Question 1. [9 MARKS]

Consider the following Python code which simulates rolling a pair of dice and counting the number of rolls until we reach n pairs. We are interested in the number of times the `random.randint` method is called, which corresponds to the number of rolls. *Do not express the complexity in O notation but give exact expressions.*

```
from random import randint

# Attempt to roll n pairs. Use a maximum of 10n rolls.
def countRolls(n):
    count = 0
    tries = 0
    while count < n and tries < 10*n:
        die1 = randint(1,6)      # roll the first die
        die2 = randint(1,6)      # roll the second die
        if die1 == die2:
            count += 1
        tries += 2       # count these 2 rolls even if we don't get a pair
    return count, tries
```

Part (a) [1 MARK]

Perform a best-case analysis of `countRolls`.

Part (b) [3 MARKS]

Perform a worst-case analysis of `countRolls`.

Part (c) [5 MARKS]

Perform an average-case analysis of `countRolls`. You do not need to simplify your expressions.

Question 2. [14 MARKS]

You are designing an ADT that contains information about students. Each student has a name, an age, and a score. In addition to being able to insert and delete students, you must provide the following new operation in worst case complexity $\mathcal{O}(\log n)$ where n is the number of students. You may assume that the ages are unique.

- FINDBESTWITHINAGE(*agelimit*): returns the student with the highest score from all students whose age does not exceed *agelimit*.

You will accomplish this by augmenting an AVL tree.

Part (a) [1 MARK]

What will you use as the key for the tree?

Part (b) [2 MARKS]

What additional information will you store at each node?

Part (c) [2 MARKS]

Draw a valid AVL tree for the following records showing any additional information.

Name	Age	Score
Don	8	172
Eshan	10	180
Ken	14	190
Kevin	9	165
Matt	11	185
Nick	6	167
Sam	7	169
Scott	12	187

Part (d) [4 MARKS]

Give the algorithm for FINDBESTWITHINAGE(*agelimit*) explaining briefly why it is $\mathcal{O}(\log n)$.

Part (e) [3 MARKS]

What else do you need to be concerned with when you augment this data-structure? Address this concern here.

Part (f) [2 MARKS]

The assumption that we have only one student in each category is ridiculous. What would have to change in the data structure or algorithms to accommodate non-unique ages? How would those changes affect the runtime of the operations?