COL100 Major Exam

Date: Saturday 13 February, 2021

Instructions:

1. The duration of the exam is 90 minutes. The total marks are 42.
2. This is a closed book exam. You cannot look at your notes or browse the internet.
3. Attempt each question in a new page.
4. For fill-in-the-blanks questions please only mention what the blanks must get filled with.

1 Exam week

A student in COL100 has n topics to study, numbered from 1 to n. He decided to study one topic a day and has designed a *schedule* : $\mathbb{N} \rightarrow \mathbb{N}$ (function that takes input day no. and outputs the topic to study on that day). For eg. on day K, he will study topic numbered $schedule(K)$. He will study these n topics in n days.

A topic(say k) is said to be *well studied* if he has studied all the topics numbered from 1 to k. For example, if $n = 5$ and the schedule is given as

$$schedule(1) = 2, \quad schedule(2) = 1, \quad schedule(3) = 3, \quad schedule(4) = 5, \quad schedule(5) = 4$$

i.e. on day 1 he will study topic 2 and on day 2 he will study topic 1 and so on.

Day	Studied topics	Well studied topics
1	2	None
2	1, 2	1, 2
3	1, 2, 3	1, 2, 3
4	1, 2, 3, 5	1, 2, 3
5	1, 2, 3, 4, 5	1, 2, 3, 4, 5

Then on day 2, 3, 5 all the studied topics are well studied.

Write a function in mathematical notation or an SML function which takes input *schedule*, n and returns the number of days when all the studied topics are well studied (e.g. 3 in the above example). Also explain your algorithm in words. For full marks, your algorithm should take $O(n)$ time. [5 marks]

2 Balanced Parentheses

The problem of balanced parenthesis is that given a string `s` containing just the characters `‘(’, ‘)’, ‘{’, ‘}’, ‘[’ & ‘]’`, determine if all parentheses are correctly matched.

For instance, `"()"`, `"()[]{}"`, `"{[]}"` are examples of balanced strings. `"(]"`, `"([)]"`, `"())"`, `"()("` are examples of unbalanced strings.

Suppose I have a class `Stack` with a private list attribute, and the following public methods:

1. `size(self)`: return the size of the stack.
2. `top(self)`: return the top element of the stack (given `size` > 0).
3. `pop(self)`: remove the top element of the stack (given `size` > 0).
4. `push(self, x)`: push `x` to the stack.

Also, I have function `index(l, a)` which takes a list `l` and an element `x` and returns the index of `x` in `l`, assuming `x` is present in `l`. For example, `index(['a','b','c','d'],'c')` gives 2.

The following function `balanced` takes as input a string and returns a bool denoting whether the string is balanced or not. The helper function `check` ensures that the execution is stopped if we are unable to find an open bracket corresponding to some closed bracket. Fill in the five blanks (a), (b), (c), (d) and (e) to make the code work. You don't need to write the entire code to your sheet again. Just writing the values for (a)-(e) is enough. [5 marks]

```
def check(stk, x):
    if stk.size() == 0:
        return True
    if __(a)__ != x:
        return True
    __(b)__
    return False
def balanced(s):
    stk = Stack()
    op1 = [‘(’, ‘[’, ‘{’]
    op2 = [‘)’, ‘]’, ‘}’]
    for x in s:
        if __(c)__:
            stk.push(x)
        else:
            if check(stk, __(d)__):
                return False
    return __(e)__
```

3 Remove all occurrences

1. Write a function `remove_all` in Python that takes in an integer list (not necessarily sorted) and an integer x. The function should remove all occurrences of integer x from the list. [4 marks]

   Note: Do not return anything from the function, make all changes in the same list. The extra space required by your program should not exceed $O(1)$. The final order of other elements does not matter.

2. Analyse the time complexity of your program. For full marks, the time complexity should be $O(n)$, where n is the length of the list. Remember that `list.pop()` takes $O(1)$ time, but `list.pop(k)` might not. [3 marks]

4 Numerical Computing

The solutions of the quadratic equation $ax^2 + bx + c = 0$ are given by

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

An alternate formula for the same roots is

$$x_{1,2} = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}$$

According to the coefficients a, b and c, how will you decide which formula to use so that you can ensure that the obtained roots are as numerically stable as possible? [3 marks]

5 Recursion

You live at location (x_1, y_1) and want to go to the airport which is at location (x_2, y_2). However there are only two valid movements from any location say (x, y), that is $(x + y, y)$ and $(x, x + y)$. You want to determine if it is possible for you to reach the airport or not. Note: $x_1, y_1, x_2, y_2 > 0$.

Example:
Input: $(x_1, y_1) = (2, 10)$, $(x_2, y_2) = (26, 12)$
Output: true, because $(2, 10) \to (2, 12) \to (14, 12) \to (26, 12)$ is a valid path.

1. Design a recursive function `isReachable(x1,y1,x2,y2)` to check if the airport is reachable from your home, and return a boolean (`true` or `false`). [3 marks]
2. Prove the correctness of your algorithm. [4 marks]

6 Overlapping Intervals

Given a list of intervals of the form $[start_i, end_i]$, design an algorithm to merge all the overlapping intervals and return a list of the non-overlapping intervals that cover all the intervals in the input. Assume that the intervals are given in increasing order of $start_i$.

(Note: All numbers are positive integers, and for any given interval we have $start_i \leq end_i$.)

Input: $[[1, 3], [2, 6], [8, 10], [15, 18]]$
Output: $[[1, 6], [8, 10], [15, 18]]$

Explanation: Since intervals $[1, 3]$ and $[2, 6]$ overlap, we merge them into $[1, 6]$.

1. Design a Python function `nonOverlappingIntervals(intervals)`, which takes a list of intervals as input and return a list of the non-overlapping intervals. [4 marks]
2. Provide the correctness of your algorithm. [4 marks]

7 Object Oriented Programming

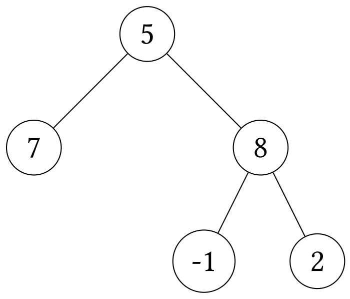
Consider the class for a tree node given below:

```
class TreeNode:
    def __init__(self, x, left, right):
        self.data = x
        self.left = left
        self.right = right
```

Some sample code that uses this class is given below:

```
a = TreeNode(7, None, None)
b = TreeNode(-1, None, None)
c = TreeNode(2, None, None)
d = TreeNode(8, b, c)
root = TreeNode(5, a, d)
```

This code builds the tree represented by the diagram below.



1. Modify the above class to add a method `getLargest(self)` that finds the largest element in the tree rooted at the given TreeNode object. [3 marks]

   For example, for the above tree `root.getLargest()` should give 8 since it is the largest value in the tree.

2. Prove the correctness of your method. [4 marks]

   (**Hint**: You may want to use induction on the total number of nodes in the tree.)