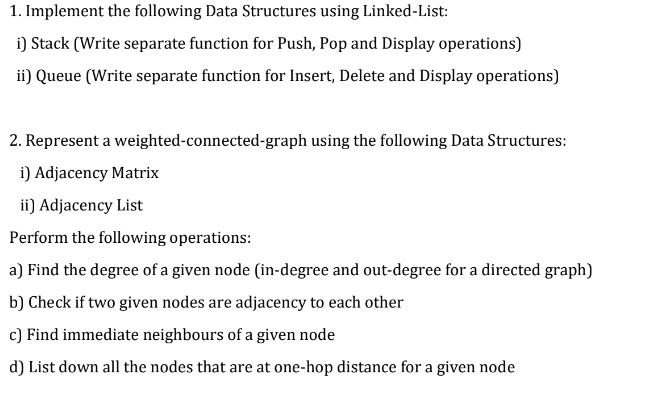
***Practice Programming Assignments***

***D.V.Harsha Vardhan Reddy***

***CSE-2;210C2030061***



STACK

CODE:

#include <iostream>

using namespace std;

struct Node

{

int data;

Node\* link;

};

Node\* top;

void push(int data)

{

Node\* temp = new Node();

temp->data = data;

temp->link = top;

top = temp;

}

void pop()

{

Node\* temp;

if (top == NULL)

{

cout << "\nStack Underflow" << endl;

exit(1);

}

else

{

temp = top;

top = top->link;

free(temp);

}

}

void display()

{

Node\* temp;

if (top == NULL)

{

cout << "\nStack Underflow";

exit(1);

}

else

{

temp = top;

while (temp != NULL)

{

cout << temp->data ;

cout<<"-> ";

temp = temp->link;

}

}

cout<<endl;

}

int main()

{

push(243);

push(98);

push(321);

push(256);

push(564);

display();

pop();

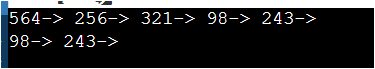
pop();

pop();

display();

}

OUTPUT:



QUEUE:

#include <iostream>

using namespace std;

struct Queuenode {

int data;

Queuenode\* next;

Queuenode(int d)

{

data = d;

next = NULL;

}

};

struct Queue {

Queuenode \*front, \*rear;

Queue()

{

front = rear = NULL;

}

void enQueue(int x)

{

Queuenode\* temp = new Queuenode(x);

if (rear == NULL) {

front = rear = temp;

return;

}

rear->next = temp;

rear = temp;

}

void deQueue()

{

if (front == NULL)

return;

Queuenode\* temp = front;

front = front->next;

if (front == NULL)

rear = NULL;

delete (temp);

}

void display()

{

while(front!=NULL)

{

cout<<front->data<<" ";

front=front->next;

}

}

};

int main()

{

Queue q;

q.enQueue(654);

q.enQueue(908);

q.enQueue(123);

q.enQueue(543);

q.deQueue();

q.deQueue();

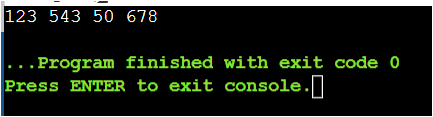
q.enQueue(50);

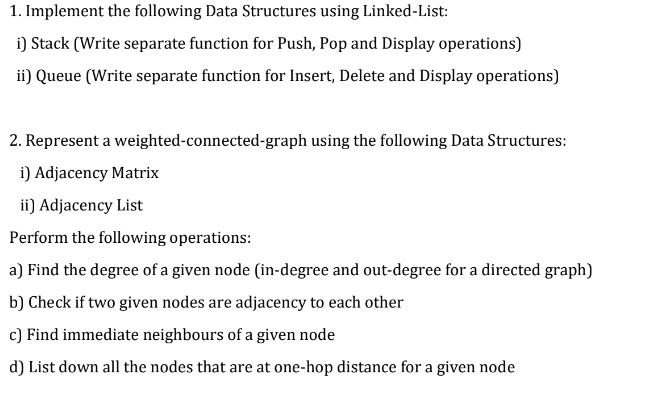
q.enQueue(678);

q.display();

}

OUTPUT:





CODE:

#include<iostream>

using namespace std;

int arr[5][5];

void insert(int u,int v)

{

arr[u][v]=1;

}

void displayMatrix()

{

int i, j;

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

{

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

{

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

}

cout << endl;

}

}

void degree()

{

cout<<"\n1)Degree\n";

int u,count=0;

cout<<"\nEnter node: ";

cin>>u;

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

{

if(arr[u][i]==1)

count++;

}

cout<<"Degree of node "<<u<< " is "<<count<<endl;

}

void adjacency\_nodes()

{

cout<<"\n2)Adjacency node\n";

int u,v;

cout<<"\nEnter first node: ";

cin>>u;

cout<<"\nEnter second node: ";

cin>>v;

if(arr[u][v]==1)

cout<<"Node "<<u <<" and "<<v<< " are adjacency to each other";

else

cout<<"Node "<<u <<" and "<<v<< " are not adjacency to each other";

}

void immediate\_neighbours()

{

int n;

cout<<"\n\n3)Immediate Neighbours\n";

cout<<"\nEnter Node: ";

cin>>n;

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

{

if(arr[n][i]==1)

cout<<i<<" ";

}

}

void one\_hop()

{

int n,i,j;

cout<<"\n\n4)one-hop distance \n";

cout<<"\nEnter Node: ";

cin>>n;

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

{

if(arr[n][i]==1)

break;

}

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

{

if(arr[i][j]==1)

break;

}

if(j!=5)

cout<<"one-hop distance for "<<n<<" is "<<j;

}

int main()

{

insert(0,0);

insert(0,3);

insert(0,4);

insert(1,2);

insert(1,4);

insert(4,0);

insert(1,2);

insert(2,3);

insert(3,4);

insert(2,4);

insert(3,3);

insert(4,4);

displayMatrix();

degree();

adjacency\_nodes();

immediate\_neighbours();

one\_hop();

}

OUTPUT:

