFS::DFS(int ad, int edge)
{
int k;
for(k=ad;k<edge;k++)
for(int j=0;j<edge;j++)
if(adjmat[k][j]==1)
{
if(pathvisit[j]==0)
{
pathvisit[j]=1;
cout<<Point[j]<<“==>>”;
DFS(j,edge);
}
}
}
void BFSDFS::create(int edge)
{
int i,j;
for(i=0;i<edge;i++)
{

cin>>Point[i];
pathvisit[i]=0;
}
for(i=0;i<edge;i++)
for(j=0;j<edge;j++)
adjmat[i][j]=0;
cout<<“enter the adjadency list for each edges of graph”;
cout<<“\n”;
int m,k,p;
for(i=0;i<edge;i++)
{
cout<<“enter the no of adjacency Point”;
cout<<Point[i]<<“:”;
cin>>p;
cout<<“enter the adjacency Point”;
for(j=1;j<=p;j++)
{
cin>>m;
for (k=0;k<edge;k++)
{
if(Point[k]==m)
adjmat[i][k]=1;
}
}
}
```

```
clrscr();
cout<<"\n graph created with no. of Points="<<edge<<endl<<endl;
cout<<"\n\n the adjacency matrix is :\n\n";
for(i=0;i<edge;i++)
{
for(j=0;j<edge;j++)
cout<<adjmat[i][j]<<"";
cout<<endl;
}
}
void BFSDFS::depthfirst(int edge)
{

int i=0;
for(i=0;i<edge;i++)
pathvisit[i]=0;
adjmat[0][0]=1;
pathvisit[0]=1;
cout<<"\t\t Depth First Traversal\n\n";
cout<<Point[0]<<"==>>";
DFS(0,edge);
}
void BFSDFS::breathfirst(int edge)
{
int i,j;
for(i=0;i<edge;i++)
pathvisit[i]=0;
cout<<"\t\t Breath First Traversal\n\n";
cout<<Point[0]<<"==>>";
pathvisit[0]=1;
for(i=0;i<edge;i++)
{
for(j=0;j<edge;j++)
```

```
{
    if(adjmat[i][j]==1)
    {
        if(pathvisit[j]==0)
        {
            cout<<Point[j]<<“==>>”;
            pathvisit[j]=1;
    }
    }
}
}
cout<<“x\n\n”;
}
void main()
{
    BFSDFS BD;
    int ch,edge;
do
{
    clrscr();
    cout<<“\t\t graph creation and traversal\n\n”;
    cout<<“\t\t 1.Create Graph\n\n”;
    cout<<“\t\t 2.Breadth First Traversal\n\n”;
    cout<<“\t\t 3.Depth First Traversal\n\n”;
    cout<<“\t\t 4.Exit\n\n”;
    cout<<“\t\t Enter Ur Choice\n\n”;
    cin>>ch;
    switch(ch)
    {
    case 1:
        clrscr();
        cout<<“\n\t\t Graph Creation\n\n”;
        cout<<“Enter the no. of Pointes to be created”;
```

```
                cin>>edge;
                cout<<“enter the Point value”;
                BD.create(edge);
                cout<<“press any key to continue”;
                getch();
                break;
        case 2:
                clrscr();
                 BD.breathfirst(edge);
                cout<<“press any key to continue”;
                getch();
                break;
        case 3:
                clrscr();
                 BD.depthfirst(edge);
                cout<<“X\n\n”;
                cout<<“press any key to continue”;
                getch();
                break;
        default:
                break;
        }

}while(ch!=4);
}
```

Output:

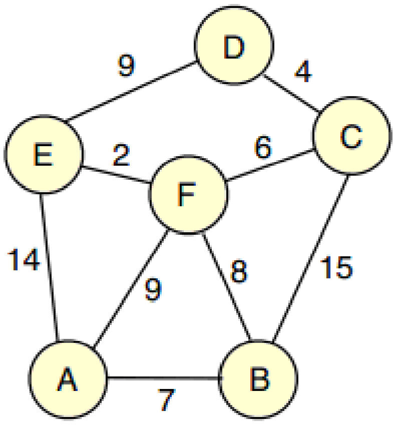
```
            graph creation and traversal

              1.Create Graph

              2.Breadth First Traversal
```

```
                3.Depth First Traversal

                4.Exit

                Enter Ur Choice

1

                Graph Creation

Enter the no. of Pointes to be created10
enter the node value1 2 3 4 5 6 7 8 9 10
                Graph Creation

Enter the no. of Pointes to be created10
enter the Point value1 2 3 4 5 6 7 8 9 10
enter the adjadency list for each edges of graph
enter the no of adjacency Point1:3
enter the adjacency Point5 6 7
enter the no of adjacency Point2:2
enter the adjacency Point4 9
enter the no of adjacency Point3:2
enter the adjacency Point7 8
enter the no of adjacency Point4:2
enter the adjacency Point2 5
enter the no of adjacency Point5:3
enter the adjacency Point1 4 9
enter the no of adjacency Point6:2
enter the adjacency Point1 10
enter the no of adjacency Point7:3
enter the adjacency Point1 3 9
enter the no of adjacency Point8:2
enter the adjacency Point3 10
enter the no of adjacency Point9:3
enter the adjacency Point2 5 7
enter the no of adjacency Point10:2
enter the adjacency Point6 8

 graph created with no. of Points=10

 the adjacency matrix is :

0000111000
0001000010
0000001100
0100100000
1001000010
1000000001
1010000010
0010000001
0100101000
0000010100
press any key to continue

           Breath First Traversal

1==>>5==>>6==>>7==>>4==>>9==>>8==>>2==>>10==>>3==>>x

press any key to continue

           Depth First Traversal

1==>>5==>>4==>>2==>>9==>>7==>>3==>>8==>>10==>>6==>>X

press any key to continue
```

Lesson 10

SHORTEST PATH

a) Write a program to find the Shortest distance [using Dijikstra's algorithm]



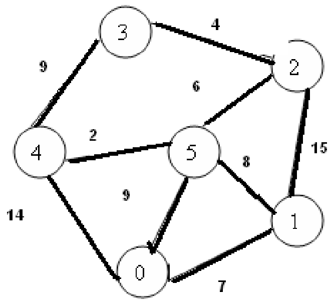
Program purpose input the Vertix A as 0, Vertix B as 1, Vertix C as 2, Vertix D as 3, Vertix E as 4, Vertix F as 5

Source code

```
#include<iostream.h>

#define INFINITY 9999

#include <stdio.h>

#include<stdlib.h>

#include<conio.h>

#define MAX 10


typedef struct node

{

        struct node *next;

        int vertex,weight;

}node;
```

```
node *G[10];//adjacency list
int n,t;// Number of vertices
void readgraph();
void insert(int vi,int vj,int w);
void dijkstra(int startnode);

void main()
{
        int u,u1;
        clrscr();
        readgraph();
        cout<<“\nEnter the starting node : “;
        cin>>u;
        dijkstra(u);
        }

void dijkstra( int startnode)
{
        int distance[MAX],pred[MAX];
        int visited[MAX],count,mindistance,nextnode,i,j;
        //pred[] stores the predecessor of each node
        //count gives the number of nodes seen so far
        // A node picked up for expansion is marked as visited[node no.]=1
                //initialize
        node *p;
        for(i=0;i<n;i++)
                {
                distance[i]=INFINITY;
                pred[i]=startnode;visited[i]=0;
                }
        distance[startnode]=0;
        count=0;
        while(count<=n)
                {
                mindistance=INFINITY ;
// nextnode is the node at minimum distance
                for(i=0;i<n+1;i++)
                      if(distance[i] < mindistance && !visited[i])
                                 {
                                mindistance=distance[i];
                                nextnode=i;
                                 }
//check if a better path exist through nextnode
                visited[nextnode]=1;
                for(p=G[nextnode];p!=NULL;p=p->next)
                      if(!visited[p->vertex])
                                if(mindistance+p->weight<distance[p->vertex])
                                           {
                                          distance[p->vertex]=mindistance+p->weight;
                                          pred[p->vertex]=nextnode;
                                           }
                count++;
                }

 //print the path and distance of each node
        for(i=0;i<n;i++)
                if(i!=startnode)
                      {
```

```
                cout<<“\n Distance of”<<i<<“=”<<distance[i];
                                cout<<“      Path = “<<i;
                j=i;
                do
                        {
                        j=pred[j];
                        cout<<“<- “<<j;
                        }while(j!=startnode);
                }
}

void readgraph()
{       int i,j;
        int adj[10][10];
        cout<<“\nEnter no. of vertices :”;
        cin>>n;
        for(i=0;i<n;i++)
                 {
                for(j=0;j<n;j++)
                {
                     cout<<“\nEnter the distance for”<<i<<“to”<<j<<“:”;
                     cin>>adj[i][j];
                     }
                     }
        //initialise G[] with NULL
        for(i=0;i<n;i++)
                G[i]=NULL;

        for(i=0;i<n;i++) //create adjacency list
                for(j=0;j<n;j++)
                              if(adj[i][j]!=0)
                        insert(i,j,adj[i][j]);
}


void insert(int vi,int vj,int w)
{
        node *p,*q;
        //acquire memory for the new node
        q=(node *)malloc(sizeof(node));
        q->vertex=vj;
        q->next=NULL;
        q->weight=w;
        //insert the node in the linked list for the vertex no. vi
        if(G[vi]==NULL)
                G[vi]=q;
        else
        {
                // go to the end of linked list
                p=G[vi];
                while(p->next!=NULL)
                    p=p->next;
                p->next=q:
        }
}
```



Sample

Input going to give as row by row manner

Node	0	1	2	3	4	5
0	0	7	0	0	14	9
1	7	0	15	0	0	8
2	0	15	0	4	0	6
3	0	0	4	0	9	0
4	14	0	0	9	0	2
5	9	8	6	0	2	0

Find the shortest distance from starting node 2 to destination.

Input

```
Enter no. of vertices :6
Enter the distance for0to0:0
Enter the distance for0to1:7
Enter the distance for0to2:0
Enter the distance for0to3:0
Enter the distance for0to4:14
Enter the distance for0to5:9
Enter the distance for1to0:7
Enter the distance for1to1:0
Enter the distance for1to2:15
Enter the distance for1to3:0
Enter the distance for1to4:0
Enter the distance for1to5:8
Enter the distance for2to0:0
Enter the distance for2to1:15
Enter the distance for2to2:0
Enter the distance for2to3:4
Enter the distance for2to4:0
Enter the distance for2to5:6
Enter the distance for3to0:0
Enter the distance for3to1:0
Enter the distance for3to2:4
Enter the distance for3to3:0
Enter the distance for3to4:9
Enter the distance for3to5:0
Enter the distance for4to0:14
Enter the distance for4to1:0
Enter the distance for4to2:0
Enter the distance for4to3:9
Enter the distance for4to4:0
Enter the distance for4to5:2
Enter the distance for5to0:9
Enter the distance for5to1:8
Enter the distance for5to2:6
Enter the distance for5to3:0
Enter the distance for5to4:2
Enter the distance for5to5:0
Enter the starting node : 2
```

Output

Passible Paths from source to destination and its corresponding total distance.

```
 Distance of0=15      Path = 0<- 5<- 2
 Distance of1=14      Path = 1<- 5<- 2
 Distance of3=4     Path = 3<- 2
 Distance of4=8     Path = 4<- 5<- 2
 Distance of5=6     Path = 5<- 2
```