Assignment 1

420-ENM-MT Algorithm, Pseudo-code and Design

Due: November 22nd 2018 before class

Instructions

- This assignment is worth **5% of your final grade**;
- You must submit three (3) files on **Omnivox**. Submit theory questions as a single **pdf** and coding questions as `.py` files;
- **Read every question carefully**;
- Some problems require you to write code. Make sure to read the instructions and submit the corresponding `.py` files with your assignment;
- Your code will be tested with **Python 3.7**;
- Using online resources is allowed, but you should cite your sources for any algorithm or code you did not write yourself. You must prove you understood the solution by properly commenting it, otherwise you will be given a grade of zero for **plagiarism**.

Problem 1 (5 points)

Consider the following operations on a **stack**.

```
stack = empty stack
stack.push(10)
stack.push(9)
item1 = stack.pop()
stack.push(8)
stack.push(7)
item2 = stack.pop()
stack.push(item1)
stack.push(item2)
```

Draw what the stack looks like **after each operation**.

Problem 2 (10 points)
You are using a **queue** to build an app that helps users to manage administrative tasks.

This queue supports three operations:

- **add**: enqueues a task in the queue
- **do**: dequeues from the queue
- **skip**: dequeues a task and sends it back to the start of the queue

Consider the following operations on that queue.

```
tasks = empty queue
tasks.add('email CEO')
tasks.add('grab coffee')
tasks.do()
tasks.add('give pay raise')
tasks.skip()
tasks.add('hire employees')
tasks.skip()
tasks.do()
```

a) (5 points) Draw what the queue looks like **after each operation**.

b) (5 points) Write down a **pseudo-code** implementation of the `skip` function. The function should take a `queue` as argument and does not return anything.

Problem 3 (15 points)

In class, we mentioned that a doubly-linked-list is a good data structure for implementing a **queue**. In this problem, we will verify if an **array-list** could also be a good choice of data structure.

a) (2 points) In one or two sentences explain which two features of a **doubly-linked-list** allows to implement a **queue** efficiently.

b) (5 points) The function **prepend** inserts an `item` at the beginning of an **array-list**. Write the pseudocode for the function prepend. The function should take a `list` and an `item` as arguments and return nothing.

c) (5 points) Using the pseudocode you wrote in the previous question, count how many operations are needed to **prepend** an item and determine the **time-complexity** of the function.

d) (3 points) In one or two sentences explain why an array-list **is not a good choice** of data structure to implement a queue.

Problem 4 (10 points)
You are working for a company developing an IDE similar to PyCharm. You have been given the task of implementing parentheses matching. Parentheses matching warns the user if their opening and closing parentheses do not match.

Using **pseudo-code**, implement a function called `match` which takes a string of parentheses as argument and returns `True` if the parentheses match and `False` if the parentheses mismatch.

Examples:

```
match(“( ( ) )”) → True
match(“( ( ) ( ) )”) → True
match(“( ( )”) → False
match(“) ) ( (“) → False
```

Hint: You will need to use a `stack`.

Problem 5 (10 points)
The administrative department of an airport wishes to automatically call passengers who are late for their flight. They already implemented a text-to-speech application that is connected to the airport's announcement system. They need you to write a function that will generate the text to be read based on the passenger's information.

Write a Python function `generate_text` that takes three parameters: the passenger's name, their flight number and the gate of departure. The function should return the formatted text in the following format.

`[name]``, go to gate `*`[gate number]`*` for flight `*`[flight number]`*`.`

Both the passenger name and the flight number are `strings`. The gate number will be provided as an `integer.`

Examples:

```
generate_text("Sam Roger", "AI221", 12)
→ "Sam Roger, go to gate 12 for flight AI221."

generate_text("Tanya Patel", "AC123", 3)
→ "Tanya Patel, go to gate 3 for flight AC123."
```

You are not allowed to use any Python library.

Answer this question by filling in the file `generate_text.py`, then submit it with your assignment.

Problem 6 (10 points)

In class, we implemented a Python function to output the n^{th} element of the Fibonacci sequence. You must now implement a function `fibonacci_sequence` that takes a positive `integer n` as input and outputs a `list` containing all of the `n` first elements of the Fibonacci sequence.

Examples:

- `fibonacci_sequence(0)` → `[]`
- `fibonacci_sequence(2)` → `[0, 1]`
- `fibonacci_sequence(8)` → `[0, 1, 1, 2, 3, 5, 8, 13, 21]`

Both the correctness and the efficiency of your algorithm will be evaluated.

You are not allowed to use any Python library.

Answer this question by filling in the file `fibonacci_sequence.py`, then submit it with your assignment.