Coláiste na Tríonóide, Baile Átha Cliath
Trinity College Dublin
Ollscoil Átha Cliath | The University of Dublin

CS2011-901

Faculty of Engineering, Mathematics & Science

School of Computer Science and Statistics

Integrated Computer Science Programme **Semester 1, 2021**
BA (Mod) CSLL
BA (Mod) Business & Computing
BA MSISS
Year 2 Examinations

CS2011: Algorithms and Data Structures I

16 December 2021 **online exam** **Two hour exam**

Dr. Vasileios Koutavas

Instructions to Candidates:

- Answer ALL SIX questions.
- The marks assigned to each question are shown next to the question.

Materials permitted for this examination:

- Open books exam.

Question 1 [0 marks]

In some of the following questions you will be asked to provide an answer which depends on the following numbers. Please calculate the numbers below:

- **SID**: this is your TCD student number. It should be an 8-digit number.
- **SID²**: this is **SID** squared; it should be a 15-digit number.
- $\mathbf{d_1}, \mathbf{d_2}, \mathbf{d_3}, \mathbf{d_4}, \mathbf{d_5}, \mathbf{d_6}, \mathbf{d_7}, \mathbf{d_8}, \mathbf{d_9}, \mathbf{d_{10}}, \mathbf{d_{11}}, \mathbf{d_{12}}, \mathbf{d_{13}}, \mathbf{d_{14}}, \mathbf{d_{15}}$: these are the fifteen digits of **SID²**, from left to right.
- $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}, \mathbf{u_5}, \mathbf{u_6}, \mathbf{u_7}, \mathbf{u_8}, \mathbf{u_9}, \mathbf{u_{10}}$: these are the first 10 **unique** single-digit numbers of the d_i sequence, from left to right. If there are less than 10 unique numbers in your d_i sequence, you should complete **the remaining** u_i sequence with single-digit numbers in ascending order. In the end, the u_i sequence should not contain duplicate numbers.

Write down the $\mathbf{u_1}, \ldots, \mathbf{u_{10}}$ numbers in the answer box on blackboard and on a piece of paper in front of you.

Question 2 [20 marks]

Consider a hash function h with the following results for single-digit numbers:

$h(0) = 0$	$h(1) = 5$	$h(2) = 5$	$h(3) = 0$	$h(4) = 4$
$h(5) = 3$	$h(6) = 4$	$h(7) = 0$	$h(8) = 5$	$h(9) = 5$

Consider the implementation of a **hashtable with linear probing** with an internal array size 10. Show the state of the array after inserting each of the numbers `u1, u2, u3, u4, u5, u6, u7, u8, u9, u10` from Question 1. That is, you should show ten snapshots of the array. *[20 marks]*

Question 3 [20 marks]

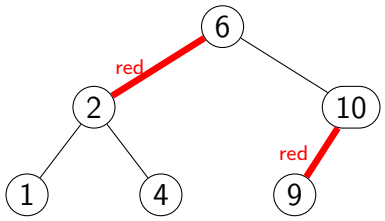
This question on Blackboard will give you a sequence of keys in a Binary Search Tree (BST), printed in **post-order**. You must

- draw the BST that produced this post-order sequence of keys and
- give the pre-order sequence of keys of this BST.

[20 marks]

Question 4 [20 marks]

Give the logical representation of the **left-leaning red-black tree (LLRB tree)** obtained by inserting the sequence $u_1, ..., u_{10}$ from Question 1 (Question 1) to the following LLRBT:

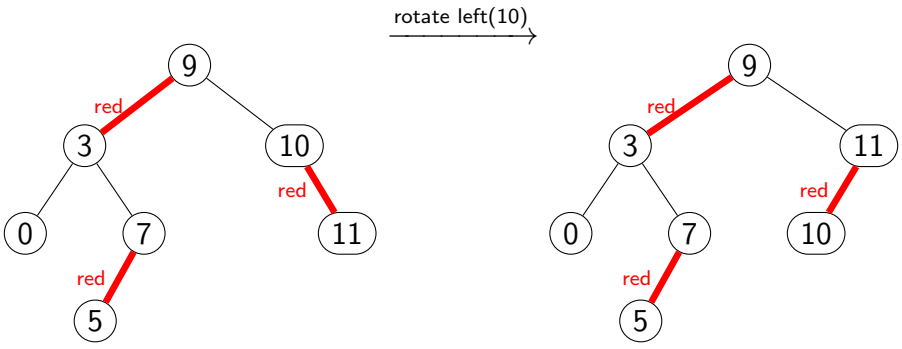


If one of the u_i is already in the tree you should not insert it again. Therefore you will need to perform five insertions.

For every insertion you should show:

- the tree after inserting the key but before any tree transformations are applied.
- which transformations should be applied, around which node, and in what order.
- the tree after each transformation.

For example inserting 11 in the above tree should be shown as:



[20 marks]

Question 5 [20 marks]

When studying the performance of algorithms we can use different analyses. Explain the intuitive difference between the **worst-case analysis** and the **amortised analysis**. When should you choose the former and when the latter? Give one example of an Abstract Data Type implementation for which worst-case analysis is better suited, and one where amortised analysis is more suitable. *[20 marks]*

Question 6 [20 marks]

Given a non-negative integer c, the following method `hasSquareSum` returns **`true`** if there exist integers a and b such that $a^2 + b^2 = c$.

```
public class Solution {
    public boolean hasSquareSum(int c) {
        for (int a = 0; a * a <= c; a++) {
            int b = c - (int)(a * a);
            if (binary_search(0, b, b))
                return true;
        }
        return false;
    }

    private boolean binary_search(int start, int end, int n) {
        if (start > end)
            return false;
        int mid = start + (end - start) / 2;
        if (mid * mid == n)
            return true;
        if (mid * mid > n)
            return binary_search(start, mid - 1, n);
        return binary_search(mid + 1, end, n);
    }
}
```

Analyse the performance of this code with respect to time and space. You are free to use any one of the analyses that have been covered in the module. You should clearly identify the analysis you use and sufficiently explain your answer. Your space analysis should take into account the heap space necessary for the recursive calls of `binary_search`. *[20 marks]*