Egypt-Japan University of Science and Technology.
Electronics, Communications and Computer Engineering (ECCE) School.
CSE Dept.
Dr. Ahmed Arafa.
Data Structure.

Exercise 7: Stack

Q1: Write a C++ program to use an array to implement a stack. Then, push integers of the series 2,4,6 ..20. Later, pop and print on screen the last 5 elements of the stack. Use a struct to combine the stack elements and implement the functions push (), pop (), isfull (), isempty () to implement the stack and call them to achieve the required task.

```
#include <iostream>
using namespace std;
#define STACKSiZE 100
struct stack
{
    int items[STACKSiZE];
    int top=-1;
};

bool isFull(stack stk)
{
    if (stk.top==STACKSiZE-1) return true;
    else return false;
}
bool isEmpty(stack stk)
{
    if (stk.top==-1) return true;
    else return false;
}
void display(stack stk)
{
    if(isEmpty(stk))    cout<<"Stack is empty!";
    else
    {   cout<<"\nStack elements are\n ";
        for(int i=stk.top; i>=0; i--)
          cout<<stk.items[i]<<"  ";
        cout<<endl;
    }
}
void push (stack &stk, int value)
{
    if (isFull(stk))
    {
        cout<<"Overflow. You can't insert new items\n";
        return;
    }
    else
    {
        stk.top++;
        stk.items[stk.top]=value;
    }
}
void pop (stack &stk)
{

    if (isEmpty(stk))
    {
        cout<<"Underflow. Nothing to pop\n";
        return;
    }
    else
    {
        cout<<"poped element is = "<<stk.items[stk.top]<<"\n";
        stk.top--;
    }

}

int main()
{
   stack s;
   int choice;
```

```
    do
    {
        cout<<"\n================================\n";
        cout<<"Stack Operations Menu\n";
        cout<<"================================\n";
        cout<<"1.Push a new item\n";
        cout<<"2.Pop an item\n";
        cout<<"3.Check Full Stack\n";
        cout<<"4.Check Empty Stack\n";
        cout<<"5.Display elements\n"<<endl;
        cout<<"6.Exit\n";
        cout<<"please Enter your choice ";
        cin>> choice;
        switch (choice)
        {
        case 1:
            int value;
            cout<<"Enter a value to push"<<endl;
            cin>>value;
            push(s,value);
            break;
        case 2:
            pop(s);
            break;
        case 3:
            if(isFull(s)) cout<<"Full Stack"<<endl;
            else cout<<"Not Full Stack"<<endl;
            break;
        case 4:
            if(isEmpty(s)) cout<<"Empty Stack"<<endl;
            else cout<<"Not Empty Stack"<<endl;
            break;
        case 5:
            display(s);
            break;
        case 6:
            return 0;
        default:
            cout<<"you entered wrong choice please try again";
            break;
        }
    }
    while (choice !=6);
}
```

****

Q2: Solve the same previous question using dynamic (linked list) based stack and implement the same functions and call them for execution.

```
#include <iostream>
using namespace std;
struct stack {int value; stack * next;};
bool isEmpty(stack * top)
{
    if (top==NULL) return true;
    else return false;
}
void display(stack *top)
{
    if(isEmpty(top))    cout<<"Stack is empty!"<<endl;
    else
    {   cout<<"\nStack elements are\n "<<endl;
        stack * temp=top;
        while (temp!=NULL)
        {
          cout<<temp->value<<"->";
          temp=temp->next;
        }
    }
}
void push (stack * &top, int value)
{
    stack * newNode=new stack;
    newNode->value=value;;
    newNode->next=NULL;
    if(isEmpty(top))
        top = newNode;
```

```
    else
    {
        newNode->next = top;
        top = newNode;
    }
}

void pop (stack *&top)
{
    if (isEmpty(top))
    {
        cout<<"Underflow. Nothing to pop\n";
        return;
    }
    else
    {
        stack *temp=top;
        cout<<"poped element is = "<<top->value<<endl;
        top=top->next;
        delete temp;
    }
}

int main()
{
   stack *s=NULL;
   int choice;
    do
    {
        cout<<"\n=================================\n";
        cout<<"Stack Operations Menu\n";
        cout<<"=================================\n";
        cout<<"1.Push a new item\n";
        cout<<"2.Pop an item\n";
        cout<<"3.Check Empty Stack\n";
        cout<<"4.Display elements\n"<<endl;
        cout<<"5.Exit\n";
        cout<<"please Enter your choice ";
        cin>> choice;
        switch (choice)
        {
        case 1:
            int value;
            cout<<"Enter a value to push"<<endl;
            cin>>value;
            push(s,value);
            break;
        case 2:
            pop(s);
            break;
        case 3:
            if(isEmpty(s)) cout<<"Empty Stack"<<endl;
            else cout<<"Not Empty Stack"<<endl;
            break;
        case 4:
            display(s);
            break;
        case 5:
            return 0;
        default:
            cout<<"you entered wrong choice please try again";
            break;
        }
    }
    while (choice !=5);
}
```

****

Q3: Modify the stack in the first question to store characters instead of integers then push a word then write a function that accepts a word and then check if the word is a palindrome or not. Add an entry to the function in the cases of the first program and call it for excusion.

```
void palindrome(string str, stack stk)
{
    int front =0;
    char b;
    for (int i = 0; str[i] != '\0'; i++)
            {
                b = str[i];
                push(stk,b);
            }
            for (int j=0; j <(str.length()/2); j++)
            {
                if (stk.items[stk.top] == stk.items[front])
                {
                    pop(stk);
                    front++;
                }
                else
                {
                    cout<<str<< " :is not a palindrome\n";
                    break;
                }
            }
            if ((str.length() / 2)  ==  front)
                cout<<str <<" :is a palindrome\n";
            front  =  0;
            stk.top  =  -1;
}
```

****

Q4: Modify the stack in the first question to store characters instead of integers. write a function that accepts a sentence of brackets to check if the brackets are balanced or not. Add an entry to the function in the cases of the first program and call it for execution.

```
bool isBalanced(string expr,stack stk)
{
    char x;
    for (int i=0; i<expr.length(); i++)
    {
        if (expr[i]=='('||expr[i]=='['||expr[i]=='{')
        {
            push(stk,expr[i]);
            continue;
        }

        if (isEmpty(stk))
           return false;

        switch (expr[i])
        {
        case ')':
            x = stk.top;
            pop(stk);
            if (x=='{' || x=='[')
                return false;
            break;
        case '}':
            x = stk.top;
            pop(stk);
            if (x=='(' || x=='[')
                return false;
            break;
        case ']':
            x = stk.top;
            pop(stk);
            if (x =='(' || x == '{')
                return false;
            break;
        }
    }

    return (isEmpty(stk));
}
```

****

Q5: This question is an assignment in the lab and delivered to the lab instructor. Write a C++ program to use a linked list to implement a dynamic stack that is used to store student id and his grade. Implement the push, pop and count functions and then call then for execution.

****

Q6: This Question is a homework assignment, and the student should deliver it within 2 days of the tutorial date. Write a C++ program to evaluate a postfix mathematical expression provided by the user using the array-based stack implementation.