1. Implement Stack and Queue operations using single Linked List

#include <iostream>

#include<stdlib.h>

using namespace std;

struct Node {

int data;

struct Node \*next;

};

struct Node\* top = NULL;

void push(int val) {

struct Node\* newnode = (struct Node\*) malloc(sizeof(struct Node));

newnode->data = val;

newnode->next = top;

top = newnode;

}

void pop() {

if(top==NULL)

cout<<"Stack Underflow"<<endl;

else {

cout<<"The popped element is "<< top->data <<endl;

top = top->next;

}

}

void display() {

struct Node\* ptr;

if(top==NULL)

cout<<"stack is empty";

else {

ptr = top;

cout<<"Stack elements are: ";

while (ptr != NULL) {

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

ptr = ptr->next;

}

}

cout<<endl;

}

int main() {

int ch, val;

cout<<"1) Push in stack"<<endl;

cout<<"2) Pop from stack"<<endl;

cout<<"3) Display stack"<<endl;

cout<<"4) Exit"<<endl;

do {

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

cin>>ch;

switch(ch) {

case 1: {

cout<<"Enter value to be pushed:"<<endl;

cin>>val;

push(val);

break;

}

case 2: {

pop();

break;

}

case 3: {

display();

break;

}

case 4: {

cout<<"Exit"<<endl;

break;

}

default: {

cout<<"Invalid Choice"<<endl;

}

}

}while(ch!=4);

return 0;

}

QUEUE:-

#include <iostream>

#include<stdlib.h>

#include<malloc.h>

#include<conio.h>

using namespace std;

struct node {

int data;

struct node \*next;

};

struct node\* front = NULL;

struct node\* rear = NULL;

struct node\* temp;

int Insert() {

int val;

cout<<"Insert the element in queue : "<<endl;

cin>>val;

if (rear == NULL) {

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

rear->next = NULL;

rear->data = val;

front = rear;

} else {

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

rear->next = temp;

temp->data = val;

temp->next = NULL;

rear = temp;

}

}

void Delete() {

temp = front;

if (front == NULL) {

cout<<"Underflow"<<endl;

return;

}

else

if (temp->next != NULL) {

temp = temp->next;

cout<<"Element deleted from queue is : "<<front->data<<endl;

free(front);

front = temp;

} else {

cout<<"Element deleted from queue is : "<<front->data<<endl;

free(front);

front = NULL;

rear = NULL;

}

}

int Display() {

temp = front;

if ((front == NULL) && (rear == NULL)) {

cout<<"Queue is empty"<<endl;

return 0;

}

cout<<"Queue elements are: ";

while (temp != NULL) {

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

temp = temp->next;

}

cout<<endl;

}

int main() {

int ch;

cout<<"1) Insert element to queue"<<endl;

cout<<"2) Delete element from queue"<<endl;

cout<<"3) Display all the elements of queue"<<endl;

cout<<"4) Exit"<<endl;

do {

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

cin>>ch;

switch (ch) {

case 1: Insert();

break;

case 2: Delete();

break;

case 3: Display();

break;

case 4: cout<<"Exit"<<endl;

break;

default: cout<<"Invalid choice"<<endl;

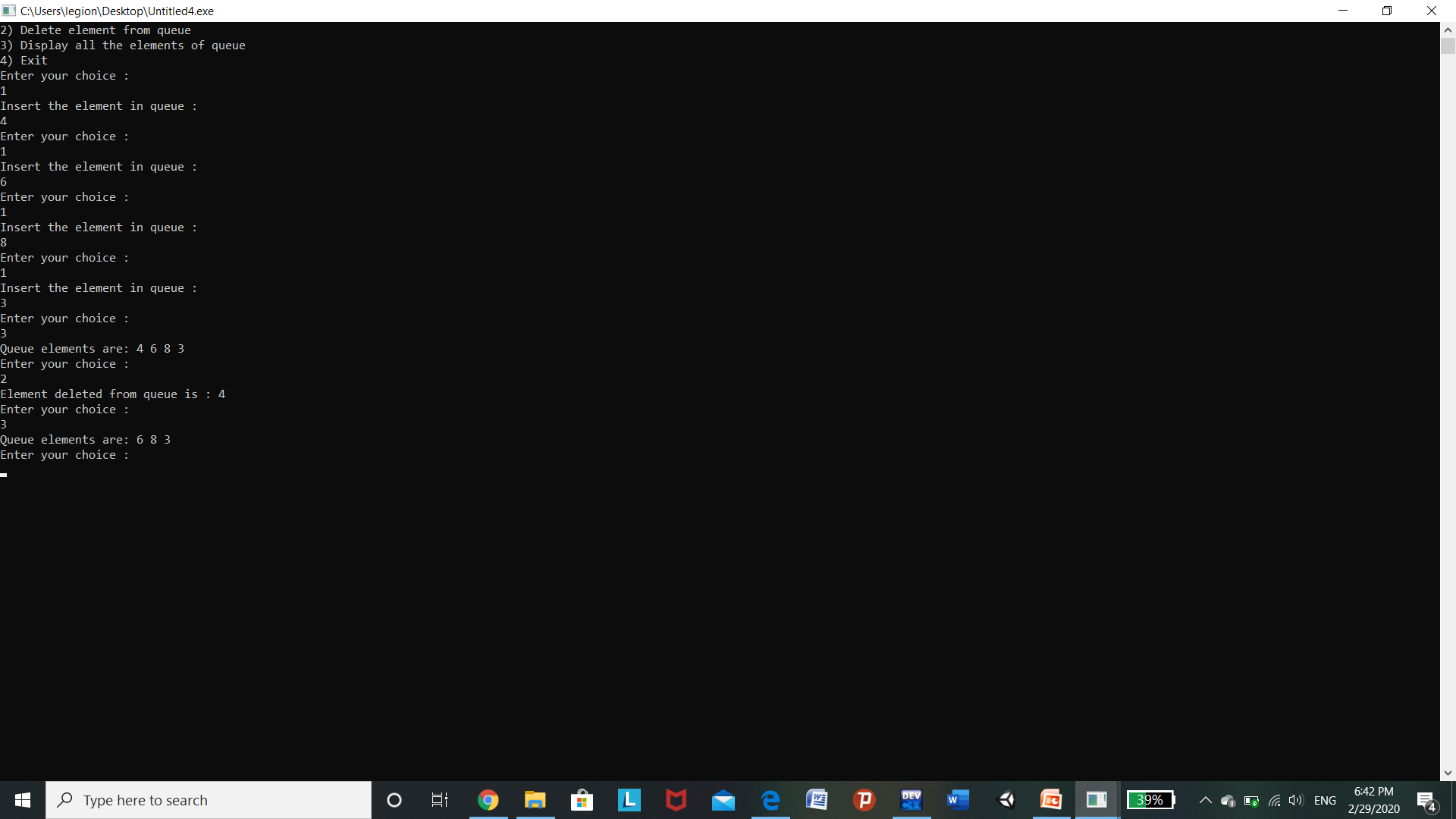
}

} while(ch!=4);

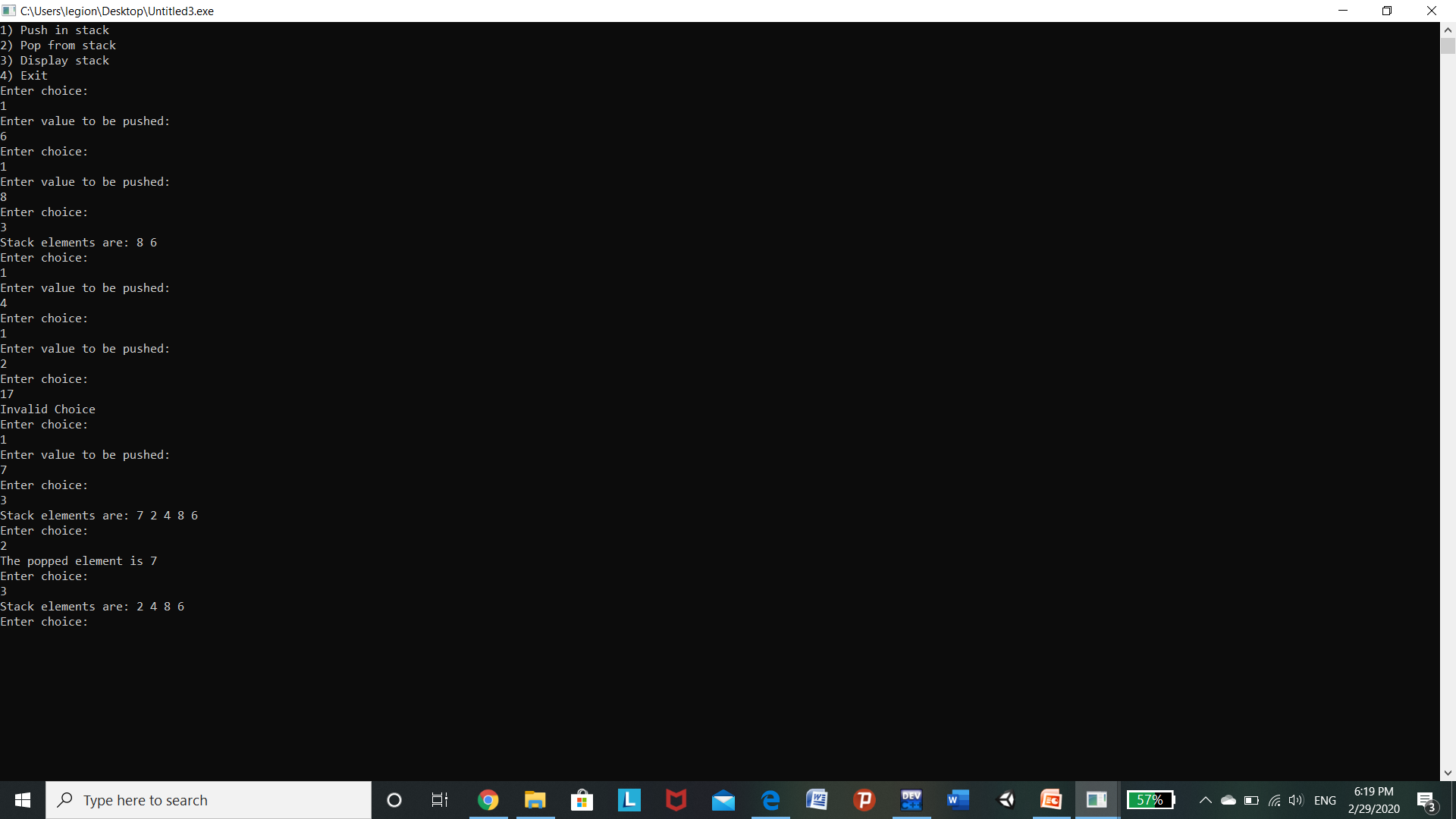
return 0;

}

OUTPUT:-



STACK OUTPUT:-



1. Create a Single linked list and perform the following operations:
   * + 1. Insert a new node
       2. Delete a node
       3. Display
       4. Search for a node

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

struct node {

int value;

struct node \*next;

};

void insert();

void display();

void Delete();

int search();

typedef struct node DATA\_NODE;

DATA\_NODE \*head\_node, \*first\_node, \*temp\_node = 0, \*prev\_node, next\_node;

int data;

int main() {

int option = 0;

printf("Singly Linked List Example - All Operations\n");

while (option < 5) {

printf("\nOptions\n");

printf("1 : Insert into Linked List \n");

printf("2 : Delete from Linked List \n");

printf("3 : Display Linked List\n");

printf("4 : search Linked List\n");

printf("Others : Exit()\n");

printf("Enter your option:");

scanf("%d", &option);

switch (option) {

case 1:

insert();

break;

case 2:

Delete();

break;

case 3:

display();

break;

case 4:

search();

break;

default:

break;

}

}

return 0;

}

void insert() {

printf("\nEnter Element for Insert Linked List : \n");

scanf("%d", &data);

temp\_node = (DATA\_NODE \*) malloc(sizeof (DATA\_NODE));

temp\_node->value = data;

if (first\_node == 0) {

first\_node = temp\_node;

} else {

head\_node->next = temp\_node;

}

temp\_node->next = 0;

head\_node = temp\_node;

fflush(stdin);

}

void Delete() {

int countvalue, pos, i = 0;

countvalue = search();

temp\_node = first\_node;

printf("\nDisplay Linked List : \n");

printf("\nEnter Position for Delete Element : \n");

scanf("%d", &pos);

if (pos > 0 && pos <= countvalue) {

if (pos == 1) {

temp\_node = temp\_node -> next;

first\_node = temp\_node;

printf("\nDeleted Successfully \n\n");

} else {

while (temp\_node != 0) {

if (i == (pos - 1)) {

prev\_node->next = temp\_node->next;

if(i == (countvalue - 1))

{

head\_node = prev\_node;

}

printf("\nDeleted Successfully \n\n");

break;

} else {

i++;

prev\_node = temp\_node;

temp\_node = temp\_node -> next;

}

}

}

} else

printf("\nInvalid Position \n\n");

}

void display() {

int count = 0;

temp\_node = first\_node;

printf("\nDisplay Linked List : \n");

while (temp\_node != 0) {

printf("# %d # ", temp\_node->value);

count++;

temp\_node = temp\_node -> next;

}

printf("\nNo Of Items In Linked List : %d\n", count);

}

int search() {

while(head\_node!=NULL)

{

if(head\_node->value==data)

{

return 1;

}

head\_node=head\_node->next;

}

return 0;

}

Output:-

