

King Abdulaziz University Faculty of Computing and Information  
Technology Computer Science Department

**CPCS204, Spring 2023 Program 3**

**Assignment 3: stack**

**Assigned: Thursday 4<sup>th</sup> May 2023    Due: Thursday, 18<sup>th</sup> May 2023**

**Purpose**

1. Learn to implement a stack for a real-world problem.
2. Review file I/O (input/output).

**Read Carefully:**

This program is worth 5% of your final grade.

**WARNING:** This is an individual project; you must solve it by yourself. Any form of cheating will result in receiving a **zero** in the assignment.

The deadline for this project is **Thursday, 18th May 2023 by 11:59 PM.**

**LATE SUBMISSION:** No assignment will be accepted after the deadline.

**Blackboard Submission:**

This project must be submitted online via Blackboard.

The source file(s) of your program should be zipped up. You must name the zip file using the following naming convention:

SectionNumber\_StudentID\_ProgramNumber.zip

Question:	1	2	3	Total
Points:	20	20	60	100

### Question 1: (20 points)

1. Suppose the following operations are performed on the given stack S containing integers. What is the output of the following code segment? Also show the content of Stack S.

**Note:** Use the stack operations push(), pop(), and top() methods, and the top is a reference to element **11** in the stack.

```
S.pop();
S.push(11);
S.pop();
S.pop();
S.pop();
S.push(2);
S.push(3);
S.push(4);
S.push(5);
```

11
30
25
17
1
4
20

S

```
for(int i = 0; i < 7; i++)
{
    if (S.top() % 2 == 0)
        System.out.print(S.top() + " ");
    S.pop();
}
```

### Question 2:(20 points)

1. Convert the following infix expression to postfix  $a+b*c+d*e\#$ .
2. Evaluate the following postfix notation of expression. What is the top of the stack after third \* operations?

6 2 3 \* / 3 4 \* + 3 6 \* -

3. Show the step-by-step process of evaluation of the postfix expression  $2\ 4\ 6\ +\ *$  and show the status of the stack after every operation.

### Question 3: (60 points)

#### Program 1: URL Stack System

##### Objective:

The goal of this program is to develop a browser software that stores recently visited web pages. The primary objective of this program is to implement a stack using an array. The secondary objective is to practice with File I/O.

##### Program Description:

Write a program that uses stack to push the URL links for the recently visited web pages. If the user clicks on **go back** or **visits a new page**, the operation **pop** and **push** will be used respectively. Also, the user can use only 3 browser tabs to create a stack for each tab.

There is a main history stack for all the **visited web pages for all the tabs**. You are required to write the following methods:

1. Write a function to push the URL link for the recently visited web page to a specific tab stack.
2. Write a function to pop the tab stack element to get the URL link for the previously visited page.
3. Write a function to push any **visited web pages in any tabs** using its URL link to the history stack.

The program deals with three files. Two input files and one output file. Information on the same line is separated by spaces. The description of these files is as follow:

- The first input file (**initialInformationStack.txt**) contains important information for the system, including information about the recently visited web pages, and the favorite bookmark. The information in this file is arranged as follow:

- The first line contains **the number of recently visited web pages**. (Note: The size of each tab **stack** is 10 while the size of the **history stack** stacks is 50)
- The following lines contain the URL link for **recently visited web pages for all tabs**.

- The commands for the system are found in the second file called **commandsStack.txt**. The commands in this file are as follow:

- **STARTUP**: This command will use the first input file (initialInformation.txt) to initialize the system by creating the **tab stacks** arrays (equal to the number of the specified number in the file) and the **history stack** array. All the URL links for **recently visited web pages** should be pushed to its **tab stack** (see **initialInformation.txt**)
- **GO\_FORWARD**: This command requires two values that determine the tab name and the URL link for the **recently visited web page**. It will push the URL

link into the **tab stack**. If no **available space in the tab stack** exists, the system shows “The (tab..) stack is full, the following URL (....) cannot be stored!!”. If no **tab stack** is found, the system shows “This tab is not found”.

- **GO\_BACK**: This command requires two values that determine the tab name and the URL link for the **recently visited web page**. It will pop the URL link from the **specified tab stack** and push it to the **history stack**. If no **available space in the history stack** exists, the system shows “The history stack is full, the following URL (....) cannot be stored!!”.
  - **DISPLAY\_ALL\_VISITED**: This command will display all the URL links for the **recently visited web page for all tabs**. It will pop all the URL links from the all **tab stack** as it is shown in the **ouput.txt** file.
  - **QUIT**: This command will stop the program.
- The output of the program should be written to the file name output.txt, which content should be similar to the contents of the file provided to you.

### **Implementation**

For this program, you will create the following classes:

- **webPageStackArray.java**: This class will be used to create objects of type webpage stack array. The stack object will store the URL for the web pages as a string. All the stack methods will be implemented in this class.
- **MainProgram.java**: This is the class that will contain the main.

### **Sample Input & Output File**

We have provided you with a sample for two input files and one output file.

### **Grading Details**

Your program will be graded upon the following criteria:

- 1) Adhering to the implementation specifications listed on this write-up.
- 2) Your algorithmic design.
- 3) Correctness.
- 4) **Use of the three classes, as specified. If your program is missing these elements, you will lose marks.**
- 5) The frequency and utility of the comments in the code, as well as the use of white space for easy readability. (If your code is poorly commented and spaced and works perfectly, you could earn as low as 80-85% on it.)
- 6) Compatibility to the **newest version** of NetBeans. (If your program does not compile in NetBeans, you will get a large deduction from your grade.)
- 7) Your program should include a header comment with the following information: your name, **email**, account number, section number, assignment title, and date.
- 8) Your output MUST adhere to the EXACT output format shown in the sample output file.

## Deliverables

You should submit a zip file with four files inside:

1. *ConceptPart.doc (Algorithm and Method Write up)*
2. *webPageStackArray.java*
3. *MainProgram.java*

**\*\*\*These three files should all be INSIDE the same package called `URLStackSystem`. If they are not in this specific package, you will lose points.**

**NOTE: your name, ID, section number AND EMAIL should be included as comments in all files!**

## UML Diagrams:

For this program, you will create **two** Classes (UML diagram shown below):



