**1) Implementation of array (Insert/Del)**

**#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();**

**}**

**2) Program for matrix addition, substraction, multiplication and**

**transpose of matririx**

**#include<iostream.h>**

**#include<conio.h>**

**class matrix**

**{**

**int a[5][5],b[5][5],c[5][5],d[5][5],e[5][5],f[5][5];**

**int p,q,i,j,k,n,m;**

**public:**

**void get();**

**void add();**

**void sub();**

**void trans();**

**void mul();**

**};**

**void matrix::get()**

**{**

**cout<<"\nEnter Number of Row & Column :\t";**

**cin>>n>>m;**

**cout<<"\nEnter the first Matrix:\n";**

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

**{**

**for(j=0;j<m;j++)**

**cin>>a[i][j];**

**}**

**cout<<"\nEnter Number of Row & Column :\t";**

**cin>>p>>q;**

**cout<<"\nEnter the Second Matrix:\n";**

**for(i=0;i<p;i++)**

**{**

**for(j=0;j<q;j++)**

**cin>>b[i][j];**

**}**

**}**

**void matrix::add()**

**{**

**cout<<"\nThe addition of two matrix is :\n";**

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

**{**

**for(j=0;j<m;j++)**

**{**

**c[i][j]=a[i][j]+b[i][j];**

**cout<<c[i][j]<<"\t";**

**}**

**cout<<"\n";**

**}**

**4**

**}**

**void matrix::sub()**

**{**

**cout<<"\nThe Subtraction of two matrix is :\n";**

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

**{**

**for(j=0;j<m;j++)**

**{**

**d[i][j]=a[i][j]-b[i][j];**

**cout<<d[i][j]<<"\t";**

**}**

**cout<<"\n";**

**}**

**}**

**void matrix::trans()**

**{**

**cout<<"\nThe Transpose of first matrix is :\n";**

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

**{**

**for(j=0;j<m;j++)**

**{**

**e[i][j]=a[j][i];**

**cout<<e[i][j]<<"\t";**

**}**

**cout<<"\n";**

**}**

**}**

**void matrix::mul()**

**{**

**cout<<"\nThe Matrix Multiplication is : \n";**

**if(m==p)**

**{**

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

**{**

**for(j=0;j<q;j++)**

**{**

**c[i][j]=0;**

**for(k=0;k<p;k++)**

**{**

**c[i][j]=c[i][j]+a[i][k]\*b[k][j];**

**}**

**cout<<c[i][j]<<"\t";**

**}**

**cout<<"\n";**

**}**

**}**

**else**

**cout<<"\n Matrix Multiplication not possible";**

**}**

**void main()**

**{**

**clrscr();**

**matrix m;**

**m.get();**

**5**

**m.add();**

**m.sub();**

**m.trans();**

**m.mul();**

**getch();**

**}**

**3) Implement Stack for Integer/character perform different**

**operation on stack (push, pop, peep, change).**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**int n;**

**class stack**

**{**

**private:**

**int s[10],top,ele,i; // char s[10] for character**

**public:**

**stack()**

**{**

**top=-1;**

**}**

**void push();**

**void dis();**

**void pop();**

**void peep();**

**void change();**

**};**

**void stack::push()**

**{**

**if(top>=n-1)**

**cout<<"\nStack is overflow:";**

**else**

**{**

**cout<<"\nEnter element:";**

**cin>>ele;**

**top++;**

**s[top]=ele;**

**}**

**}**

**void stack::dis()**

**{**

**if(top==-1)**

**{**

**cout<<"\n Stack is Empty";**

**}**

**else**

**{**

**cout<<"\nElements in stack are:\n";**

**for(i=top;i>=0;i--)**

**cout<<s[i]<<"\t";**

**}**

**}**

**void stack::pop()**

**{**

**if(top==-1)**

**{**

**cout<<"\nUnderflow";**

**}**

**else**

**{**

**cout<<"\nPop ele is "<<s[top];**

**top--;**

**}**

**}**

**void stack::peep()**

**{**

**cout<<"\nEnter position:";**

**cin>>i;**

**if((top-i+1)<0)**

**{**

**cout<<"\nUnderflow";**

**}**

**else**

**{**

**cout<<"\nPeep ele is "<<s[top-i+1];**

**}**

**}**

**void stack::change()**

**{**

**cout<<"\nEnter position ";**

**cin>>i;**

**if((top-i+1)<0)**

**{**

**cout<<"\nUnderflow";**

**}**

**else**

**{**

**int n; //char n; for character**

**cout<<"\nEnter element:";**

**cin>>n;**

**s[top-i+1]=n;**

**}**

**}**

**void main()**

**{**

**clrscr();**

**stack s;**

**cout<<"Enter size of stack";**

**cin>>n;**

**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: s.pop();break;**

**case 4: s.peep(); break;**

**case 5: s.change(); break;**

**case 6: exit(0);**

**}**

**}**

**getch();**

**}**

**6) Program to Implement Stack using LL**

**#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();**

**}**

**19) Implement Infix to Postfix operation using stack.**

**#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();**

**}**

**4) Implement linear queue for integer / character perform different**

**operation on queue ( insert,delete,display)**

**#include<iostream.h>**

**#include<conio.h>**

**#include<process.h>**

**int m;**

**class queue**

**{**

**int f,r,q[10],n,i; //char q[10],n for character**

**public:**

**queue()**

**{**

**f=r=0;**

**}**

**void insert();**

**void del();**

**void dis();**

**};**

**void queue::insert()**

**{**

**if(r==m)**

**cout<<"\nOverflow";**

**else**

**{**

**cout<<"\nEnter Element in Queue=";**

**cin>>n;**

**if(f==0)**

**f=1;**

**r++;**

**q[r]=n;**

**}**

**}**

**void queue::del()**

**{**

**if(f==0)**

**{**

**cout<<"\nUnderflow";**

**}**

**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**

**14**

**{**

**cout<<"\nElements in queue are:";**

**for(i=f;i<=r;i++)**

**cout<<q[i]<<"\t";**

**}**

**}**

**void main()**

**{**

**clrscr();**

**queue q;**

**int ch;**

**cout<<”Enter size of queue”;**

**cin>>m;**

**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();**

**}**

**7) Implement Queue using Link List**

**#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();**

**}**

**5) Implement Circular Queue, perform different operation of**

**circular queue (push ,pop, show)**

**#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**

**{**

**18**

**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[j];**

**}**

**}**

**}**

**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);**

**}**

**8) Implement Linear and Binary Search**

**#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();**

**}**

**9) Perform Bubble Sort Ascending/Descending order for int/String**

**#include<iostream.h>**

**#include<conio.h>**

**class demo**

**{**

**int a[10],temp; //For string char a[10][10],temp[10];**

**int,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++) // for string**

**{**

**if(a[j]>a[j+1]) // if(strcmp(a[j],a[j+1])>0)**

**{**

**temp=a[j]; // strcpy(temp,a[j]);**

**a[j]=a[j+1]; // strcpy(a[j],a[j+1]);**

**a[j+1]=temp; // strcpy(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++) //for string**

**{**

**if(a[j]<a[j+1]) // if(strcmp(a[j],a[j+1])<0)**

**{**

**temp=a[j]; // strcpy(temp,a[j]);**

**a[j]=a[j+1]; // strcpy(a[j],a[j+1]);**

**a[j+1]=temp; // strcpy(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();**

**}**

**10) Perform Selection Sort Ascending/Descending order for int/String**

**#include<iostream.h>**

**#include<conio.h>**

**class demo**

**{**

**int a[10],temp; // int a[10][10],temp[10] for string**

**int i, min\_index,j,n;**

**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++) // for string**

**{**

**if(a[j]<a[min\_index]) // if(strcmp(a[j],a[min\_index])<0)**

**min\_index=j;**

**}**

**if(min\_index!=i)**

**{**

**temp=a[min\_index]; // strcpy(temp,a[min\_index]);**

**a[min\_index]=a[i]; // strcpy(a[min\_index],a[i]);**

**a[i]=temp; // strcpy(a[i],temp);**

**}**

**}**

**}**

**void demo::dsc\_sort()**

**{**

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

**{**

**min\_index=i;**

**for(j=i+1;j<=n;j++) // for string**

**{**

**if(a[j]>a[min\_index]) // if(strcmp(a[j],a[min\_index])>0)**

**min\_index=j;**

**}**

**if(min\_index!=i)**

**{**

**37**

**temp=a[min\_index]; // strcpy(temp,a[min\_index]);**

**a[min\_index]=a[i]; // strcpy(a[min\_index],a[i]);**

**a[i]=temp; // strcpy(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();**

**}**

**11) Implement Insertion Sort**

**#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();**

**}**

**12 ) Implement Quick sort for integer in Ascending / Descending order**

**#include<iostream.h>**

**#include<conio.h>**

**#include<string.h>**

**class demo**

**{**

**int x[20],temp;**

**int a,n,i,j,left,right;**

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

**cin>>x[i];**

**asort(1,n);**

**}**

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

**{**

**if(p<q)**

**{**

**j=partition(p,q);**

**asort(p,j-1);**

**asort(j+1,q);**

**}**

**}**

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

**{**

**a=x[lb];**

**left=lb+1;**

**right=ub;**

**do //for Descending**

**{**

**while(x[left]<a) // while(x[left]>a)**

**left++;**

**while(x[right]>a) // while(x[right]<a)**

**right--;**

**if(left<right)**

**{**

**temp=x[left];**

**x[left]=x[right];**

**x[right]=temp;**

**}**

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

**x[lb]=x[right];**

**42**

**x[right]=a;**

**return(right);**

**}**

**void demo::disp()**

**{**

**cout<<"\nThe array element are:";**

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

**cout<<x[i]<<"\t";**

**}**

**void main()**

**{**

**clrscr();**

**demo d;**

**d.get();**

**cout<<"\nAfter Ascending sort"; // Descending**

**d.disp();**

**getch();**

**}**

**13) Implement Merge sort in ascending / descending order**

**#include<iostream.h>**

**#include<conio.h>**

**#include<stdio.h>**

**int n; //remember that n should be declare global**

**class merge**

**{**

**int a[10],b[10],i,j;**

**public:**

**void read();**

**void merge\_sort(int l,int h);**

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

**void disp();**

**};**

**void merge::read()**

**{**

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

**cin>>a[i];**

**}**

**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 m,int high)**

**{**

**int h=low;**

**int i=low;**

**j=m+1;**

**while((h<=m)&&(j<=high))**

**{**

**if(a[h]<=a[j]) //Change descending order if(a[h]>=a[j])**

**{**

**b[i]=a[h];**

**i++;**

**h++;**

**}**

**else**

**{**

**b[i]=a[j];**

**i++;**

**j++;**

**}**

**44**

**}**

**if(h<=m)**

**{**

**while(h<=m)**

**{**

**b[i]=a[h];**

**i++;**

**h++;**

**}**

**}**

**else**

**{**

**while(j<=h)**

**{**

**b[i]=a[j];**

**i++;**

**j++;**

**}**

**}**

**for(int k=low;k<=high;k++)**

**a[k]=b[k];**

**}**

**void merge::disp()**

**{**

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

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

**}**

**void main()**

**{**

**clrscr();**

**int l,h;**

**merge m;**

**cout<<"\nEnter Elements";**

**cin>>n;**

**h=n-1;**

**l=0;**

**m.read();**

**cout<<"\n\nDisplay the array elements\n";**

**m.disp();**

**m.merge\_sort(l,h);**

**cout<<"\nAfter Sorting\n";**

**m.disp();**

**getch();**

**}**

**15) mplement Heap Sort in ascending / descending order**

**#include<iostream.h>**

**#include<conio.h>**

**class heap**

**{**

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

**public:**

**void get();**

**void create();**

**void sort();**

**void display();**

**};**

**void heap::get()**

**{**

**cout<<"\nEnter range:";**

**cin>>n;**

**cout<<"\nEnter the elements\n";**

**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 descending order key<a[j]**

**{**

**a[i]=a[j];**

**i=j;**

**j=i/2;**

**if(j<1)**

**j=1;**

**}**

**a[i]=key;**

**}**

**}**

**void heap::sort()**

**{**

**create();**

**cout<<"\nMax Heap Tree";**

**display();**

**for(q=n;q>=2;q--)**

**{**

**temp=a[1];**

**a[1]=a[q];**

**a[q]=temp;**

**i=1;**

**key=a[1];**

**j=2;**

**if(j+1<q)**

**if(a[j+1]>a[j])**

**j++;**

**while(j<=q-1 && a[j]>key)**

**{**

**a[i]=a[j];**

**i=j;**

**j=i\*2;**

**if(j+1<q)**

**if(a[j+1]>a[j])**

**j++;**

**else**

**if(j>n)**

**j=n;**

**a[i]=key;**

**}**

**}**

**}**

**void heap::display()**

**{**

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

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

**}**

**void main()**

**{**

**clrscr();**

**heap h;**

**h.get();**

**h.sort();**

**cout<<"\nSorted element are:";**

**h.display();**

**getch();**

**}**

**\*/ Output \*/**

**Enter range: 5**

**Enter the elements**

**12 3 45 6 18**

**Max Heap Tree45 18 12 3 6**

**Sorted element are:3 6 12 18 45**

**Implement Tree Traversal**

**#include<iostream.h>**

**#include<process.h>**

**#include<conio.h>**

**struct ver**

**{**

**int data;**

**ver \*left,\*right;**

**};**

**class tree**

**{**

**public:**

**ver\* create(int,ver\*);**

**void in(ver\*);**

**void post(ver\*);**

**void pre(ver\*);**

**};**

**ver \*tree::create(int c, ver \*node)**

**{**

**if(node==NULL)**

**{**

**node=new ver;**

**node->data=c;**

**node->left=NULL;**

**node->right=NULL;**

**return node;**

**}**

**else**

**{**

**if(c<node->data)**

**node->left=create(c,node->left);**

**else**

**node->right=create(c,node->right);**

**return node;**

**}**

**}**

**void tree::in(ver \* node)**

**{**

**if(node)**

**{**

**in(node->left);**

**cout<<node->data<<"\t";**

**in(node->right);**

**}**

**}**

**void tree::pre(ver \* node)**

**{**

**if(node)**

**{**

**cout<<node->data<<"\t";**

**pre(node->left);**

**pre(node->right);**

**}**

**}**

**void tree::post(ver \* node)**

**{**

**if(node)**

**{**

**post(node->left);**

**post(node->right);**

**cout<<node->data<<"\t";**

**}**

**}**

**void main()**

**{**

**clrscr();**

**tree t;**

**ver \*r=new ver;**

**r=NULL;**

**int n,ch;**

**cout<<"\n 1:insert 2:inorder 3:preorder 4:postorder 5:exit :";**

**while(ch!=5)**

**{**

**cout<<"\nEnter Choice:";**

**cin>>ch;**

**switch(ch)**

**{**

**case 1: cout<<"\nEnter Node:";**

**cin>>n;**

**r=t.create(n,r);**

**break;**

**case 2: cout<<"\nInorder Traversal:";**

**t.in(r);**

**break;**

**case 3: cout<<"\nPreorder Traversal:";**

**t.pre(r);**

**break;**

**case 4: cout<<"\nPostorder Traversal:";**

**t.post(r);**

**break;**

**case 5: exit(0);**

**}**

**}**

**getch();**

**}**

**b.get();**

**b.dbfs();**

**getch();**

**}**

**16) Implement All Pair Shortest Path (Floyd-Warshall)**

**#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();**

**}**