DSA Lab-05 Tasks

Name:	Syed Muhammad Raza Ali
Enrolment:	02-134231-028
Course:	DSA Lab
Faculty:	Miss Rabia

Lab 5: Stack

Objective(s): Upon completion of this lab session, students will be able to:

Implement the concept of stack.

Exercise 1: Library Bookshelf Challenge

Write a In the vibrant world of "Library Bookshelf Challenge," you are tasked with implementing a program to manage a bookshelf using a stack. The bookshelf has a maximum capacity of seven books. Your program should include the following functions: PUSH(), POP(), and Display().

Program Specifications:

1. Bookshelf Initialization:
   - Initialize a stack to represent the bookshelf with a maximum capacity of seven books.
2. Book Interaction:
   - Implement a function addBook (equivalent to PUSH) that allows librarians to add a book to the bookshelf. The function should perform the following tasks:
     • If the bookshelf is not full, add a book to the stack.
     • Display the title of the book added.
     • If the bookshelf is full, print "Bookshelf is full; cannot add more books."
3. Display Books on Shelf:
   - Implement a function displayBooks that displays the titles of all books currently on the bookshelf.
4. Book Removal:
   - Implement a function removeBook (equivalent to POP) that allows patrons to borrow a book. The function should perform the following tasks:
     • If the bookshelf is not empty, remove a book from the stack.
     • Display the title of the book borrowed.
     • If the bookshelf is empty, print "Bookshelf is empty; no books available to borrow."
5. Now, test your program using the following procedure:
   a) Call addBook("Harry Potter")
   b) Call addBook("To Kill a Mockingbird")
   c) Call addBook("The Great Gatsby")
   d) Call addBook("1984")
   e) Call addBook("The Catcher in the Rye")
   f) Call addBook("Pride and Prejudice")
   g) Call addBook("The Hobbit")
   h) Call addBook("The Lord of the Rings")
   i) Display all books on the shelf.
   j) Call removeBook

k) Display all books on the updated shelf.

Code:

```
#include <iostream>
using namespace std;

bool isEmpty(int&top) {
	if (top == 6)
		return true;
	else
		return false;
}
void push(string stack[],string item, int& top) {
	if (isEmpty(top))
		cout << "Stack Overflow\n";
	else {
		top =top+ 1;
		stack[top] = item;
	}
}
void pop(int&top) {
	if (top == -1)
		cout << "Stack underflow";
	else
		top = top - 1;
}
void displayStack(string stack[],int&top) {
	for (int i = top; i >= 0; i--) {
		cout << stack[i] << endl;
	}
}

int main() {

	string stack[7];
	int top = -1;

	cout << "Entering books and printing stack\n";
	push(stack, "Harry Potter", top);
	push(stack, "To Kill a mocking bird", top);
	push(stack, "The Great Gatsby", top);
	push(stack, "1984", top);
	push(stack, "The Catcher in the Rye", top);
	push(stack, "Pride and Prejudice", top);
	push(stack, "The Hobbit", top);
	push(stack, "Lord of the rings", top);
	displayStack(stack,top);
	cout << "Removing an item and printing stack\n";
	pop(top);
	displayStack(stack, top);

	return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
Entering books and printing stack
Stack Overflow
The Hobbit
Pride and Prejudice
The Catcher in the Rye
1984
The Great Gatsby
To Kill a mocking bird
Harry Potter
Removing an item and printing stack
Pride and Prejudice
The Catcher in the Rye
1984
The Great Gatsby
To Kill a mocking bird
Harry Potter
```

Exercise 2: drawing game

Colorful ball

In the vibrant world of the "Colorful Ball Drawing Game," a jar is filled with five distinct balls of colors – Red, Green, Blue, Orange, and Yellow. Enthusiastic players gather to draw a ball fromthe jar, each turn marked by excitement and curiosity. Your task is to create a C++ program that captures the essence of this playful scenario, managing the jar's stack, and providing following essential information to the players.

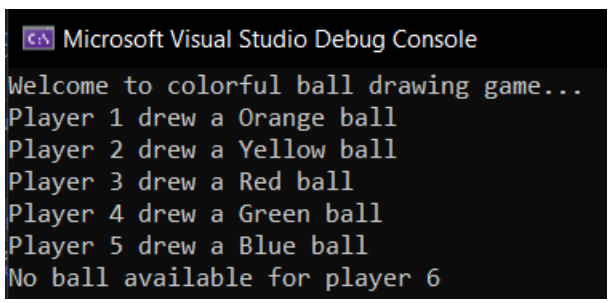
a. How many balls are left in a jar?
b. Colour of ball taken by each player.
c. “No ball available” if palyer6 arrives.

Code:

```
#include <iostream>
using namespace std;

void draw(string stack[],int&top,int&player) {
	if (top == -1)
		cout << "No ball available for player " << player;
	else {
		cout << "Player " << player << " drew a " << stack[top] << " 
ball" << endl;
		top -= 1;
		player += 1;
	}
}
```

```
int main() {
	string drawer[5] = { "Blue","Green","Red","Yellow","Orange" };
	int top = 4;
	int player = 1;
	cout << "Welcome to colorful ball drawing game...\n";
	draw(drawer, top, player);
	draw(drawer, top, player);
	draw(drawer, top, player);
	draw(drawer, top, player);
	draw(drawer, top, player);
	draw(drawer, top, player);




	return 0;
}
```

Output:



Exercise 3: Balanced symbols Checker

Imagine you are building a compiler. Implement a C++ program that uses a stack to check whethera given expression has balanced parentheses, square brackets, and curly braces. Provide meaningful error messages for unbalanced cases.

Code:

```
#include <iostream>
using namespace std;

void push(char stack[],char item,int&top) {
	if (top == 5)
		cout << "Stack Overflow\n";
	else {
		top += 1;
		stack[top] = item;
	}
}
void pop(char stack[],  int& top) {
```

```
	if (top == -1)
		cout << "Stack Underflow\n";
	else {
		cout << stack[top] << " has been removed successfully\n";
		top -= 1;
	}
}
void displayStack(char stack[], int& top) {
	for (int i = top; i >= 0; i--) {
		cout << stack[i] << endl;
	}
}
bool isEmpty(int&top) {
	if (top == -1)
		return true;
	else
		return false;
}
int main() {

	char arr[6] = {'{','[','(',')',']','}'};
	char stack[6];
	int top = -1;

	for (int i = 0; i < 6; i++) {
		if(arr[i]=='{' or arr[i]=='[' or arr[i]=='(')
			push(stack, arr[i], top);
	}
	displayStack(stack, top);
	for (int i = 0; i < 6; i++) {
		for (int j = 0; j < 6; j++) {
			if ((stack[i] == '(' and arr[j] == ')') or (stack[i] ==
'[' and arr[j] == ']') or (stack[i] == '{' and arr[j] == '}'))
				pop(stack, top);
		}
	}
	if (isEmpty(top))
		cout << "Equation is balanced\n";
	else
		cout << "Equation is imbalance";
	return 0;
}
```

Output:

```
Microsoft Visual Studio Debug Console
{ has been pushed in stack
[ has been pushed in stack
( has been pushed in stack
( has been removed successfully
[ has been removed successfully
{ has been removed successfully
Equation is balanced
```

Exercise 4: Postfix Expression Evaluator

Create a C++ program that validates the correctness of a given postfix expression using a stack.

1. **<u>Stack Implementation:</u>**
   a) Implement a stack data structure with functions for push and pop operations.

   b) Define a maximum size for the stack (e.g., MAX_SIZE = 100).

2. **<u>Postfix Expression Validation:</u>**
   a. Implement a function validatePostfix that takes a postfix expression as input and uses astack to validate its correctness.
   b. Process each character in the postfix expression:
   - If the character is an operand, push it onto the stack.
   - If the character is an operator, pop the required number of operands from the stack.
   - If there are not enough operands for the operator, the expression is invalid.
   - After processing the entire expression, the stack should contain only one operand (the final result).
3. **<u>User Interaction:</u>**
   a. Allow the user to input a postfix expression.
   b. Call the validatePostfix function with the input expression.
   c. Display whether the postfix expression is valid or not based on the stack's finalstate.
4. **<u>Example:</u>**
   If the user inputs the postfix expression "23*5+", the program should validate it asa valid expression (2 * 3) + 5.

```
Enter a postfix expression: 23*5+
Postfix expression is valid.
```

```
Enter a postfix expression: 23*+
Postfix expression is invalid.
```

Code:

```
#include <iostream>
#include <array>
using namespace std;

bool isEmpty( int& top) {
	if (top == -1)
		return true;
	else
		return false;
}
void push(char stack[], char item, int& top) {
	if (top == 4)
		cout << "Stack Overflow\n";
	else {
		top += 1;
		stack[top] = item;
	}
}
//masla
char pop(char stack[],  int& top) {
	if (isEmpty(top))
		cout << "Stack Underflow\n";
	else {
		char poppedElement = stack[top];
		top -= 1;
		return poppedElement;
	}
}
void validatePostfix(char stack[],int&top) {
	for (int i = 0; i < 5; i++) {
		if (stack[i] == '+' or stack[i] == '-' or stack[i] == '*' or
stack[i] == '/' or stack[i] == '^' or stack[i] == '%') {
			int a = pop(stack, top) - '0';
			int b = pop(stack, top) - '0';
			int c;
			if (stack[i] == '+')
				c = b + a;
			else if (stack[i] == '-')
				c = b - a;
			else if (stack[i] == '*')
				c = b * a;
			else if (stack[i] == '/')
				c = b / a;
			else if (stack[i] == '^')
				c = b ^ a;
			else if (stack[i] == '%')
				c = b % a;
			push(stack, c + '0', top);
		}
		else
			push(stack, stack[i], top);
	}
	if (top == 0) {
		cout << stack[top];
		cout << "Valid postfix expression...\n";
```

```
        }
        else {
                cout << "Invalid postfix expression...\n";
        }
}

void displayStack(char stack[], int& top) {
        for (int i = top; i >= 0; i--)
                cout << stack[i] << endl;
}
int main() {

        char expression[5] = {'2','3','*','3','+'};
        int top = -1;
        validatePostfix(expression, top);

        return 0;
}
```

Output: