Middle East Technical University | Department of Computer Engineering

CENG 310

ALGORITHMS AND DATA STRUCTURES WITH PYTHON

Spring 2022-2023

Homework 2 - Stacks and Queues

Due Date: 23/04/2023 23:59

Instructions. In this assignment you will write a simple program by using stacks and queues. In order to complete the task you first need to implement **your own** stack and queue classes. Your codes should be written by using Python3. You may use built in functions.

1 Implementation of Stack

You need to implement your stack as a class with a member variable(a container type, i.e list) and member functions. You will initialize the stack as:

myStack = Stack()

Your Stack class will have the following member functions:

- __init__(): A function that initializes the stack object.
- push(element): inserts a particular element into the stack.
- pop(): removes element from the top of the stack and returns the element.
- top(): returns the element that is at the top of the stack without removing the element.
- isempty(): return whether the stack is empty or not. Returns True when the stack is empty, False otherwise.

You will be graded based on your implementations of the stack, therefore do not change the name of the class and its member functions. You will not be graded based on the efficiency (run time) but the functionality of the implementation.

2 Implementation of Queue

You need to implement your queue as a class with a member variable(a container type, i.e list) and member functions. You will initialize the queue as:

myQueue = Queue()

Your Queue class will have the following member functions:

- __init__(): A function that initializes the queue object.
- enqueue(element): inserts a particular element into the queue.
- dequeue(): removes element from the front of the queue and returns the element.
- front(): returns the element that is at the front of the queue without removing the element.
- isempty(): return whether the queue is empty or not. Returns True when the queue is empty, False otherwise.

3 Programming Task

A formal language consists of words whose letters are taken from an alphabet and are well formed according to a set of specific rules.[1]. Those languages are represented as sets and each element, called word or string, consists of one or more symbols from the alphabet, which is the set of available symbols. In this task, you will be given 2 different languages called L_1 and L_2 and will develop an algorithm for each one using either a stack or a queue to determine whether a given word belongs to those languages.

$$L_1 = \{a^i b^j c^k \mid i, j, k \geq 1 \text{ and } i = j \text{ or } i = k\} \text{ where } a, b \text{ and } c \text{ are only letters of the alphabet} \quad (1)$$

$$L_2 = \{w\$w \mid w \in \{0,1\}^* \text{ and } |w| \geq 1\} \quad (2)$$

where w is any word consists of only 0s and 1s and \$ is a special token to distinguish start and end point of word w.

3.1 Specifications

- You will implement your algorithm with a function named languageRecognizer(word, languageType).
- If the word is in the language then your function must return "Word is in the language". Otherwise, return "Word is not in the language".
- Both word and languageType will be given as strings and languageType can be either "l1" or "l2".
- You must select whether to use a stack or a queue when developing your algorithm for both languages since you are not allowed to use both of them for the same language.
- Hint: One of the languages is more suitable for solving with a stack and the other one is more suitable for solving with a queue.

3.2 Sample I/O

```
Input:
>>> languageRecognizer("aabbcccccc", "l1")
Output:
Word is in the language

Input:
>>> languageRecognizer("aaaabcccc", "l1")
Output:
```

[1] https://en.wikipedia.org/wiki/Formal_language

```
Word is in the language

Input:
>>> languageRecognizer("010100$01010", "l2")
Output:
Word is not in the language
```

4 Submission & Grading

- Grading will be done for each parts separately. Stack and Queue implementations will be 15 points each and the programming task will account for 70 points.
- Any solutions without these data structures will get 0.
- You need to make sure that you do not have unnecessary prints and function calls in your program. Otherwise you may get penalized.
- You need to submit a single file with a *.py* extension.
- Do not print any unnecessary words/sentences and make sure you obey the output format.
- Be sure that you use the exact names for the classes and function that you will implement.
- You should not worry about the run time of your program as long as it gives the desired output. However, extraordinarily long (in terms of time) solutions may result in getting no credit from particular test case.
- You are expected to handle erroneous conditions when implementing the stack and queue(you may refer to lecture slides or textbook).