Algorithms Problem Sheet (Java)

Assignment weighting:

This problem sheet is worth **20%** of the total marks for this module. You must attempt all questions on this problem sheet; the marks allocated to each question are displayed overleaf.

Due date:

This assignment is due by **midnight on 21st March 2019**. Extensions to the due date may be possible in certain circumstances, but only if an extension is requested **before** the due date.

Please be aware that late submissions may be penalised, and therefore will not be eligible to receive full marks.

Submission instructions:

Where code samples are requested, the code should be neatly laid out and formatted and commented appropriately.

Where diagrams are requested, computer-generated diagrams, as well as clear and legible scans or pictures of **neat** hand-drawn diagrams are acceptable.

Where an explanation or discussion is requested, your answer is expected to be spell-checked, neatly laid out, and to use correct and appropriate grammar and terminology.

Your answers to the problem sheet questions are to be uploaded to Moodle in a single .zip folder (**NOT** in a .rar, tar.gz, .7z etc.), with the naming convention g00123456.zip, where g00123456 is your student number.

Failure to adhere to these submission instructions may lead to penalties being applied.

Note on plagiarism and copying:

Plagiarism is passing off the work of another person as one's own.

While you are allowed to collaborate with your classmates and review online and print resources for high-level problem solving and background research, you are each expected to code, write and complete this assignment **individually**. If you use material from an external source (e.g. textbook, webpage, lecture notes) as part of your answer(s), you must explicitly acknowledge the source of the material.

Please see Section 4 of the GMIT Code of Student Conduct 2018/2019 for further information on plagiarism: https://www.gmit.ie/sites/default/files/public/general/docs/7-1-code-student-conduct-2018-2019.pdf

Plagiarism is a serious academic offence and may lead to a loss of marks and/or disciplinary proceedings if it is detected in your submission.

Question 1 (3 marks)

Consider the following method:

```
public static void mystery (int n) {
    System.out.print(n);
    if (n < 4) {
        mystery(n+1);
    }
    System.out.print(n);
}
```

What will the output of the call `mystery(1);` be?

Write an explanation of the reasoning behind your answer, using the aid of either a recursion trace diagram or a stack diagram. Include any code which you write for testing or explanation purposes as part of your answer.

Question 2 (8 marks)

Consider the following methods:

```
public static int finder(int[] input) {
    return finderRec(input,input.length-1);
}

public static int finderRec(int[] input, int x) {
    if(x==0) {
        return input[x];
    }

    int v1 = input[x];
    int v2 = finderRec(input, x-1);

    if(v1>v2) {
        return v1;
    }
    else {
        return v2;
    }
}
```

Q2 (a) What is the output of a call to `finder` when the following array is used as input? **(1 mark)**

[0,-247,341,1001,741,22]

Q2 (b) What characteristic of the input data set does the `finder` method determine? How does it determine this result? **(3 marks)**

Q2 (c) Can you add some inline comments to the code above to explain how it works? **(2 marks)**

Q2 (d) Write a method which achieves the same result as `finder`, but which uses an iterative approach instead of recursion. **(2 marks)**

Write an explanation of the reasoning behind your answers to the above questions, using the aid of either a recursion trace diagram or a stack diagram for Q2 (b). Include any code which you write for testing or explanation purposes as part of your answer.

Question 3 (9 marks)

Consider the following method which checks if an array of integers contains duplicate elements:

```
public static boolean containsDuplicates(int[] elements) {
    for (int i=0; i<elements.length; i++){
        for (int j=0; j<elements.length; j++){
            if(i == j){ // avoid self comparison
                continue;
            }
            if(elements[i] == elements[j]) {
                return true; // duplicate found
            }
        }
    }
    return false;
}
```

Q3 (a) What is the best-case time complexity for this method, and why? **(2 marks)**

Q3 (b) What is the worst-case time complexity for this method, and why? **(2 marks)**

Q3 (c) Modify the code above, so that instead of returning a boolean indicating whether or not a duplicate was found, it instead returns the number of comparisons the method makes between different elements until a duplicate is found. **(2 marks)**

Q3 (d) Construct an input instance with 5 elements for which this method would exhibit its best-case running time. **(1 mark)**

Q3 (e) Construct an input instance with 5 elements for which this method would exhibit its worst-case running time. **(1 mark)**

Q3 (f) Which of the following input instances, `[10,0,5,3,-19,5]` or `[0,1,0,-127,346,125]` would take longer for this method to process, and why? **(1 mark)**

Write an explanation of the reasoning behind your answers to the above questions. Include any code which you write for testing or explanation purposes as part of your answer.