**Sorting Techniques**

**Practical No 1**

**Aim:** Write a Program to demonstrate Bubble Sort.

**Code:**

#include<iostream>

using namespace std;

class bubble\_sort

{

public:

int n; //n=no of elements in array

int a[100]; //a[i]=elements of array

int temp;

void get\_size()

{

cout<<"\nEnter Array Size: "; //insert the no of elements that user wants

cin>>n;

}

void get\_val()

{

cout<<"\nEnter Elements in array: ";

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

{

cin>>a[i];

} // inserting elements in array

}

void show()

{

cout<<"\nElements in array: \n";

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

{

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

}

}

void sort\_array()

{

for(int i=0;i<n;i++) //i=step or iteration for sorting

{

for(int j=0;j<n;j++) //j=position of elements in array

{

if(a[j]>a[j+1]) //check for the elements in array

{ //such as a[0] is greter than a[1] and so on

temp=a[j+1]; //then swap both the elements

a[j+1]=a[j]; //simillarly check for all the elements in array

a[j]=temp; //and sort the array with bubble sort

}

}

}

}

};

int main()

{

cout<<"Bubble Sort"<<endl;

bubble\_sort b;

b.get\_size();

b.get\_val();

b.show();

b.sort\_array();

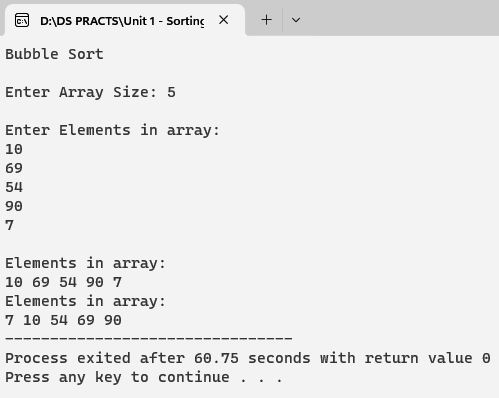
b.show();

//getch();

return 0;

}

**Output:**

****

**Practical No 2**

**Aim:** Write a Program to demonstrate Insertion Sort.

**Code:**

#include<iostream>

using namespace std;

class insertion

{

public:

int ar[10];

int n,j,i,temp;

void insert()

{

cout<<"Enter element size:";

cin>>n; //n=no of elements in array

cout<<"Elements are:";

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

{

cin>>ar[i]; //inserting elements in array

}

}

void display()

{

cout<<"\n Before sorted array:";

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

{

cout<<ar[i]<<" "; //display the sorted array elements

}

}

void sort()

{

for(j=1;j<n;j++) //j=position of elements in array

{

i=j-1; //first element is stored in left side of array i.e.sorted sublist

temp=ar[j]; //temp=sorted elements are stored in temporary sublist

while((i>=0)&&(temp<ar[i])) //check for first two elements are greter than or not

{

ar[i+1]=ar[i]; //such as a[0] is greter than a[1] then swap both the elements

i--; //decrement the position of array

}

ar[i+1]=temp; //swap the elements in sorted sublist and is stored in temp

}

}

void show()

{

cout<<"\n After sorted array:";

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

{

cout<<ar[i]<<" "; //display the sorted array elements

}

}

};

int main()

{

insertion ob;

ob.insert();

ob.display();

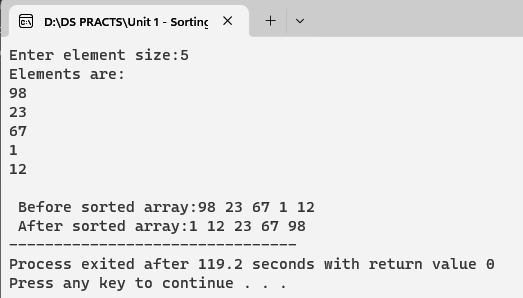
ob.sort();

ob.show();

//getch();

}

**Output:**

****

**Practical No 3**

**Aim:** Write a Program to demonstrate Selection Sort.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

class selection\_sort

{

public:

int i,j,n,loc,temp,min,a[30];

void get\_values()

{

cout<<"Enter the size:";

cin>>n; //n=no of elements in array

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

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

{

cin>>a[i]; //inserting elements in array

}

}

void display()

{

cout<<"\n Before sorting list is as follows: \n";

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

{

cout<<a[i]<<" "; //display the array

}

}

void sort()

{

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

{

min=a[i]; //finds the lowest value in array

loc=i; //that element will stored in loc i.e.position of elements

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

{

if(min>a[j]) //check that lowest value with first element in array

{

min=a[j]; //if the that value is less than first element

loc=j; //then store the element in loc

}

}

temp=a[i]; // swap both the elements

a[i]=a[loc];

a[loc]=temp;

}

}

void show()

{

cout<<"\n After sorting list is as follows\n";

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

{

cout<<a[i]<<" "; //display the sorted array

}

}

};

int main()

{

selection\_sort s;

s.get\_values();

s.display();

s.sort();

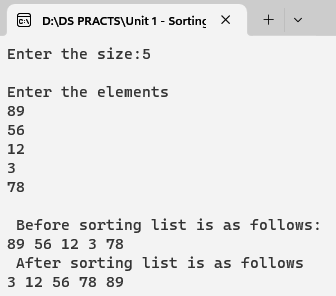
s.show();

getch();

return 0;

}

**Output:**

****

**Practical No 4**

**Aim:** Write a Program to demonstrate Shell Sort.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

class Shell

{

public:

int i,j,n;

double array[10];

double temp;

void insert()

{

cout<<"Enter size for array";

cin>>n; //n=no of elements in array

cout<<"Enter the elements in array=";

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

{

cin>>array[i]; //inerting elements in array

}

}

void sort()

{

int inc=3; //assume interval is 3

while(inc>0)

{

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

{

j=i;

temp=array[i]; //store the first element in temp

while((j>=inc)&&array[j-inc]>temp) //make sublist of all values located at interval

{ //3 positions

array[j]=array[j-inc]; //check with that element in sublist

j=j-inc; //decrement the value of interval

}

array[j]=temp; //swap the elements with bubble sort

}

if(inc/2!=0) //check interval is divided by 2 is not equal 0

inc=inc/2; //then interval is divided by 2

else if(inc==1) //else check interval=1

inc=0; //then interval=0

else

inc=1; //else interval=1

}

}

void display()

{

cout<<"Array after bubble sort:";

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

{

cout<<" "<<array[i]; //display the sorted array

}

}

};

int main()

{

Shell obj;

obj.insert();

obj.sort();

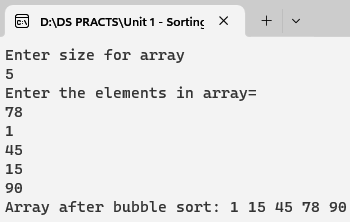
obj.display();

getch();

return 0;

}

**Output:**

****

**Practical No 5**

**Aim:** Write a Program to demonstrate Radix Sort.

**Code:**

#include <iostream>

#include <cstdlib>

#include<conio.h>

using namespace std;

int getMax(int arr[], int n)

{

int max = arr[0]; //declare first element i.e.arr[0]=max

for (int i = 1; i < n; i++) //check elements are less than maximum size and no of elemts

if (arr[i] > max) //check first element of array is greter than maximum element

max = arr[i];

return max;

}

void countSort(int arr[], int n, int exp)

{

int output[n];

int i, count[10] = {0};

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

count[(arr[i] / exp) % 10]++;

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

for (i = n - 1; i >= 0; i--)

{

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

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

arr[i] = output[i];

}

void radixsort(int arr[], int n)

{

int m = getMax(arr, n);

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(arr, n, exp);

}

int main()

{

int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};

int n = sizeof(arr)/sizeof(arr[0]);

radixsort(arr, n);

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

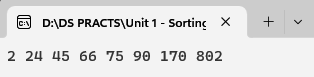
cout << arr[i] << " ";

getch();

return 0;

}

**Output:**

****

**Searching and Hashing Techniques**

**Practical No 6**

**Aim:** Write a Program to demonstrate Linear Search.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

class linear\_search

{

public:

int a[5],num,i,pos, n;

bool flag;

void getdata();

void search();

};

void linear\_search::getdata()

{

cout<<"Enter the array elements: ";

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

{

cin>>a[i];

}

cout<<"\nEntered array elements: ";

for(i=0;i<5;i++)

{

cout<<a[i]<<" "; //inserting elements in a array

}

}

void linear\_search::search()

{

cout<<"\nEnter number to search: ";

cin>>num; //num=number to be searched in array

for(i=0;i<5;i++) //check with a[0]

{

if(a[i]==num) //if a[0] = that no to be searched

{

flag=true; //then set the boolean flag is true

pos=i; //store the ith position into pos

break;

}

else

{

flag=false; //else flag set to false

}

}

if(flag) //if flag=true

{

cout<<"\nNumber found at position "<<pos; //then print the position of number

}

else

{

cout<<"\nNumber not found"; //else print no not found

}

}

int main()

{

linear\_search s;

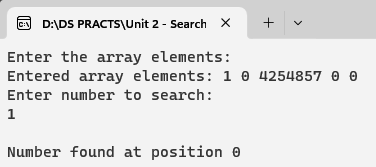
s.getdata();

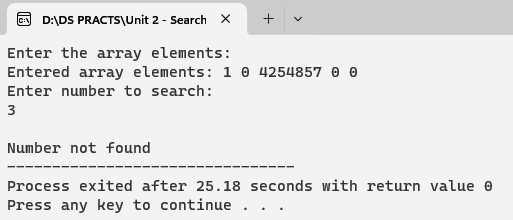
s.search();

getch();

}

**Output:**

****

****

**Practical No 7**

**Aim:** Write a Program to Binary Search.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

class binary\_search

{

public:

int a[5],flag;

int i,j,index,num,temp,mid,low,high;

void getdata();

void search();

void sort\_array();

};

void binary\_search::sort\_array()

{

for(int i=0;i<5;i++)

{

for(int j=0;j<5;j++)

{

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

{

temp=a[j+1];

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

a[j]=temp;

}

}

}

}

void binary\_search::getdata()

{

cout<<"\nEnter array Elements: ";

for(i=0;i<5;i++)

{

cin>>a[i+1];

}

sort\_array();

cout<<"\nSorted Array Elements: ";

for(i=0;i<5;i++)

{

cout<<a[i+1];

}

}

void binary\_search::search()

{

cout<<"\nEnter an element to search: ";

cin>>num;

low=0;

high=4;

while(low<=high)

{

mid=(low+high)/2;

if(a[mid]==num)

{

cout<<"\nNumber is found at position "<<mid;

break;

}

if(a[mid]>num)

{

high=mid-1;

}

if(a[mid]<num)

{

low=mid +1;

}

if(a[mid]!=num)

{

flag=false;

}

}

if(flag==false)

{

cout<<"\nNumber is not found!!!";

}

}

int main()

{

binary\_search b;

b.getdata();

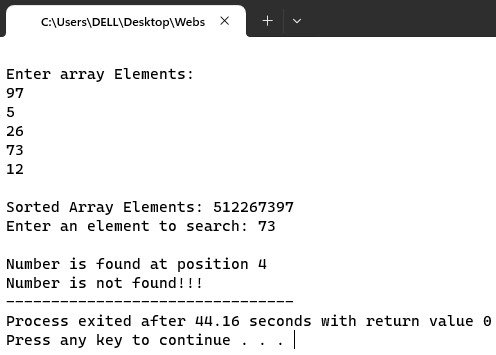
b.search();

getch();

return 0;

}

**Output:**

****

**Practical No 8**

**Aim:** Write a Program to demonstrate Modulo Division.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int hashsearch(int a[],int x , int n)

{

int index,start;

index=x%n;

if(a[index]==x)

return index;

else

{

start=index;

do

{

index=(index+1)%n;

if(a[index]==x)

return index;

else

if(a[index]==-1)

break;

}

while(index!=start);

return -1;

}

}

int main()

{

int hash[13],i,x,index,k;

cout<<"Hashing using Modulo division and linear probing\n";

cout<<"Hash table creation\n";

for(i=0;i<13;i++)

hash[i]=-1;

for(i=0;i<7;i++)

{

cout<<"Enter number\n";

cin>>x;

index=x%13;

while(hash[index]!=-1)

index=(index+1)%13;

hash[index]=x;

}

cout<<"Hash table\n";

for(i=0;i<13;i++)

cout<<i<<"\t"<<hash[i]<<"\n";

do

{

cout<<"Elements to be searched to stop enter -1\n";

cin>>x;

if(x>=0)

{

k=hashsearch(hash,x,13);

if(k>=0)

cout<<"Element"<<x<<"is present at hash index["<<k<<"]"<<endl;

else

cout<<"not present";

}

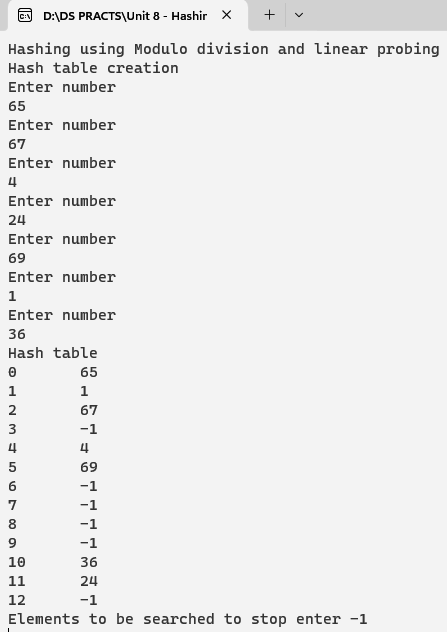
}

while(x>=0);

getch();

}

**Output:**

****

**Practical No 9**

**Aim:** Write a Program to demonstrate Digit Extraction.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int hash[19],i=0,j=0,x,index,k,first,last,fifth,third;

cout<<"Hashing using Digit extraction and linear probing\n";

cout<<"Hash table creation\n";

for(i=0;i<19;i++)

hash[i]=-1;

for(j=0;j<7;j++)

{

i=0;

cout<<"Enter number\n";

cin>>x;

int temp =x;

while(x>0)

{

last = x%10;

if(i==1)

{

fifth = last;

}

if(i==3)

{

third = last;

}

if(i==5)

{

first = last;

}

x = x/10;

i++;

}

index = first \* 100 + third \* 10 + fifth;

index = index % 19;

while(hash[index]!=-1)

index=(index+1)%19;

hash[index] = temp;

}

for(i=0;i<19;i++)

{

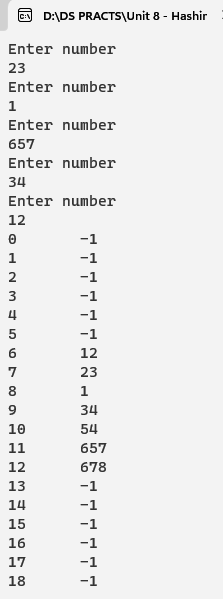
cout<<i<<"\t"<<hash[i]<<"\n";

}

getch();

}

**Output:**

****

**Practical No 10**

**Aim:** Write a Program to demonstrate Fold Shift.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int hash[19],i=0,j=0,x,index,k,first, second, four, last, fifth, third, six;

cout<<"Hashing using linear probing\n";

cout<<"Hash table creation\n";

for(i=0;i<19;i++)

hash[i]=-1;

for(j=0;j<9;j++)

{

i=1;

cout<<"Enter number\n";

cin>>x;

int temp =x;

while(x>0)

{

last = x%10;

if(i==2)

{

fifth = last;

}

if(i==4)

{

third = last;

}

if(i==6)

{

first = last;

}

if(i==1)

{

six = last;

}

if(i==3)

{

four = last;

}

if(i==5)

{

second = last;

}

x = x/10;

i++;

}

int temp1 = first\*10+second;

int temp2 = third\*10+four;

int temp3 = fifth\*10+six;

int tempres = temp1+temp2+temp3;

cout << tempres;

int r=0;

if(tempres>19)

{

index = tempres % 19;

}

else

{

index = tempres;

}

cout<<"("<<index<<")\n"; while(hash[index]!=-1)

index=(index+1)%19;

hash[index] = temp;

}

for(i=0;i<19;i++)

{

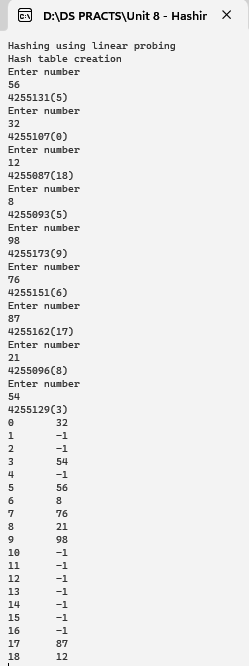
cout<<i<<"\t"<<hash[i]<<"\n";

}

getch();

}

**Output:**

****

**Practical No 11**

**Aim:** Write a Program to demonstrate Fold Boundary.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int hash[19],i=0,j=0,x,index,k,first, second, four, last, fifth, third, six;

cout<<"Hashing using linear probing\n";

cout<<"Hash table creation\n";

for(i=0;i<19;i++)

hash[i]=-1;

for(j=0;j<9;j++)

{

i=1;

cout<<"Enter number\n";

cin>>x;

int temp =x;

while(x>0)

{

last = x%10;

if(i==2)

{

fifth = last;

}

if(i==4)

{

third = last;

}

if(i==6)

{

first = last;

}

if(i==1)

{

six = last;

}

if(i==3)

{

four = last;

}

if(i==5)

{

second = last;

}

x = x/10;

i++;

}

int temp1 = second\*10+first;

int temp2 = third\*10+four;

int temp3 = six\*10+fifth;

int tempres = temp1+temp2+temp3;

cout << tempres;

int r=0;

if(tempres>19)

{

index = tempres % 19;

}

else

{

index = tempres;

}

cout<<"("<<index<<")\n";

while(hash[index]!=-1)

index=(index+1)%19;

hash[index] = temp;

}

for(i=0;i<19;i++)

{

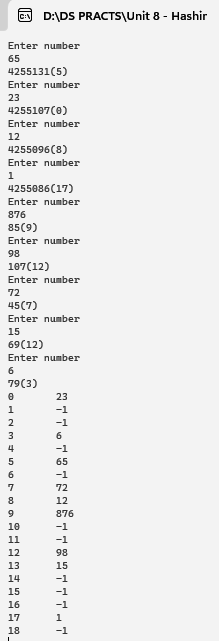
cout<<i<<"\t"<<hash[i]<<"\n";

}

getch();

}

**Output:**

****

**Practical No 12**

**Aim:** Write a Program to demonstrate Linear Probe for Collision Resolution.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int hashsearch(int a[],int x , int n)

{

int index,start;

index=x%n;

if(a[index]==x)

return index;

else

{

start=index;

do

{

index=(index+1)%n;

if(a[index]==x)

return index;

else

if(a[index]==-1)

break;

}

while(index!=start);

return -1;

}

}

int main()

{

int hash[13],i,x,index,k;

cout<<"Hashing using Modulo division and linear probing\n";

cout<<"Hash table creation\n";

for(i=0;i<13;i++)

hash[i]=-1;

for(i=0;i<7;i++)

{

cout<<"Enter number\n";

cin>>x;

index=x%13;

while(hash[index]!=-1)

index=(index+1)%13;

hash[index]=x;

}

cout<<"Hash table\n";

for(i=0;i<13;i++)

cout<<i<<"\t"<<hash[i]<<"\n";

do

{

cout<<"Elements to be searched to stop enter -1\n";

cin>>x;

if(x>=0)

{

k=hashsearch(hash,x,13);

if(k>=0)

cout<<"Element"<<x<<"is present at hash index["<<k<<"]"<<endl;

else

cout<<"not present";

}

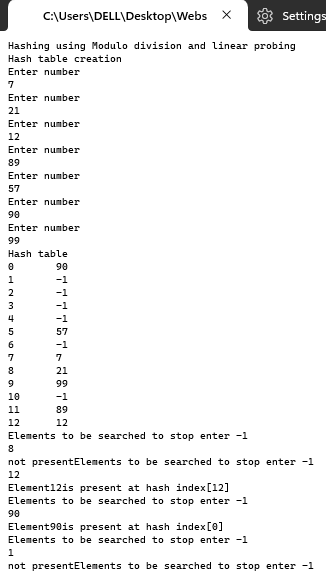
}

while(x>=0);

getch();

}

**Output:**

****

**Stacks**

**Practical No 13**

**Aim:** Write a Program to demonstrate Stack using Array Implementation.

**Code:**

#include<iostream>

#include<conio.h>

#define MAX 5

using namespace std;

int top = -1; //delcare topelement=-1

int stack\_arr[MAX]; //declare stack array size i.e 5

void push() //inserting elements in stack array

{

int pushed\_item; //declare pushed\_item=element to be inserted in stack

if(top == (MAX-1)) //check top element is max -1

cout<<"Stack Overflow\n"; //then stack is full print stack overflow

else

{

cout<<"Enter the item to be pushed in stack : "; //else it will ask for no to be enetered in stack

cin>>pushed\_item; //that stored in pushed\_item

top=top+1; //top is incremented by 1

stack\_arr[top]=pushed\_item; //pushed\_item is stored in top of the stack array

}

}

void pop() //deleting elements in stack array

{

if(top == -1) //check top =-1 i.e stack is not full

cout<<"Stack Underflow\n"; //print stack undeflow

else

{

cout<<"Popped element is : "<<stack\_arr[top]<<"\n";; //else deleted the top element in array

top=top-1; //top is decreamented

}

}

void display() //displaying elements in stack array

{

int i;

if(top == -1) //check top is not full

cout<<"Stack is empty\n";

else

{

cout<<"Stack elements :\n"; //else print the stack elements

for(i = top; i >=0; i--) //check with top elements till 0th position

cout<<"\n"<<stack\_arr[i]<<"\n"; ; //print the stack elements

}

}

int main()

{

int choice;

while(1)

{

cout<<"1.Push\n";

cout<<"2.Pop\n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1 :

push();

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

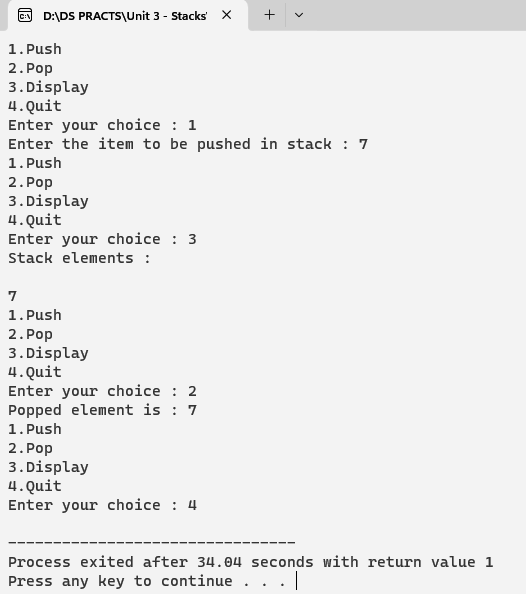
cout<<"Wrong choice\n";

}

}

}

**Output:**

****

**Practical No 14**

**Aim:** Write a Program to demonstrate Stack using Linked List Implementation.

**Code:**

#include<stdio.h>

#include<iostream>

#include<conio.h>

using namespace std;

void push();

void pop();

void display();

struct node

{

int info; //delclare info i.e elements in linked list

struct node \*link; //create structure node that points to the array elements i.e named as link

} \*top=NULL; //initially top pointer delcares as null

int main()

{

int choice;

while(1)

{

cout<<"\n";

cout<<"1.Push\n";

cout<<"2.Pop\n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

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

cin>>choice;

switch(choice)

{

case 1:push();break;

case 2:pop();break;

case 3:display();break;

case 4:exit(1);

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

}

}

}

void push()

{

struct node \*tmp; //create tmp pointer that points to the new node

int pushed\_item,no; //declare inserting element and no in array linked list

cout<<"How many element u want to insert: ";

cin>>no;

for(int i=1; i<=no; i++) //check the no is less than 1

{

tmp = new node; //declare tmp is new node for linked list

cout<<"Input the new value to be pushed on the stack : ";

cin>>pushed\_item; //insert the new value into linked list stack array

tmp->info=pushed\_item; //new value is points to the tmp pointer in linkedlist info

tmp->link=top; //that tmp i.e new node points to the top element of linked list stack array

top=tmp; //new value i.e stored in tmp that stored top node of linked list stack array

}

}

void pop()

{

struct node \*tmp; //create a node that points to tmp i.e elements inserted in linked list

if(top == NULL) //check top is null i.e stack does nove any elements

cout<<"Stack is empty\n"; //print stack is empty

else

{

tmp=top; //else top stores the tmp i.e new node inserted into array

cout<<"Popped item is"<<"\t->"<<tmp->info<<"\n\n"; //print the deleted item

top=top->link; //top points to next node i.e link

delete tmp; //delete the new node that stored in tmp

}

}

void display()

{

struct node \*ptr; //create ptr pointer to store the elements in array to display

ptr=top; //top element points to ptr pointer i.e top value stored in ptr

if(top==NULL) //check top is null

cout<<"Stack is empty\n"; //then print stack is empty

else

{

cout<<"Stack elements :\n"; //else print stack elements

while(ptr!= NULL) //check ptr is not equal to null i.e ptr points to the node that stored in info linked list

{

cout<<"\n"<<ptr->info<<"\n"; //print the elements in info linked list stack array

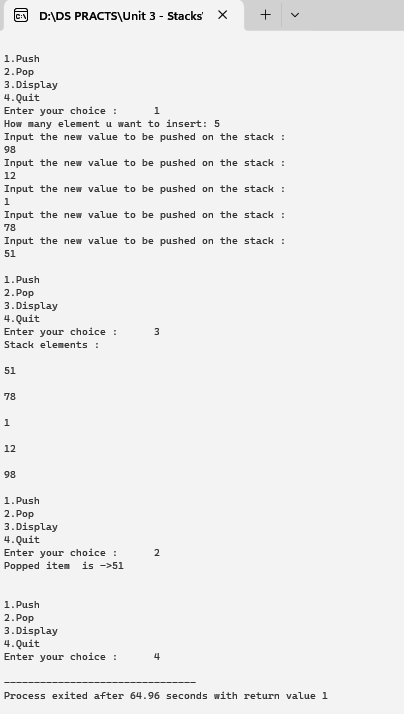
ptr = ptr->link; //link points to the ptr pointer

}

}

}

**Output:**

****

**Practical No 15**

**Aim:** Write a Program to demonstrate Stack Application - Evaluation of postfix Expression.

**Code:**

#include <bits/stdc++.h>

using namespace std;

int evaluatePostfix(string exp)

{

stack<int> st;

for (int i = 0; i < exp.size(); ++i) {

if (isdigit(exp[i]))

st.push(exp[i] - '0');

else {

int val1 = st.top();

st.pop();

int val2 = st.top();

st.pop();

switch (exp[i]) {

case '+':

st.push(val2 + val1);

break;

case '-':

st.push(val2 - val1);

break;

case '\*':

st.push(val2 \* val1);

break;

case '/':

st.push(val2 / val1);

break;

}

}

}

return st.top();

}

int main()

{

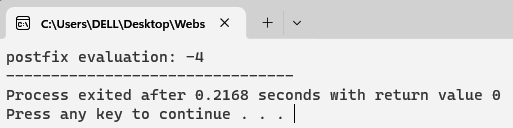
string exp = "231\*+9-";

cout << "postfix evaluation: " << evaluatePostfix(exp);

return 0;

}

**Output:**

****

**Practical No 16**

**Aim:** Write a Program to demonstrate Stack Application - Balancing of parenthesis.

**Code:**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#include<conio.h>

#include<string.h>

using namespace std;

const int MAX=5;

class stack

{

public:

int count,top;

char arr[MAX];

stack()

{

count=0;

top=-1;

}

void push(char);

void pop();

};

void stack::push(char d)

{

int flag;

if(count==MAX)

{

cout<<"\nStack is full";

}

else

{

count++;

top++;

arr[top]=d;

}

}

void stack:: pop()

{

if(top==-1)

{

cout<<"\nStack is empty";

}

else

{

char d=arr[top];

cout<<d<<endl;

top--;

count--;

}

}

int main()

{

stack s1;

char exp[20];

char ch;

int num,n,i;

cout<<"\nEnter the expression\n";

cin>>exp;

num=strlen(exp);

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

{

if(exp[i]=='(')

{

s1.push(exp[i]);

}

else if(exp[i]==')')

{

s1.pop();

}

}

if(s1.top!=-1)

{

cout<<"\nNo matching parenthesis,Wrong expression\n";

}

else

{

cout<<"\nMatching parenthis found expressions is correct\n";

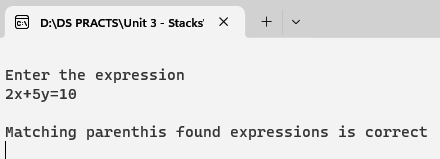
}

getch();

return 0;

}

**Output:**

****

**Queue**

**Practical No 17**

**Aim:** Write a Program to demonstrate Queue using Linked List Implementation.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

struct node

{

int info;

struct node \*link;

}\*front=NULL,\*rear=NULL;

void insert()

{

struct node \*tmp;

int added\_item,no;

cout<<"how many element u want to insert into queue:";

cin>>no;

for(int i=1; i<=no; i++)

{

tmp = new node;

cout<<"Input the element for adding in queue : ";

cin>>added\_item;

tmp->info = added\_item;

tmp->link=NULL;

if(front==NULL)

front=tmp;

else

rear->link=tmp;

rear=tmp;

}

}

void del()

{

struct node \*tmp;

if(front == NULL)

cout<<"Queue Underflow\n";

else

{

tmp=front;

cout<<"Deleted element is\t" <<tmp->info<<"\n";

front=front->link;

delete tmp;

}

}

void display()

{

struct node \*ptr;

ptr = front;

if(front == NULL)

cout<<"Queue is empty\n";

else

{

cout<<"Queue elements :\n";

while(ptr != NULL)

{

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

ptr = ptr->link;

}

cout<<"\n";

}

}

int main()

{

int choice;

while(1)

{

cout<<"1.Insert\n";

cout<<"2.Delete\n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

insert();

break;

case 2:

del();

break;

case 3:

display();

break;

case 4:

exit(1);

default :

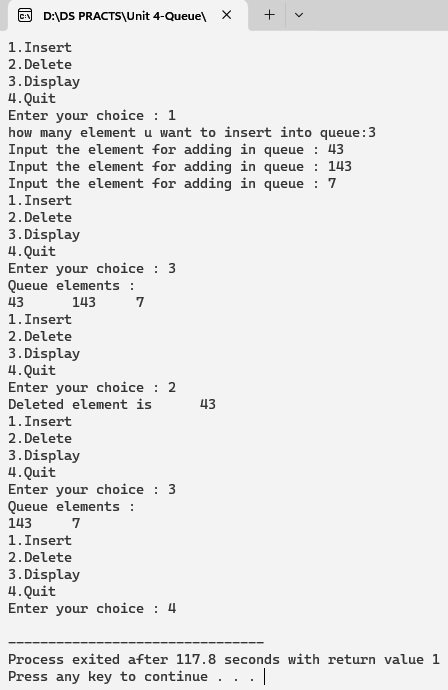
cout<<"Wrong choice\n";

}

}

}

**Output:**

****

**Practical No 18**

**Aim:** Write a Program to demonstrate Circular Queue using Array Implementation.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

struct node

{

int info;

struct node \*link;

}\*rear=NULL;

void insert()

{

int data,no;

struct node \*q,\*tmp;

cout<<"How many element u want to insert:";

cin>>no;

for(int i=1; i<=no; i++)

{

cout<<"Enter the element for insertion : ";

cin>>data;

tmp= new node;

tmp->info = data;

if(rear == NULL)

{

rear = tmp;

tmp->link = rear;

}

else

{

tmp->link = rear->link;

rear->link = tmp;

rear = tmp;

}

}

}

void del()

{

struct node \*tmp,\*q;

if(rear==NULL)

{

cout<<"Queue underflow\n";

return;

}

if( rear->link == rear )

{

tmp = rear;

rear = NULL;

delete tmp;

return;

}

q=rear->link;

tmp=q;

rear->link = q->link;

cout<<"Deleted element is:\t"<<tmp->info<<"\n";;

delete tmp;

}

void display()

{

struct node \*q;

if(rear == NULL)

{

cout<<"Queue is empty\n";

return;

}

q = rear->link;

cout<<"Queue is :\n";

while(q != rear)

{

cout<<q->info<<"\n";

q = q->link;

}

cout<<rear->info<<"\n\n";

}

int main()

{

int choice;

while(1)

{

cout<<"1.Insert \n";

cout<<"2.Delete \n";

cout<<"3.Display\n";

cout<<"4.Quit\n";

cout<<"Enter your choice : ";

cin>>choice;

switch(choice)

{

case 1:

insert();

break;

case 2:

del();

break;

case 3:

display();

break;

case 4:

exit(0);

default:

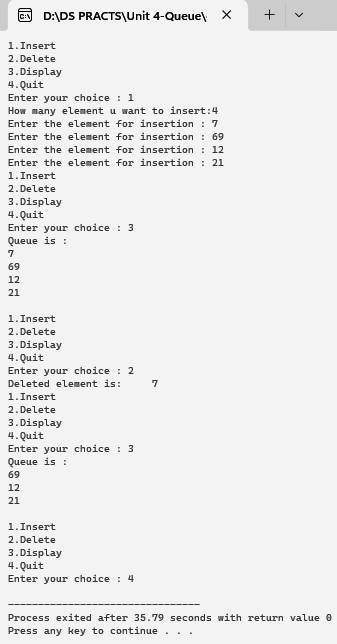
cout<<"Wrong choice\n";

}

}

}

**Output:**

****

**Practical No 19**

**Aim:** Write a Program to demonstrate Priority Queue using Linked List Implementation.

**Code:**

#include <iostream>

#include<conio.h>

#include <stdlib.h>

using namespace std;

class node

{

public:

int priority;

int data;

node \*next;

};

node \*front = NULL;

void insert()

{

node \*tmp,\*q;

int added\_item,item\_priority;

tmp = new node;

cout<<"input the item value to be added in the queue";

cin>>added\_item;

cout<<"enter it's priority";

cin>>item\_priority;

tmp->data = added\_item;

tmp->priority = item\_priority;

if(front == NULL || item\_priority<front->priority )

{

tmp->next = front;

front = tmp;

}

else

{

q = front;

while( q->next != NULL && q->next->priority <= item\_priority )

q=q->next;

tmp->next = q->next;

q->next = tmp;

}

}

void del()

{

node \*tmp;

if(front == NULL)

cout<<"queue underflow";

else

{

tmp=front;

cout<<"delected item is "<<tmp->data;

front = front->next;

delete tmp;

}

}

void display()

{

node \*ptr;

ptr = front;

if(front == NULL)

cout<<"queue is empty";

else

{

cout<<"queue is";

cout<<"priority";

while(ptr != NULL)

{

cout<<endl<<ptr->priority <<ptr->data;

ptr = ptr->next;

}

}

}

int main()

{

int choice;

while(1)

{

cout<<"\nmain menu";

cout<<"\n1.insert\n";

cout<<"2.delete\n";

cout<<"3.display\n";

cout<<"4.quit\n";

cout<<"enter your choice: ";

cin>>choice;

switch(choice)

{

case 1:

insert();

break;

case 2:

del();

break;

case 3:

display();

break;

case 4:

exit(1);

default:

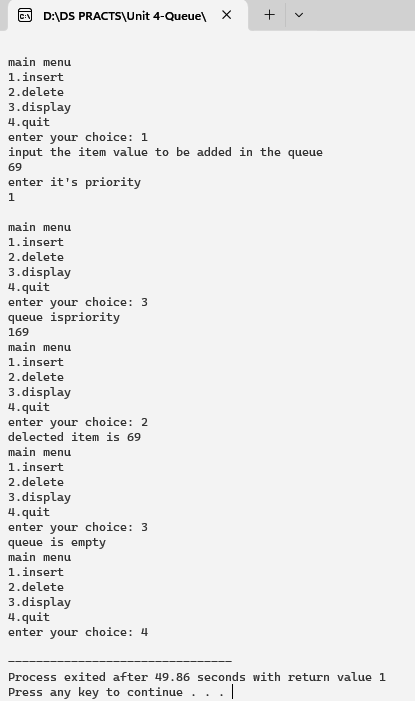
cout<<"wrong choice";

}

}

}

**Output:**

****

**Practical No 20**

**Aim:** Write a Program to demonstrate Double Ended Queue.

**Code:**

#include <iostream>

#include <cstdlib>

using namespace std;

struct node

{

int info;

node \*next;

node \*prev;

}\*head, \*tail;

class dqueue

{

public:

int top1, top2;

void insert();

void del();

void display();

dqueue()

{

top1 = 0;

top2 = 0;

head = NULL;

tail = NULL;

}

};

int main()

{

int choice;

dqueue dl;

while (1)

{

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

cout<<"Operations on Deque"<<endl;

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

cout<<"1.Insert Element into the Deque"<<endl;

cout<<"2.Delete Element from the Deque"<<endl;

cout<<"3.Traverse the Deque"<<endl;

cout<<"4.Quit"<<endl;

cout<<"Enter your Choice: ";

cin>>choice;

cout<<endl;

switch(choice)

{

case 1:

dl.insert();

break;

case 2:

dl.del();

break;

case 3:

dl.display();

break;

case 4:

exit(1);

break;

default:

cout<<"Wrong Choice"<<endl;

}

}

return 0;

}

void dqueue::insert()

{

struct node \*temp;

int ch, value;

if (top1 + top2 >= 50)

{

cout<<"Dequeue Overflow"<<endl;

return;

}

if (top1 + top2 == 0)

{

cout<<"Enter the value to be inserted: ";

cin>>value;

head = new (struct node);

head->info = value;

head->next = NULL;

head->prev = NULL;

tail = head;

top1++;

cout<<"Element Inserted into empty deque"<<endl;

}

else

{

while (1)

{

cout<<endl;

cout<<"1.Insert Element at first"<<endl;

cout<<"2.Insert Element at last"<<endl;

cout<<"3.Exit"<<endl;

cout<<endl;

cout<<"Enter Your Choice: ";

cin>>ch;

cout<<endl;

switch(ch)

{

case 1:

cout<<"Enter the value to be inserted: ";

cin>>value;

temp = new (struct node);

temp->info = value;

temp->next = head;

temp->prev = NULL;

head->prev = temp;

head = temp;

top1++;

break;

case 2:

cout<<"Enter the value to be inserted: ";

cin>>value;

temp = new (struct node);

temp->info = value;

temp->next = NULL;

temp->prev = tail;

tail->next = temp;

tail = temp;

top2++;

break;

case 3:

return;

break;

default:

cout<<"Wrong Choice"<<endl;

}

}

}

}

void dqueue::del()

{

if (top1 + top2 <= 0)

{

cout<<"Deque Underflow"<<endl;

return;

}

int ch;

while (1)

{

cout<<endl;

cout<<"1.Delete Element at first"<<endl;

cout<<"2.Delete Element at last"<<endl;

cout<<"3.Exit"<<endl;

cout<<endl;

cout<<"Enter Your Choice: ";

cin>>ch;

cout<<endl;

switch(ch)

{

case 1:

head = head->next;

head->prev = NULL;

top1--;

break;

case 2:

tail = tail->prev;

tail->next = NULL;

top2--;

break;

case 3:

return;

break;

default:

cout<<"Wrong Choice"<<endl;

}

}

}

void dqueue::display()

{

struct node \*temp;

int ch;

if (top1 + top2 <= 0)

{

cout<<"Deque Underflow"<<endl;

return;

}

while (1)

{

cout<<endl;

cout<<"1.Display Deque from Beginning"<<endl;

cout<<"2.Display Deque from End"<<endl;

cout<<"3.Exit"<<endl;

cout<<endl;

cout<<"Enter Your Choice: ";

cin>>ch;

cout<<endl;

switch (ch)

{

case 1:

temp = head;

cout<<"Deque from Beginning:"<<endl;

while (temp != NULL)

{

cout<<temp->info<<" ";

temp = temp->next;

}

cout<<endl;

break;

case 2:

cout<<"Deque from End:"<<endl;

temp = tail;

while (temp != NULL)

{

cout<<temp->info<<" ";

temp = temp->prev;

}

temp = tail;

cout<<endl;

break;

case 3:

return;

break;

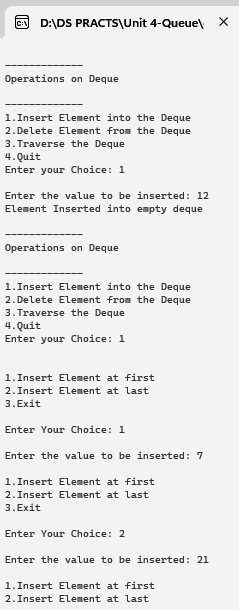
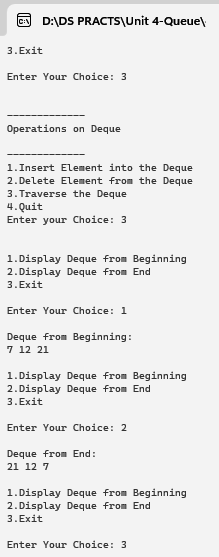
default:

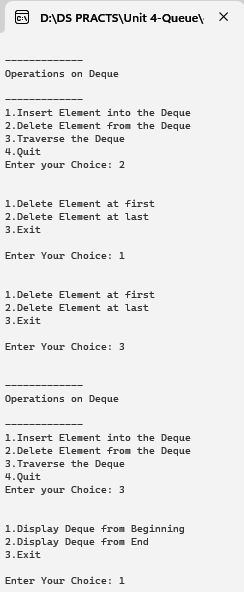
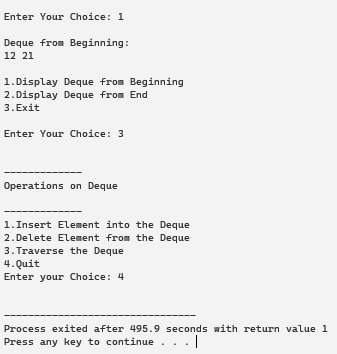
cout<<"Wrong Choice"<<endl;

}

} }

**Output:**

 ****

**** ****

**Linked List**

**Practical No 21**

**Aim:** Write a Program to demonstrate Singly Linked List. (Create, Display, Insert, Delete, Search, Count, Sort, Reverse)

**Code:**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

struct node

{

int data;

struct node \*link;

}\*start=NULL;

class linklist

{

public:

void create\_list();

void disp();

void inslast(int);

void insbeg(int);

void insnext(int,int);

void delbeg();

void delelement(int);

void dellast();

void count();

void rev();

void sorting();

int search(int);

};

void linklist::create\_list()

{

struct node \*q,\*tmp;

int d, n;

tmp= new node;

cout<<"Enter how many elements u want to enter";

cin>>n;

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

{

cout<<"Enter the element:";

cin>>d;

tmp= new node;

tmp->data=d;

tmp->link=NULL;

if(start==NULL)

start=tmp;

else

{

q=start;

while(q->link!=NULL)

{

q=q->link;

}

q->link=tmp;

}

}

return;

}

void linklist::disp()

{

node \*q;

q=start;

if(q==NULL)

{

cout<<" No data is in the list..";

}

cout<<" The items present in the list are :"<<endl;

while(q!=NULL)

{

cout<<" "<<q->data;

q=q->link;

cout<<endl;

}

}

void linklist:: insbeg(int x)

{

node \*q;

q=start;

start=new node;

start->data=x;

start->link=q;

cout<<" Inserted successfully at the begining.."<<endl;

disp();

}

void linklist:: insnext(int value,int position)

{

node \*temp,\*q;

int i;

q=start;

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

{

q=q->link;

if(q==NULL)

{

cout<<"there are less than "<<position << "elements";

return;

}

}

temp=new node;

temp->link=q->link;

temp->data=value;

q->link=temp;

}

void linklist::inslast(int x)

{

node \*q,\*t;

if(start==NULL)

{

start=new node;

start->data=x;

start->link=NULL;

}

else

{

q=start;

while(q->link!=NULL)

q=q->link;

t=new node;

t->data=x;

t->link=NULL;

q->link=t;

}

cout<<" Inserted successfully at the end.."<<endl;

disp();

}

void linklist::delbeg()

{

node \*tmp;

tmp=start;

start=start->link;

delete tmp;

return;

}

void linklist::delelement(int x)

{

node \*q,\*r;

q=start;

if(q->data==x)

{

start=q->link;

delete q;

return;

}

r=q;

while(q!=NULL)

{

if(q->data==x)

{

r->link=q->link;

delete q;

return;

}

r=q;

q=q->link;

}

cout<<" Element u entered"<<x<<"is not found.."<<endl;

}

void linklist::dellast()

{

cout<<" The list before deletion:"<<endl;

disp();

node \*q,\*t;

q=start;

if(q==NULL)

{

cout<<" There is no data in thelist.."<<endl;

return;

}

if(q->link==NULL)

{

start=q->link;

delete q;

return;

}

while(q->link->link!=NULL)

q=q->link;

q->link=NULL;

}

void linklist::count()

{

node \*q=start;

int cnt=0;

while(q!=NULL)

{

q=q->link;

cnt++;

}

cout<<"Number of elements are: "<<cnt<<endl;

}

void linklist::rev()

{

node \*p1,\*p2,\*p3;

if(start->link==NULL)

return;

p1=start;

p2=p1->link;

p3=p2->link;

p1->link=NULL;

p2->link=p1;

while(p3!=NULL)

{

p1=p2;

p2=p3;

p3=p3->link;

p2->link=p1;

}

start=p2;

}

int linklist::search(int value)

{

node \*temp;

temp=start;

int position=0;

while(temp!=NULL)

{

if(temp->data==value)

return position+1;

else

{

temp=temp->link;

position=position+1;

}

}

cout<<" Element "<<value<<" not found";

}

void linklist::sorting()

{

node \*ptr,\*s;

int value;

if(start == NULL)

{

cout<<"the list is empty"<<endl;

return;

}

ptr= start;

while (ptr!=NULL)

{

for (s=ptr->link; s!=NULL; s=s->link)

{

if (ptr->data > s->data)

{

value = ptr->data;

ptr->data = s->data;

s->data = value;

}

}

ptr = ptr->link;

}

}

int main()

{

linklist l;

linklist l1,l2,l3;

int ch,v,p,ps;

do

{

cout<<"\*\*\*\*\*Operations on SINGLY LINKED LIST\*\*\*\*\*"<<endl;

cout<<"1.Create linklist"<<endl;

cout<<"2.Insertion"<<endl;

cout<<"3.Deletion"<<endl;

cout<<"4.Display"<<endl;

cout<<"5.Count"<<endl;

cout<<"6.Reverse the link list"<<endl;

cout<<"7.Search"<<endl;

cout<<"8.Sorting"<<endl;

cout<<"9.Exit"<<endl;

cout<<" Enter ur choice:"<<endl;

cin>>ch;

switch(ch)

{

case 1:

l.create\_list();

l.disp();

break;

case 2:

cout<<"1.Insertion at begining"<<endl;

cout<<"2.Insertion at the end"<<endl;

cout<<"3.Insertion after the mentioned position"<<endl;

cin>>ps;

cout<<" Enter the value to insert:"<<endl;

cin>>v;

switch(ps)

{

case 1:l.insbeg(v);break;

case 2:l.inslast(v);break;

case 3:

cout<<" Enter the position to insert the value:"<<endl;

cin>>p;

l.insnext(v,p);

l.disp();

break;

default:

cout<<" The choice is invalid";

}

break;

case 3:

cout<<"1.Delete the first element "<<endl;

cout<<"2.Delete the last element"<<endl;

cout<<"3.Enter the element to delete from the list"<<endl;

cout<<" Enter ur choice:"<<endl;

cin>>ps;

switch(ps)

{

case 1:

l.delbeg();

cout<<" Thelist after deletion:"<<endl;

l.disp();

break;

case 2:

l.dellast();

cout<<" The list after deletion:"<<endl;

l.disp();

break;

case 3:

l.disp();

cout<<"Enter the element to delete :"<<endl;

cin>>v;

l.delelement(v);

cout<<" Thelist after deletion:"<<endl;

l.disp();

break;

default:

cout<<" The option is invalid...";

break;

}

break;

case 4:l.disp();break;

case 5:l.count();break;

case 6:

l.rev();

l.disp();

break;

case 7:

l.disp();

cout<<" Enter the element to search:"<<endl;

cin>>v;

cout<<" The position of the element "<< v<<"is"<<l.search(v);

getch();

break;

case 8:

cout<<"sorted link list:"<<endl;

l.sorting();

l.disp();

break;

case 9:exit(1);

default:

cout<<" The option is invalid...";

}

getch();

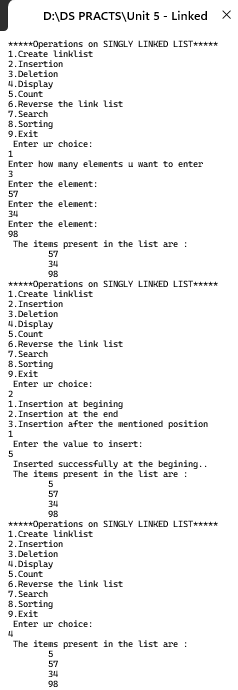
}

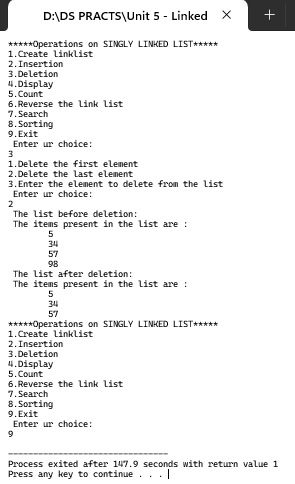
while(1);

getch();

}

**Output:**





**Practical No 22**

**Aim:** Write a Program to demonstrate Circular Linked List.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

struct cnode

{

int data;

struct cnode \*link;

struct cnode \*prev;

};

struct cnode \*start=NULL, \*last=NULL;

void create();

void insert();

void delete1();

void display();

void search();

void count();

void reverse();

void sort();

int main()

{

int choice;

cout<<"\*\*\*\*\*Operations on CIRCULAR LINKED LIST\*\*\*\*\*\n";

cout<<"1:Create\n 2:Insert\n 3.Delete\n 4:Display\n 5:Search\n 6:Count\n 7:Reverse\n 8:Sort\n 9.Exit\n";

cout<<"\nEnter Your choice\n";

cin>>choice;

while(choice)

{

switch(choice)

{

case 1:create();display();break;

case 2:insert();display();break;

case 3:delete1();break;

case 4:display();break;

case 5:search();break;

case 6:count();break;

case 7:reverse();display();break;

case 8:sort();display();break;

case 9:exit(1);

default : cout<<"\nInvalid Choice";break;

}

cout<<"\n 1:Create\n 2:Insert\n 3.Delete\n 4:Display\n 5:Search\n 6:Count\n 7:Reverse\n 8:Sort\n 9.Exit\n";

cout<<"\nEnter Your choice\n";

cin>>choice;

}

}

void create()

{

struct cnode \*tmp;

int no;

cout<<"How many node u want to enter\n";

cin>>no;

for(int i=1; i<=no;i++)

{

tmp=new cnode;

cout<<"Enter the item to be created: ";

cin>>tmp->data;

tmp->link=NULL;

if(start==NULL)

start=tmp;

else

last->link=tmp;

last=tmp;

last->link=start;

}

}

void display()

{

if(start==NULL)

cout<<"\ndata element to be displayed";

else

{

struct cnode \*temp=start;

cout<<"The element of a linked p are:\n";

do

{

cout<<temp->data<<"->\t";

temp=temp->link;

}

while(temp!=start);

}

}

void search()

{

if(start==NULL)

cout<<"\n!!! data Element !!!\n";

else

{

struct cnode \*temp=start;

int ele,posi=1;

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

cin>>ele;

while((temp!=last)&&(temp->data!=ele))

{

temp=temp->link;

posi++;

}

if(temp->data==ele)

cout<<"The element is present in the position :\n"<<posi;

else

cout<<"The element is datat present\n";

}

}

void insert()

{

struct cnode \*tmp,\*prev,\*cur,\*temp;

int count=1,posi,choice,length=1;

tmp = new cnode;

cout<<"\nEnter the data to be inserted : ";

cin>>tmp->data;

tmp->link=NULL;

prev=NULL; cur=start; temp=start;

cout<<"\nSelect the option of insertion";

cout<<"\n1.At the begin\n 2.At the end\n 3.In between : ";

cin>>choice;

switch(choice)

{

case 1:

tmp->link=start;

start=tmp;

last->link=start;

break;

case 2:

last->link=tmp;

last=tmp;

last->link=start;

break;

case 3:

while(temp!=last)

{

temp=temp->link;

length=length+1;

}

cout<<"\nEnter the position between 1st & "<<length<<"\t";

cin>>posi;

if((posi<2)||(posi>length))

{

cout<<"Invalid Position";

break;

}

while((count<posi)&&(cur!=NULL))

{

prev=cur;

cur=cur->link;

count++;

}

prev->link=tmp;

tmp->link=cur;

break;

default:

cout<<"\nInvalid Choice\n";

}

}

void count()

{

struct cnode \*temp=start;

int length=1;

while(temp!=last)

{

temp=temp->link;

length++;

}

cout<<"Total data. of cnode :"<<length;

}

void sort()

{

struct cnode \*s, \*ptr;

int temp;

if (last == NULL)

{

cout<<"List is empty, nothing to sort"<<endl;

return;

}

s = last->link;

while (s != last)

{

ptr = s->link;

while (ptr != last->link)

{

if(ptr != last->link)

{

if(s->data > ptr->data)

{

temp = s->data;

s->data = ptr->data;

ptr->data = temp;

}

}

else

break;

ptr = ptr->link;

}

s = s->link;

}

}

void reverse()

{

if (start == last && start == NULL)

{

cout<<"The List is empty, nothing to reverse"<<endl;

return;

}

struct cnode \*p1, \*p2;

p1 = start;

p2 = p1->link;

p1->link = NULL;

p1->prev = p2;

while (p2 != start)

{

p2->prev = p2->link;

p2->link = p1;

p1 = p2;

p2 = p2->prev;

}

last = start;

start = p1;

cout<<"List Reversed"<<endl;

}

void delete1()

{

struct cnode \*temp=start,\*prev=NULL,\*cur=start;

int choice,posi,count=1,item,length=1;

if(start==NULL)

cout<<"\ndata cnode";

else

{

cout<<"\nEnter the type of deletion:";

cout<<"\n1.Delete the start cnode\n 2.Delete the last cnode\n 3.Delete based on position\n 4.Delete all \n";

cin>>choice;

if(choice<1||choice>4)

cout<<"\nInvalid choice";

else

{

switch(choice)

{

case 1:

cout<<"The deleted element is "<<start->data;

start=start->link;

last->link=start;

break;

case 2:

cout<<"\nThe deleted element is"<<last->data;

if(start==last)

start=last=NULL;

else

{

while(temp!=last)

{

prev=temp;

temp=temp->link;

}

prev->link=start;

last=prev;

}

break;

case 3:

while(temp!=last)

{

temp=temp->link;

length++;

}

cout<<"\nEnter the position of deletion : ";

cin>>posi;

if((posi<1)||(posi>length))

{

cout<<"\nInvalid Position";

break;

}

else

{

while(count<posi)

{

prev=cur;

cur=cur->link;

count++;

}

cout<<"\nThe deleted element is"<<cur->data;

cur=cur->link;

prev->link=cur;

break;

}

case 4:

start=last=NULL;

cout<<"\nAll elements have been deleted";

break;

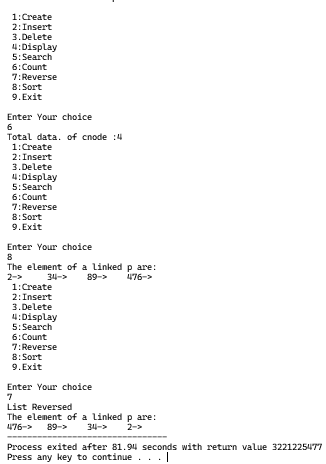
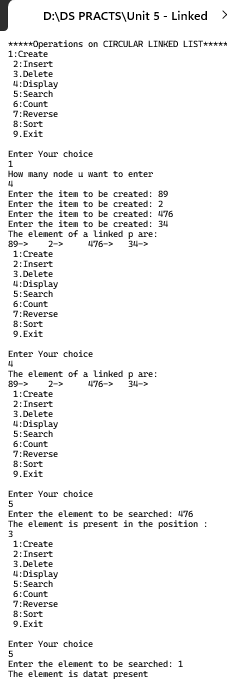
}

}

}

}

**Output:**

****

**Practical No 23**

**Aim:** Write a Program to demonstrate Doubly Linked List. (Create, Display, Insert, Delete, Search, Count, Sort, Reverse)

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

struct dnode

{

int data;

struct dnode \*prev;

struct dnode \*next;

};

struct dnode \*start=NULL, \*last=NULL;

void create();

void insert();

void display();

void delete1();

void search();

void count();

void rev();

int main()

{

int choice;

cout<<"1:Create\n 2:Insert\n 3.Display\n 4.Delete\n 5.Search\n 6.Count\n 7.Reverse\n 8.Exit\n";

cout<<"Enter Your choice:";

cin>>choice;

while(choice)

{

switch(choice)

{

case 1:create();display();break;

case 2:insert();display();break;

case 3:display();break;

case 4:delete1();display();break;

case 5:search();break;

case 6:count();break;

case 7:rev();break;

case 8:exit(1);break;

default : cout<<"\nInvalid Choice";break;

}

cout<<"\nEnter Your choice\n";

cout<<"1:Create\n2:Insert\n3.Display\n4.Delete\n5.Search\n6.Count\n7.Reverse\n 8.Exit\n";

cout<<"\nEnter Your choice:";

cin>>choice;

}

}

void create()

{

struct dnode \*tmp;

int no;

cout<<"How many element u want to insert\t";

cin>>no;

for(int i=1;i<=no;i++)

{

tmp=new dnode;

cout<<"Enter the item to be created: ";

cin>>tmp->data;

tmp->prev=NULL;

tmp->next=NULL;

if(start==NULL)

start=tmp;

else

{

last->next=tmp;

tmp->prev=last;

}

last=tmp;

}

}

void display()

{

if(start==NULL)

cout<<"\nNo element to be displayed";

else

{

struct dnode \*temp=start;

cout<<"\nThe element of a doubly linked list are:\n";

while(temp!=NULL)

{

cout<<temp->data<<"->\t";

temp=temp->next;

}

}

}

void search()

{

if(start==NULL)

cout<<"\n!!! No Element !!!\n";

else

{

struct dnode \*temp=start;

int ele,posi=1;

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

cin>>ele;

while((temp!=last)&&(temp->data!=ele))

{

temp=temp->next;

posi++;

}

if(temp->data==ele)

cout<<"The element is present in the position : \n"<<posi;

else

cout<<"The element is not present\n";

}

}

void insert()

{

struct dnode \*tmp,\*q,\*cur,\*temp;

int count=1,posi,choice,length=1;

q=NULL; cur=start; temp=start;

tmp = new dnode;

cout<<"\nEnter the data to be inserted : ";

cin>>tmp->data;

tmp->prev=NULL;

tmp->next=NULL;

cout<<"\nSelect the option of insertion";

cout<<"\n1.At the begin\n2.At the end\n3.In between :\n ";

cin>>choice;

switch(choice)

{

case 1:

tmp->next=start;

start->prev=tmp;

start=tmp;

break;

case 2:

last->next=tmp;

tmp->prev=last;

last=tmp;

break;

case 3:

while(temp!=last)

{

temp=temp->next;

length=length+1;

}

cout<<"\nEnter the position between 1st &"<<length;

cin>>posi;

if((posi<1)||(posi>length))

{

cout<<"Invalid Position";

break;

}

else

{

while((count<posi)&&(cur!=NULL))

{

q=cur;

cur=cur->next;

count++;

}

q->next=tmp;

tmp->prev=q;

tmp->next=cur;

cur->prev=tmp;

break;

}

default:

cout<<"\nInvalid Choice\n";

}

}

void delete1()

{

struct dnode

\*temp=start,\*q=NULL,\*cur=start;

int choice,posi,count=1,item,length=1;

if(start==NULL)

cout<<"\nNo node";

else

{

cout<<"\nEnter the type of deletion:\n";

cout<<"\n1.Delete the start node\n 2.Delete the last node\n 3.Delete based on position\n 4.Delete all\n ";

cin>>choice;

if(choice<1||choice>4)

cout<<"\nInvalid choice";

else

{

switch(choice)

{

case 1:

cout<<"The deleted element is"<<start->data;

start=start->next;

start->prev=NULL;

break;

case 2:

cout<<"\nThe deleted element is:\n"<<last->data;

last=last->prev;

last->next=NULL;

break;

case 3:

while(temp!=last)

{

temp=temp->next;

length++;

}

cout<<"\nEnter the position of deletion : ";

cin>>posi;

if((posi<1)||(posi>length))

{

cout<<"\nInvalid Position";

break;

}

else

{

while((count<posi)&&(cur!=NULL))

{

q=cur;

cur=cur->next;

count++;

}

cout<<"\nThe deleted element is:\n"<<cur->data;

cur=cur->next;

q->next=cur;

cur->prev=q;

break;

}

case 4:

start=last=NULL;

cout<<"\nAll elements have been deleted";

break;

}

}

}

}

void rev()

{

struct dnode \*p1,\*p2;

p1=start;

p2=p1->next;

p1->next=NULL;

p1->prev=p2;

while(p2!=NULL)

{

p2->prev=p2->next;

p2->next=p1;

p1=p2;

p2=p2->prev;

}

start=p1;

display();

}

void count()

{

struct dnode \*q=start;

int cnt=0;

while(q!=NULL)

{

q=q->next;

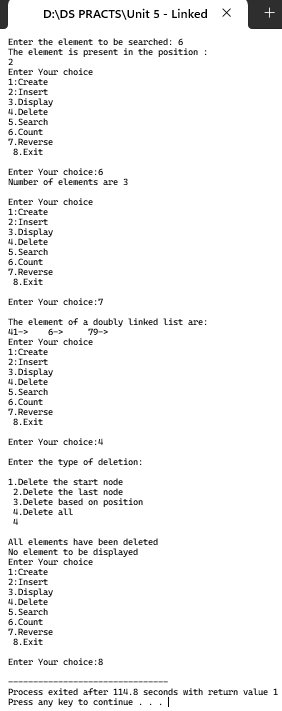
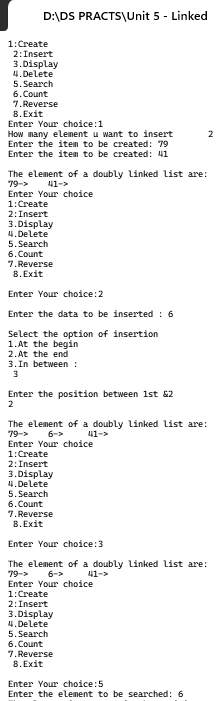
cnt++;

}

cout<<"Number of elements are "<<cnt<<"\n";

}

**Output:**

****

**Practical No 24**

**Aim:** Write a Program to demonstrate Application of a Linked List – Polynomial Addition.

**Code:**

#include <bits/stdc++.h>

using namespace std;

// Represents the single polynomial

class Node{

public:

int x;

int y;

Node \*next;

Node(int x, int y, Node \*next){

this->x = x;

this->y = y;

this->next = next;

}

}; Node \*result = NULL;

// Simply Prints the List to the Screen

void printList(Node \*head){

while(head!=NULL){

cout<<head->x<<"x"<<"^"<<head->y << " ";

head = head->next;

}

cout << endl;

}

// Creates a new node and inserts it into the end

// of the head node

void insertNode(Node \*head, int x, int y){

Node \*newNode = new Node(x, y, NULL);

while(head->next != NULL){

head = head->next;

}

head->next = newNode;

}

// Function to add both list.

void add(Node \*first, Node \*second){

// Resultant list

Node \*result = NULL;

// Loop to add both the list

// until one of the list reaches to its end

while(first && second){

// saves current result

int x, y;

// if power of both the polynomial is same

// we simply add their coefficient and increment the

// pointer to next node

if(first->y == second->y){

x = first->x+second->x;

y = first->y;

first = first->next;

second = second->next;

}

// if power of first polynomial is greater than

// the second one then save its coefficient and power

// into the result and increment its pointer

else if(first->y > second->y){

x = first->x;

y = first->y;

first = first->next;

}

// if power of second polynomial is greater than

// the first one. then save its coefficient and power

// into the result and increment its pointer

else{

x = second->x;

y = second->y;

second = second->next;

}

// if resultant list is empty we create a new node

// else we simply add the value at the end of our resultant list

if(result == NULL) result = new Node(x, y, NULL);

else insertNode(result, x, y);

}

// After completion of the above loop there might be a possibility

// that one of the two polynomial lists have some unchecked data

// below two loops are just adding the remaining data into our

// resultant list

while(first){

if(result == NULL) result = new Node(first->x, first->y, NULL);

else insertNode(result, first->x, first->y);

first = first->next;

}

while(second){

if(result == NULL) result = new Node(second->x, second->y, NULL);

else insertNode(result, second->x, second->y);

second = second->next;

}

// printing the resultant list

printList(result);

}

int main(){

// First polynomial

Node \*head = new Node(5, 2, NULL);

insertNode(head, 4, 1);

// second polynomial

Node \*head2 = new Node(5, 2, NULL);

insertNode(head2, 2, 1);

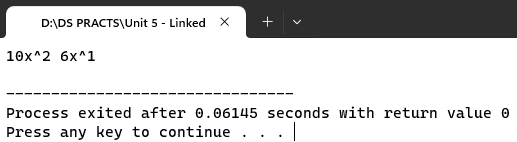
// our algorithm

add(head, head2);

return 0;

}

**Output:**

****

**Practical No 25**

**Aim:** Write a Program to demonstrate Application of a Linked List – Sparse Matrix.

**Code:**

#include<iostream>

using namespace std;

class Node

{

public:

int row;

int col;

int data;

Node \*next;

};

void create\_new\_node(Node \*\*p, int row\_index,

int col\_index, int x)

{

Node \*temp = \*p;

Node \*r;

if (temp == NULL)

{

temp = new Node();

temp->row = row\_index;

temp->col = col\_index;

temp->data = x;

temp->next = NULL;

\*p = temp;

}

else

{

while (temp->next != NULL)

temp = temp->next;

r = new Node();

r->row = row\_index;

r->col = col\_index;

r->data = x;

r->next = NULL;

temp->next = r;

}

}

void printList(Node \*start)

{

Node \*ptr = start;

cout << "row\_position:";

while (ptr != NULL)

{

cout << ptr->row << " ";

ptr = ptr->next;

}

cout << endl;

cout << "column\_position:";

ptr = start;

while (ptr != NULL)

{

cout << ptr->col << " ";

ptr = ptr->next;

}

cout << endl;

cout << "Value:";

ptr = start;

while (ptr != NULL)

{

cout << ptr->data << " ";

ptr = ptr->next;

}

}

int main()

{

int sparseMatrix[4][5] = { { 0 , 0 , 3 , 0 , 4 },

{ 0 , 0 , 5 , 7 , 0 },

{ 0 , 0 , 0 , 0 , 0 },

{ 0 , 2 , 6 , 0 , 0 } };

Node \*first = NULL;

for(int i = 0; i < 4; i++)

{

for(int j = 0; j < 5; j++)

{

if (sparseMatrix[i][j] != 0)

create\_new\_node(&first, i, j,

sparseMatrix[i][j]);

}

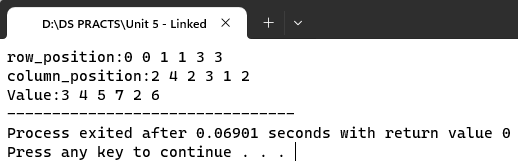
}

printList(first);

return 0;

}

**Output:**

****

**Trees**

**Practical No 26**

**Aim:** Write a Program to demonstrate Binary Search Tree (Create, Preorder, Inorder, Postorder, Search Largest Node, Search Smallest Node, Delete, Count number of Nodes).

**Code:**

#include<iostream>

#include<process.h>

#include<conio.h>

using namespace std;

struct node

{

int data;

struct node \*left;

struct node \*right;

};

class BST

{

public:

node \*tree;

BST()

{

tree=NULL;

}

void insertNodeTree(node\*\*,int item);

void preorder(node \*);

void inorder(node \*);

void postorder(node \*);

int totalNodes(node \*);

void removeTree(node \*\*);

void findsmallestNode(node \*);

void findLargestNode(node \*);

void deleteNode(int);

};

//Create tree

void BST::insertNodeTree(node \*\*tree,int item)

{

if (\*tree==NULL)

{

\*tree=new node;

(\*tree)->data=item;

(\*tree)->left=NULL;

(\*tree)->right=NULL;

}

else

{

if((\*tree)->data>item)

insertNodeTree(&((\*tree)->left),item);

else

insertNodeTree(&((\*tree)->right),item);

}

}

// Display- Preorder(Root-Left-Right)

void BST::preorder(node\*tree)

{

if(tree!=NULL)

{

cout<<" "<<tree->data;

preorder(tree->left);

preorder(tree->right);

}

}

// Display- Inorder(Left-Root-Right)

void BST::inorder(node\*tree)

{

if(tree!=NULL)

{

inorder(tree->left);

cout<<" "<<tree->data;

inorder(tree->right);

}

}

// Display- Postorder(Left-Right-Root)

void BST::postorder(node\*tree)

{

if(tree!=NULL)

{

postorder(tree->left);

postorder(tree->right);

cout<<" "<<tree->data;

}

}

//Count num

int BST::totalNodes(node \*tree)

{

if(tree==NULL)

return 0;

else

return(totalNodes(tree->left)+totalNodes(tree->right)+1);

}

//Smallest Node

void BST::findsmallestNode(node \*tree)

{

if(tree==NULL||tree->left==NULL)

cout<<tree->data;

else

findsmallestNode(tree->left);

}

//Largest Node

void BST::findLargestNode(node \*tree)

{

if(tree==NULL||tree->right==NULL)

cout<<tree->data;

else

findLargestNode(tree->right);

}

//Remove tree

void BST::removeTree(node \*\*tree)

{

if((\*tree)!=NULL)

{

removeTree(&(\*tree)->left);

removeTree(&(\*tree)->right);

delete(\*tree);

}

}

//Delete Node

node \*find\_Insucc(node \*curr)

{

node \*succ=curr->right;

if(succ!=NULL)

{

while(succ->left!=NULL)

succ=succ->left;

}

return(succ);

}

void BST::deleteNode(int item)

{

node \*curr=tree,\*succ,\*pred;

int flag=0,delcase;

while(curr!=NULL && flag!=1)

{

if(item<curr->data)

{

pred=curr;

curr=curr->left;

}

else if(item>curr->data)

{

pred=curr;

curr=curr->right;

}

else

{

flag=1;

}

}

if(flag==0)

{

cout<<"\n item does not exist:no deletion\n";

getch();

}

if(curr->left==NULL && curr->right==NULL)

delcase=1;

else if(curr->left!=NULL && curr->right!=NULL)

delcase=3;

else

delcase=2;

if(delcase==1)

{

if(pred->left==curr)

pred->left=NULL;

else

pred->right=NULL;

delete(curr);

//pred->right;

}

if(delcase==2)

{

if(pred->left==curr)

{

if(curr->left==NULL)

pred->left=curr->right;

else

pred->left=curr->left;

}

else

{

if(curr->left==NULL)

pred->right=curr->right;

else

pred->right=curr->left;

} delete(curr);

}

if(delcase==3)

{

succ=find\_Insucc(curr);

int item1=succ->data;

deleteNode(item1);

curr->data=item1;

}

}

int main()

{

BST obj;

int choice;

int height=0,total=0,n,item;

//node \*\*tmp;

while(1)

{

cout<<"\n\*\*\*\*\*BINARY SEARCH TREE Operations\*\*\*\*\*\n";

cout<<"1.Create Tree \n";

cout<<"2.Traversal \n";

cout<<"3.Total Nodes\n";

cout<<"4.Insert Nodes\n";

cout<<"5.Remove Tree\n";

cout<<"6.Find Smallest Nodes \n";

cout<<"7.Find Largest Node \n";

cout<<"8.Delete Node\n";

cout<<"9.Exit\n";

cout<<"Enter your choice :";

cin>>choice;

switch(choice)

{

case 1:

cout<<"\n Creating Tree----";

cout<<"how many nodes u want to enter :";

cin>>n;

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

{

cout<<"Enter Values :";

cin>>item;

obj.insertNodeTree(&obj.tree,item);

}

break;

case 2:

cout<<"\n Inorder Traversal :";

obj.inorder(obj.tree);

cout<<"\n preorder Traversal :";

obj.preorder(obj.tree);

cout<<"\n postorder Traversal :";

obj.postorder(obj.tree);

getch();

break;

case 3:

total=obj.totalNodes(obj.tree);

cout<<"Total nodes :"<<total;

getch();

break;

case 4:

cout<<"\n insert node in a tree \n";

cout<<"enter value :";

cin>>item;

obj.insertNodeTree(&obj.tree,item);

cout<<"\nitem is inserted\n";

getch();

break;

case 5:

obj.removeTree(&obj.tree);

cout<<"\n tree is removed from memory";

getch();

break;

case 6:

cout<<"\n\nsmallest node is:";

obj.findsmallestNode(obj.tree);

getch();

break;

case 7:

cout<<"\n\nlargest node is:";

obj.findLargestNode(obj.tree);

getch();

break;

case 8:

cout<<"\n\n Deleting a node from a tree--";

cout<<"enter value";

cin>>item;

obj.deleteNode(item);

break;

case 9:

exit(1);

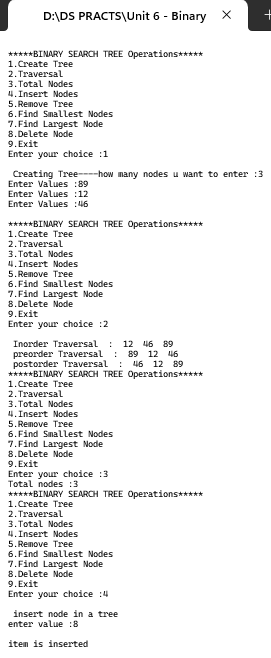
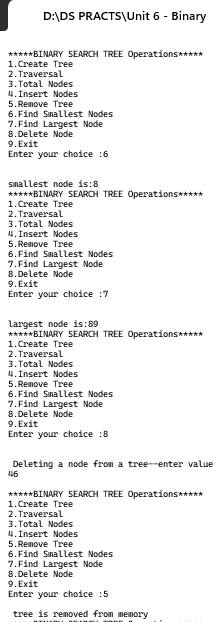
break;

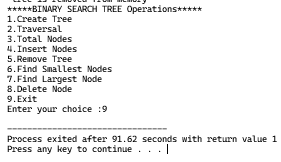
}

}

}

**Output:**

**** ****



**Practical No 27**

**Aim:** Write a Program to demonstrate MinHeap.

**Code:**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#define SIZE 15

using namespace std;

class Heap

{

public:

int last;

int a[SIZE];

Heap();

void get();

void buildHeap(int \*);

void reheapup(int \*,int);

void insert(int);

void delete\_node(int\*,int\*);

void reheapdown(int \*,int, int);

void display();

};

Heap::Heap()

{

last =0;

for(int i=0;i<SIZE;i++)

a[i]=0;

}

void Heap::get()

{

cout<<"\nEnter size of the heap:";

cin>>last;

for(int i=0;i<last;i++)

{

cout<<"\nEnter the elements: ";

cin>>a[i];

}

buildHeap(a);

}

void Heap::buildHeap(int \*a)

{

int walk,i;

walk=0;

while(walk<last)

{

reheapup(a,walk);

walk=walk+1;

}

}

void Heap::reheapup(int \*a,int h)

{

int parent,hold;

if(h)

{

parent=(h-1)/2;

if(a[h]<a[parent])

{

hold=a[parent];

a[parent]=a[h];

a[h]=hold;

reheapup(a,parent);

}

}

}

void Heap::insert(int item)

{

if(last+1==SIZE)

{

cout<<"\nHeap full";

}

else

{

a[last]=item;

reheapup(a,last);

last=last+1;

cout<<"\nItem is inserted successfully";

}

}

void Heap::delete\_node(int \*a,int \*last)

{

int item;

if(\*last<=0)

{

cout<<"\nCan't delete";

}

else

{

item=a[0];

a[0]=a[\*last];

reheapdown(a,0,\*last);

\*last=\*last-1;

cout<<"\nThe deleted item is: "<<item;

}

}

void Heap::reheapdown(int \*a,int root,int last)

{

int hold,parent,lkey,rkey,largercindex;

if((root\*2+1)<=last)

{

lkey=a[root\*2+1];

if((root\*2+2)<=last)

{

rkey=a[root\*2+2];

}

else

rkey=-1;

if(lkey>rkey)

{

largercindex=root\*2+1;

}

else

{

largercindex=root\*2+2;

}

if(a[root]>=a[largercindex])

{

hold=a[root];

a[root]=a[largercindex];

a[largercindex]=hold;

reheapdown(a,largercindex,last);

}

}

}

void Heap::display()

{

cout<<"\nCount: "<<last;

cout<<"\nThe a constructed is: \n";

for(int i=0;i<last;i++)

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

}

int main()

{

int i,ch,item;

Heap h;

h.get();

while(1)

{

cout<<"\n1.Insert node\n2.Delete node\n3.Display\n4.Quit\nEnter your choice";

cin>>ch;

switch(ch)

{

case 1:

cout<<"\nEnter element to be inserted: ";

cin>>item;

h.insert(item);

break;

case 2:

h.delete\_node(h.a,&h.last);

break;

case 3:

h.display();

break;

case 4:

exit(1);

break;

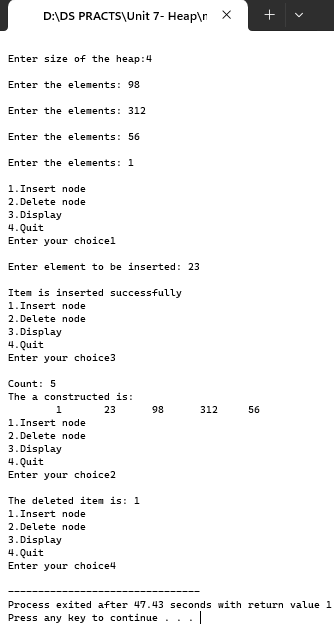
}

}

getch();

}

**Output:**

****

**Practical No 28**

**Aim:** Write a Program to demonstrate MaxHeap.

**Code:**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

#define SIZE 15

using namespace std;

class Heap

{

public:

int last;

int a[SIZE];

Heap();

void get();

void buildHeap(int \*);

void reheapup(int \*,int);

void insert(int);

void delete\_node(int\*,int\*);

void reheapdown(int \*,int, int);

void display();

};

Heap::Heap()

{

last =0;

for(int i=0;i<SIZE;i++)

a[i]=0;

}

void Heap::get()

{

cout<<"\nEnter size of the heap:";

cin>>last;

for(int i=0;i<last;i++)

{

cout<<"\nEnter the elements: ";

cin>>a[i];

}

buildHeap(a);

}

void Heap::buildHeap(int \*a)

{

int walk,i;

walk=0;

while(walk<last)

{

reheapup(a,walk);

walk=walk+1;

}

}

void Heap::reheapup(int \*a,int h)

{

int parent,hold;

if(h)

{

parent=(h-1)/2;

if(a[h]>a[parent])

{

hold=a[parent];

a[parent]=a[h];

a[h]=hold;

reheapup(a,parent);

}

}

}

void Heap::insert(int item)

{

if(last+1==SIZE)

{

cout<<"\nHeap full";

}

else

{

a[last]=item;

reheapup(a,last);

last=last+1;

cout<<"\nItem is inserted successfully";

}

}

void Heap::delete\_node(int \*a,int \*last)

{

int item;

if(\*last<=0)

{

cout<<"\nCan't delete";

}

else

{

item=a[0];

a[0]=a[\*last];

reheapdown(a,0,\*last);

\*last=\*last-1;

cout<<"\nThe deleted item is: "<<item;

}

}

void Heap::reheapdown(int \*a,int root,int last)

{

int hold,parent,lkey,rkey,largercindex;

if((root\*2+1)<=last)

{

lkey=a[root\*2+1];

if((root\*2+2)<=last)

{

rkey=a[root\*2+2];

}

else

rkey=-1;

if(lkey>rkey)

{

largercindex=root\*2+1;

}

else

{

largercindex=root\*2+2;

}

if(a[root]>=a[largercindex])

{

hold=a[root];

a[root]=a[largercindex];

a[largercindex]=hold;

reheapdown(a,largercindex,last);

}

}

}

void Heap::display()

{

cout<<"\nCount: "<<last;

cout<<"\nThe a constructed is: \n";

for(int i=0;i<last;i++)

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

}

int main()

{

int i,ch,item;

Heap h;

h.get();

while(1)

{

cout<<"\n1.Insert node\n2.Delete node\n3.Display\n4.Quit\nEnter your choice";

cin>>ch;

switch(ch)

{

case 1:

cout<<"\nEnter element to be inserted: ";

cin>>item;

h.insert(item);

break;

case 2:

h.delete\_node(h.a,&h.last);

break;

case 3:

h.display();

break;

case 4:

exit(1);

break;

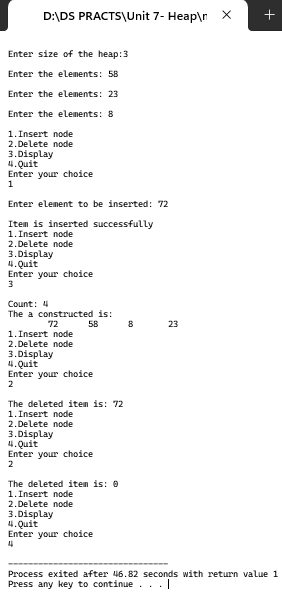
}

}

getch();

}

**Output:**

****

**Graphs**

**Practical No 29**

**Aim:** Write a Program to demonstrate a Graph using Adjacency Matrix.

**Code:**

#include<iostream>

#include<conio.h>

#define SIZE 20

using namespace std;

int main()

{

int vertex[SIZE];

int edge[SIZE][SIZE];

int i,j,k,no\_edges,no\_vertex,from,to;

cout<<"enter number of vertices\n";

cin>>no\_vertex;

cout<<"enter vertices: "<<endl;

for(i=0;i<no\_vertex;i++)

cin>>vertex[i];

cout<<"enter number of edges\n";

cin>>no\_edges;

for(i=1;i<=no\_vertex;i++)

for(j=1;j<=no\_vertex;j++)

edge[i][j]=0;

for(i=1;i<=no\_edges;i++)

{

cout<<"Enter from vertex\n";

cin>>from;

cout<<"Enter to vertex\n";

cin>>to;

edge[from][to]=1;

edge[to][from]=1;

}

cout<<"\n\nOutput:\n";

cout<<"The vertices in graph are:\n"<<endl;

for(i=0;i<no\_vertex;i++)

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

cout<<"\n Adjecancy matrix for a graph is\n";

for(i=1;i<=no\_vertex;i++)

{

for(j=1;j<=no\_vertex;j++)

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

cout<<"\n";

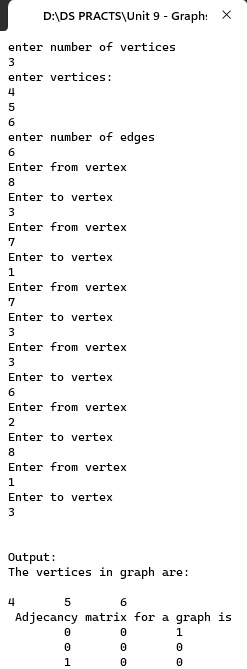
}

getch();

return 0;

}

**Output:**

****

**Practical No 30**

**Aim:** Write a Program to find the Minimum Spanning Tree using Prim’s Algorithm.

**Code:**

#include<iostream>

#include<conio.h>

using namespace std;

int main()

{

int a,b,u,v,n,i,j,ne=1;

int visited[10]={0},min,mincost=0,cost[10][10];

cout<<"\nEnter the number of nodes:";

cin>>n;

cout<<"\nEnter the adjacency matrix:\n";

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

{

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

{

cin>>cost[i][j];

if(cost[i][j]==0)

cost[i][j]=999;

}

}

visited[1]=1;

cout<<"\n";

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

{

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

{

if(cost[i][j]< min)

{

if(visited[i]!=0)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

}

if(visited[u]==0 || visited[v]==0)

{

cout<<ne++<<" edges"<<"\t"<<"("<<a<<","<<b<<")"<<"="<<"\t"<<min<<"\n";

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

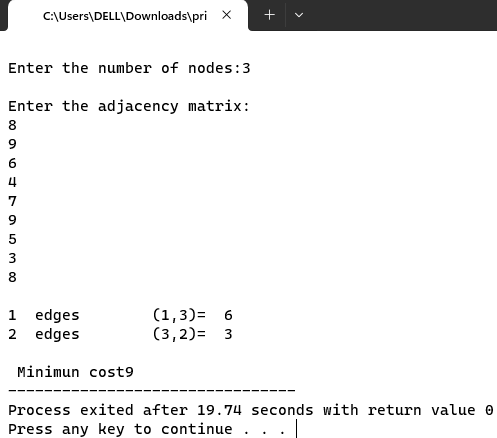
}

cout<<"\n Minimun cost"<<mincost;

getch();

}

**Output:**

****