Assessed Exercise 2

This exercise is worth a total of 30 points and counts for 10% of the total assessment mark for this course.

The submission deadline is: Friday, 22 March 2024, 4:30pm.

Submission Instructions

Your solution must be submitted in **Moodle**. It should contain *only* **three files**:

1. A pdf file named "`Name_MatrNum.pdf`", where `Name` is your last name and `MatrNum` your matriculation number. You should write all your answers in this file, except your java source code for questions 1B and 2D, which should be provided in the following files:
2. "`ValidParentheses.java`": your Java implementation for Question 1B
3. "`BSTHeightHistogram.java`": your Java implementation for Question 2D

Your source code should be able to be compiled (without errors) and executed in a standard Unix shell, by running the commands:

```
javac file_name.java
java file_name
```

No other operating-system dependent setup should be assumed, and your execution should not rely on any special PATH variable configuration, IDEs, or dependencies. In a nutshell, *we should be able to easily compile and run your code!*

Finally, you should not make use of any special Java libraries, and you should only import the classes that we explicitly allow you to in the statement of Questions 1B and 2D.

Question 1 (Balanced Expressions)

You want to design an algorithm that solves the following problem: as input you are given a string that may contain *only* the parenthesis characters

`'('` , `')'` , `'{'` , `'}'` , `'['` and `']'`

and you are asked to determine whether the input expression is "valid", i.e., if the parentheses are properly balanced. For example, expression "`[{()}]()`" is valid, but expressions "`[{()]()`" and "`[{{()})]`" are not valid.

A. Discuss how you can use a stack data structure to solve your problem.

[4]

B. Implement your algorithm in Java. Feel free to import and make use of the class `java.util.Stack` for your stack operations. When run, your code should prompt the user to input the string of parentheses; then, it should correctly output "`The expression is valid`" or "`The expression is not valid`".

You may assume that your input string will contain only parenthesis characters, and no other special characters. That is, your code is expected to be able to handle inputs like "`[{{()}}]`" but you should not worry about dealing with inputs like "`[,{,{,(,),},},]`" or "`[ { { ( ) } } ]`". You can import the class `java.util.Scanner` in order to handle the user input.

[7]

Question 2 (Binary Search Trees)

A binary tree is called *perfectly balanced*, if the left and right sub-trees of all its nodes have *exactly* the same height.

A. Prove that any (non-empty) perfectly balanced binary tree of height h has exactly $2^{h+1} - 1$ many nodes.

[4]

B. Give *two* different orderings in which we can enter the values $\{1,2,\ldots,7\}$ in a (initially empty) binary search tree (BST) so that in the end it is perfectly balanced. For one of these input sequences (whichever you like), draw *all* intermediate BSTs that arise after each insertion (that is, you have to draw seven BSTs).

[4]

C. Draw *two* different BSTs whose in-order traversal outputs the sequence $1,2,3,4,5$.

[3]

D. Implement the following algorithm in Java: you ask the user to input a positive integer n. Then, for each of the $n!$ Many different permutations of the values $\{1,2,\ldots,n\}$, you compute the heigh of the BST that arises if we insert, at an initially empty BST, the specific sequence of elements. Then, you print a histogram with the frequencies of the different heights than can arise, as well as the average height.

Your output should be in the following form (here given for $n = 7$):

```
Height  Frequency
----------------
0       0
1       0
2       80
3       2240
4       2048
5       608
6       64

Average height of BSTs:
3.6698412698412697
```

That is, for each possible value of the height $h = 0,1,2, \ldots, n-1$, at the column "Frequency" we write down how many of the $n!$ permutations resulted in a BST of height equal to h.

For your Java implementation, feel free to import the classes `java.util.ArrayList`, `java.util.Collections`, `java.util.List`, `java.util.Scanner`.

[8]