**Algorithm Exercises**

1. Create a Java program. It contains a method called maxCharacter(String). Say you pass a string to this function and the String is **‘aaafbbbdeeeda’**, you need to return the 1st index of the maximum character. In this case it is ‘a’ and first index is ‘0’. Another example is **‘eeddfssses’**; the value is ‘s’ and the first index is ‘5’.
2. Determine if a user input string is a palindrome. A palindrome is a string that has the same value if it is reversed. Examples: “abba”, “level”. What is the performance or big O value of your algorithm?
3. The enclosing characters in a program must be balanced. In other words for every opening parenthesis, brace, bracket or similar type there must be a corresponding closing type. Examples { }, [ ], ( ), < >. From a file or string input determine a good data structure to use to test for balanced enclosing characters. Input a file or string with any of these characters and determine if all the opening and closing characters match. Create a method that takes a String parameter and returns a boolean equal to the result of the test. Test the strings below or use others if you want.
4. “{{{}}}”
5. “(((()))”
6. Do the HackerrRank challenge at link [Birthday Cake Candles | HackerRank](https://www.hackerrank.com/challenges/birthday-cake-candles/problem) The problem is below also.

You are in charge of the cake for a child's birthday. You have decided the cake will have one candle for each year of their total age. They will only be able to blow out the tallest of the candles. Count how many candles are tallest.

**Example**

The maximum height candles are  units high. There are  of them, so return .

**Function Description**

Complete the function birthdayCakeCandles in the editor below.

birthdayCakeCandles has the following parameter(s):

* *int candles[n]*: the candle heights

**Returns**

* *int*: the number of candles that are tallest

**Input Format**

The first line contains a single integer, *n* , the size of *candles[]* .  
The second line contains *n* space-separated integers, where each integer  describes the height of *candles[i]* .

**Constraints**

* *1 <= n <= 105*
* *1 <= candles[i] <= 107*

**Sample Input 0**

4

3 2 1 3

**Sample Output 0**

2

**Explanation 0**

Candle heights are 1 2 3 . The tallest candles are 3 units, and there are  2 of them.

1. Note: You do not have to be concerned with this if you feel you have not covered some concepts like Collection classes/interfaces and Maps. Create two classes. The first class is called Student with two variables. The second class class Driver that contains a main method to make it a Java application.

**public** **class** Student {

**int** score;

String name;

// necessary constructors or other code

}

class Driver {

public static void main(String[] args) {

// Create at least 6 Student(s).

}

}

* 1. The scores cannot be duplicates so use a data structure that makes sense here to store the Student objects.
  2. Make sure you add any appropriate methods and constructors that conform to good programming practice.
  3. Sort the data structure by score. How do you do this? Investigate whether the structure you use can use a method to compare objects.
  4. Can you do this at least two ways? Method 1: Student implements an interface. What does this mean? Method 2: Use a lambda expression when you create the data structure. This is functional programming. Review this. Remember functional interfaces.

1. Think of an algorithm that sums the consecutive integers from 1 to that integer. The function will accept the integer of interest as a parameter. Example: The function is sum(int n). So sum(5) returns 1 + 2 + 3 + 4 + 5 = 15. Your algorithm should be much more efficient than O(n) where n is the maximum integer in the sum. Find the “big O” for your algorithm. Then based on your knowledge prove in code that big O for a bubble sort in O(n^2) -> big O n squared where n is the number of elements in an array or Collection. If you are not aware of bubble sort search for this algorithm. The upper limit of the efficiency will be for an array that is sorted in descending order assuming you want to sort it in ascending order.
2. Prove in code that a binary search is in the order of O(log n) where n is the number of elements to search. What sort algorithm do you want to use before you use binary search and what is your rationale. You don’t have to code the sort algorithm from scratch but figure out why you may choose one algorithm over others and how it works if you use it, including recursion. This can be a selection sort, insertion sort, quick sort or some other sorting algorithm.
3. You have two buckets. One has a volume of 3 gallons (liters if you prefer) and the second bucket has a volume of 5 gallons (liters). Somebody requests you to deliver exactly 4 gallons (liters) to them in the larger bucket. How can you do this? You are allowed to add or empty any amount from each bucket until you have 4 gallons. After you figure this out try to determine if this is an algorithm. Why or why not? If the steps you used can compose an algorithm then can you actually code this?
4. What is the search in an array in terms of big O? What does this mean? Answer this for HashMap also. Explain. What about a linked list in terms of best and worst case?
5. Sedgewick, Exercise 5.79). Give the preorder, inorder, postorder, preorder and level-order traversals of the following binary trees.

|  |  |  |
| --- | --- | --- |
|  |  |  |

1. Write a function that counts the number of items in a binary tree.
2. Write a function that returns the sum of all the keys in a binary tree.
3. Write a function that returns the maximum value of all the keys in a binary tree. Assume all values are nonnegative; return -1 if the tree is empty.