Practical – 13

AIM: Implement Queue using Linked List A Query Q is of 2 Types

(i) x (a query of this type means pushing 'x' into the queue)

(ii) 2 (a query of this type means to pop an element

from the queue and print the popped element)

* Program

#include<bits/stdc++.h>

using namespace std;

struct node{

int info;

struct node\* link;

};

struct node\* front = NULL;

struct node\* rear = NULL;

struct node\* create(int);

void enqueue(int);

int dequeue();

void display();

int main(){

int choice;

do{

cout << "1. Insert element in queue." << endl;

cout << "2. Delete element in queue." << endl;

cout << "3. Display" << endl;

cout << "4. exit" << endl;

cout << "Select the option : " ;

cin >> choice;

switch (choice)

{

case 1:

int x;

cout << "Enter a element you want to add : ";

cin >> x;

enqueue(x);

display();

break;

case 2:

int y;

y = dequeue();

cout << y << endl;

break;

case 3:

display();

break;

case 4:

break;

default:

break;

}

}while(choice!=4);

return 0;

}

struct node\* create(int x){

struct node\* temp;

temp = (struct node\*)malloc(sizeof(struct node));

temp->info = x;

temp->link = NULL;

return temp;

}

void enqueue(int x){

struct node\* temp = create(x);

if(front==NULL){

front=temp;

rear=temp;

}

else{

rear->link = temp;

rear = temp;

}

}

int dequeue(){

int x;

if(front==NULL){

cout << "Queue is underflow" << endl;

x=0;

}

else if(front == rear){

x = front->info;

front = NULL;

rear = NULL;

}

else{

x = front->info;

struct node\* temp;

temp = front;

front = front->link;

free(temp);

}

return x;

}

void display(){

struct node\* trav = front;

cout << endl << "Elements : ";

while(trav!=NULL){

cout << trav->info << " ";

trav=trav->link;

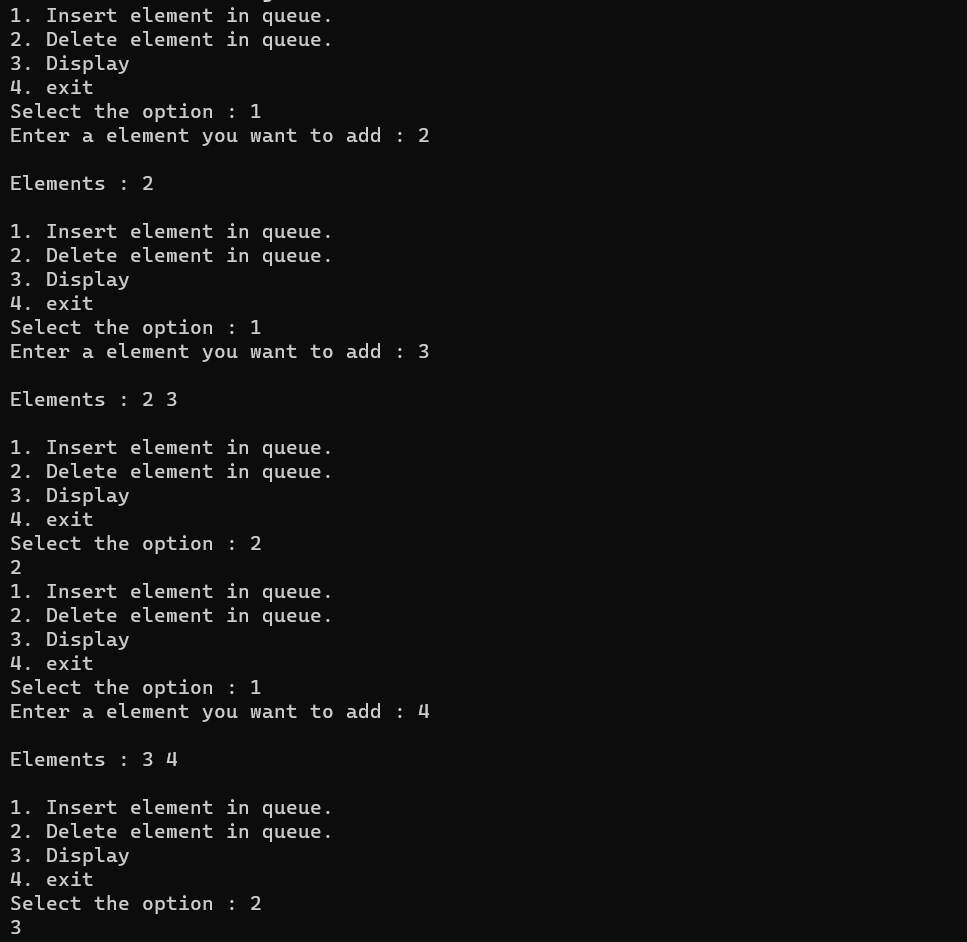
}

cout << endl << endl;

return;

}

* Output



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Student Signature Faculty Signature Marks