United International **University**

Department of Computer Science and Engineering

CSE 2216: Data Structure & Algorithms I Lab

Assignment on **Stack and Queue**

Stack

Task 1

Stacks are dynamic data structures that follow the Last In First Out (LIFO) principle. The Insertion and Deletion of an element from the stack are defined by the two corresponding operations - Push() and Pop().
Your task is to use an array of size **N** and the number of times the user wants to do any operation in the stack, **T**. Each time of the **T** operations, the user will choose whether s/he wants to do push/pop. If the operation is Push, the system will ask one more value **item** from the user. For each test case, the system will show the state & size of the stack. If the stack is empty, the state will be (size=0,items=NULL).
For ease of implementation, let's assume that Push is represented by 1 and pop is represented by 2.

Sample Input:	**Sample Output:**
10 7	
1 10	size=1 items=10
1 20	size=2 items=10 20
2	size=1 items=10
1 50	size=2 items=10 50
2	size=1 items=10
2	size=0 items=NULL
1 25	size=1 items=25

Task 2

C++ has a header file called `stack` which has built-in functionalities already implemented as library functions. Follow this link and try to understand the usage of each function.
Using these STL library functions, write a program which will take **N** strings as input and check whether they are Palindrome using `stack`. Input will be the number of test cases, **N**, followed by N strings.

Sample Input:	**Sample Output:**
3	
UIU	Yes
data	No
racecar	Yes

Task 3

Given a string s containing just the characters '(', ')', '', '', '[' and ']', determine if the input string is valid.
An input string is valid if:

- Open brackets must be closed by the same type of brackets.
- Open brackets must be closed in the correct order.
- Every close bracket has a corresponding open bracket of the same type.

Sample Input:	**Sample Output:**
()	true
()[]	true
[()]	true
[(])	false

Queue

Task 1

Queues are dynamic data structures that follow the First In First Out (FIFO) principle. The Insertion and Deletion of an element from the queue are defined by the two corresponding operations - Enqueue()/Push() and Dequeue()/Pop().
Your task is to use an array of size **N** and the number of times the user wants to do any operation in the queue, **T**. Each time of the **T** operations, the user will choose whether s/he wants to do enqueue/dequeue. If the operation is Enqueue, the system will ask one more value **item** from the user. For each test case, the system will show the state & size of the queue. If the queue is empty, the state will be (size=0, items=NULL).
For ease of implementation, let's assume that Enqueue is represented by 1 and Dequeue is represented by 2.

Sample Input:	**Sample Output:**
10 7	
1 10	size=1 items=10
1 20	size=2 items=10 20
2	size=1 items=20
1 50	size=2 items=20 50
2	size=1 items=50
2	size=0 items=NULL
1 25	size=1 items=25

Task 2

C++ has a header file called [*queue*] which has built-in functionalities already implemented as library functions. Follow this link and try to understand the usage of each function.
Using these STL library functions, write a program which will take **N** strings as input and print the reverse of them using Queue. Input will be the number of test cases, **N**, followed by N strings.
Hint: You will need to use two queues.

Sample Input:	**Sample Output:**
3	
UIU	UIU
data	atad
mozzarella	allerazzom

Task 3

Design a queue data structure using two stacks. Each stack should have push, pop, and isEmpty functions. The queue should support the following operations:
enqueue(int x): Add element x to the back of the queue.
dequeue(): Remove and retrieve the element from the front of the queue.

(Bonus) Task 4

The school cafeteria offers circular and square sandwiches at lunch break, referred to by numbers 0 and 1 respectively. All students stand in a **queue**. Each student either prefers square or circular sandwiches.
The number of sandwiches in the cafeteria is equal to the number of students. The sandwiches are placed in a **stack**. At each step:

- If the student at the front of the queue prefers the sandwich on the top of the stack, they will take it and leave the queue.
- Otherwise, they will leave it and go to the queue's end.

This continues until none of the queue students want to take the top sandwich and are thus unable to eat. Your job is to find the number of students that are unable to eat.
Input will consist of the number of students, N. Followed by an array containing the preferences of each of the N students. Followed by another array containing the sequence of the N sandwiches at the beginning.

Explanation:

- Front student leaves the top sandwich and goes to the end of the line making students = 1 0 0 1
- Front student leaves the top sandwich and goes to the end of the line making students = 0 0 1 1

Sample Input:	**Sample Output:**
4 1 1 0 0 0 1 0 1	0
Sample Input:	**Sample Output:**
6 1 1 1 0 0 1 1 0 0 0 1 1	3

- Front student takes the top sandwich and leaves the line making students = 0 1 1 and sandwiches = 1 0 1
- Front student leaves the top sandwich and goes to the end of the line making students = 1 1 0
- Front student takes the top sandwich and leaves the line making students = 1 0 and sandwiches = 0 1
- Front student leaves the top sandwich and goes to the end of the line making students = 0 1
- Front student takes the top sandwich and leaves the line making students = 1 and sandwiches = 1
- Front student takes the top sandwich and leaves the line making students = NULL and sandwiches = NULL

Hence all students are able to eat.