TASK 1:

#include <iostream>

using namespace std;

#define SIZE 10 //the size has to be 10 so we have defined it

class Stack{ //Class Stack

int\* stackarr; //int pointer that we will use to make dynamic array

int top; //top element of stack

int capacity;

public:

Stack(int size = SIZE) { // Constructor to initialize the stack

stackarr = new int[size];

capacity = size;

top = -1;

}; // constructor

~Stack() { // Destructor to free memory allocated to the stack

delete[] stackarr; //deleting the dynamic memory that we allocated

}; // destructor

void push(int x) { //function to add an element to stack

if (isFull()){

cout << "stack is already full";

}

cout << "Inserting element (" << x <<") in stack" << endl;

stackarr[++top] = x; //we are inserting elemnt in the stack here

};

int pop() { //function to pop the top element from stack

if (isEmpty()){

cout << "stack is empty";

}

cout << "Removing element from the stack " << peek() << endl;

// decrease stack size by 1 and (optionally) return the popped element

return stackarr[top--];

};

int peek() { // function to return the top element of the stack

if (!isEmpty()) {

return stackarr[top]; //this is the top element in the stack

}

else {

exit(EXIT\_FAILURE);

}

};

bool isEmpty() { // function to check if stack is empty or not empty

return top == -1;

};

bool isFull() { // function to check if the stack is full or not

return top == capacity - 1;

};

};

int main(){

int element;

int insertsize;

int removesize;

Stack var(10); //max can be 10

cout << "How many elements do you want to enter in the stack : ";

cin >> insertsize;

for (int i = 0; i < insertsize; i++) {

cout << "Enter the elemnt to enter in stack : ";

cin >> element;

var.push(element);

}

cout << "The top element present in stack is " << var.peek() << endl;

cout << "How many elements do you want to remove from the stack : ";

cin >> removesize;

for (int i = 0; i < removesize; i++) {

var.pop();

}

if (!var.isEmpty()) {

cout << "The stack is not Empty"<<endl;

}

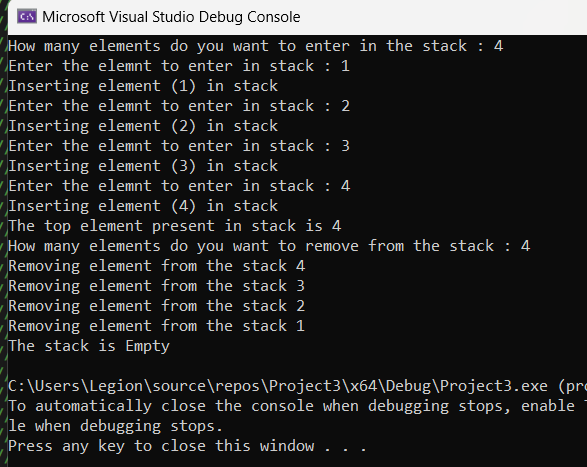
else {

cout << "The stack is Empty"<<endl;

}

return 0;

}



TASK 2:

#include <iostream>

using namespace std;

// Declare linked list node

struct Node {

int data;

Node\* link;

};

Node\* top;

void push(int data){ //function to add element in the stack at the top or beginning

Node\* temp = new Node(); //dynamic new node allocation using new keyword

if (!temp) {

cout << "Stack is already full";

exit(1);

}

// Initialize data into temp data field

temp->data = data; //making temp->data equal to the data that user enters as a parameter in the function when he access it through the main

// Put top pointer reference into temp link

temp->link = top;

top = temp; // Make temp as the top element of Stack

}

int isEmpty(){ //function to check if stack is empty or not

return top == NULL; //if top is NULL it means stack is empty

}

void pop(){ //function to pop top element from the stack

Node\* temp;

// Check for stack emptiness

if (top == NULL) {

cout << "Stack Underflow" << endl;

exit(1);

}

else {

// Assign top to temp

temp = top;

// Assign second node to top

top = top->link;

free(temp);

}

}

void display(){ //print function to print the stack elements

Node\* temp;

if (top == NULL) {

cout << "Stack Underflow";

exit(1);

}

else {

temp = top;

while (temp != NULL) {

cout << temp->data << "--> "; // Print the data in the node

temp = temp->link; // Assign temp link to temp

}

}

}

int main(){

// Push the elements of stack

int size;

int elementforstack;

cout << "Enter how many elements you want to enter in the stack : ";

cin >> size;

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

cout << "Enter the element " << i << " that you want to enter in the stack :";

cin >> elementforstack;

push(elementforstack);

}

display(); // Display stack elements

cout << endl << "Enter how many elements you want to remove from the stack : ";

cin >> size;

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

pop();

}

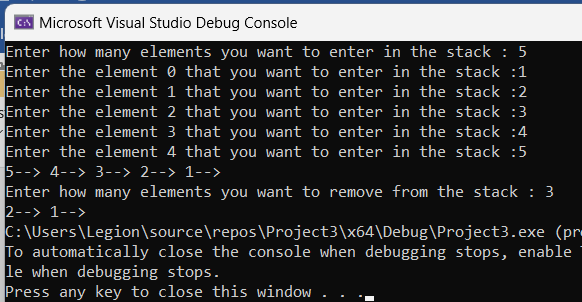
// Display stack elements

display();

// Print top element of stack

return 0;

}



TASK 3:

#include <iostream>

#include <stdlib.h>

#include <string.h>

using namespace std;

#define size 10

char\* stack;

int top = -1;

void push(char);

char pop();

int palindrome(char s[]);

int palindrome(char s[])

{

char character;

int flag = 1, i, j, mid, len;

len = strlen(s);

stack = (char\*)malloc(len \* sizeof(char));

mid = len / 2;

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

push(s[i]); // Pushing all the elements till mid into the stack

if (len % 2 != 0)

i++; // If the length of the string is odd then neglecting the middle character.

while (s[i] != '\0')

{ // Till the end of the string, popping elements from the stack and comparing it with the current character

character = pop();

if (character != s[i]) // If there is no match then the string is not a palindrome

return 0;

i++;

}

// If all the elements match then the string is a palindrome.

return 1;

}

void push(char ch)

{

stack[++top] = ch;

}

char pop()

{

return stack[top--];

}

int main()

{

char str[size];

cout << "enter the string to check it is palindrome or not: ";

cin >> str;

if (!palindrome(str))

cout << "string is not palindrome";

else

cout << "string is palindrome";

return 0;

}

