BLG 335E, Analysis of Algorithms I, Fall 2015
Project 5

Handed Out: *December 18, 2015*
Due: *December 30, 2015*

Problem: In this project, you are expected to implement certain basic Red–Black Tree operations and then augment your data structure with extra operations for order statistics.

Part A. Implementation (70 points)

1. ***(40 points)***

   Implement a basic Red–Black Tree that supports insertion and printing. You do NOT have to implement updating and deletion.

   The key for each of the nodes should be the corresponding person's age. Name and gender values should be kept as extra attributes within your nodes.

   Your insertion operation should insert a new node into the relevant position in the Red–Black Tree and then recolor and rotate existing nodes in order to meet the constraints and rebalance the tree.

   Your printing operation should use in-order traversal and recursively print the sub-trees and the root of the tree. The output should properly represent sub-trees by indentation, bracketing, or another similar method.

   Indentation example:

```
>>     ┌──Alex-27-M
>> ┌──Blair-26-F
>>     └──Casey-22-F
>> Dane-20-F
>>     ┌──Evan-19-M
>> └──Fran-18-M
>>     └──Glen-17-F
```

   Bracketing example:

```
>> [[[Glen-17-F] Fran-18-M [Evan-19-M]] Dane-20-F [[Casey-22-F] Blair-26-F [Alex-27-M]]]
```

2. ***(30 points)***

   Augment your Red–Black Tree implementation with two new methods, `nth_woman` and `nth_man`, that return the name of the n^{th} youngest woman and man respectively.

   Include the `num_women[x]` and `num_men[x]` fields in your tree nodes, respectively holding the number of women and men in the sub-tree rooted at `x`, including `x` itself. Your implementation should make use of these fields in parallel with the `size[x]` field in the `OS_SELECT` example[1]. Make sure you update these fields as necessary whenever you modify the tree by another operation.

Execution:

You are given an input file (`input.txt`) containing rows of data, each denoting a person. A name, an age value and a gender value is encoded for each person. Read the contents of the file and use your insertion operation to insert each person into your Red–Black Tree.

Your program should run from the command line with the following format:

```
./<student_ID> <input_file>
```

`input_file`: The relative path to the input file, *e.g.* '`input.txt`'.

An example execution command could be as below:

```
./040080154 input.txt
```

For your output, use your printing operation to display the final state of your Red–Black Tree after all the elements from the input file have been added. Afterwards, use the `num_women` and `num_men` methods you implemented to find and print the 2nd youngest man and the 8th youngest woman. You may display your output on the console rather than using output files.

Part B. Report (30 points)

In your report, you will be expected to address the following questions. You do NOT have to provide details about your code.

1. ***(15 points)***

   Briefly explain what you would do to correctly update the age of a person as a node in the Red–Black Tree.

.

2. ***(15 points)***

   Briefly explain what you would do to correctly update the gender of a person as a node in the Red–Black Tree.

[1] Refer to the lecture slides if you would like to review the example.

Instructions:

- All your code must be written in C++ using an object-oriented approach and able to compile and run on Linux using **`g++`**.
- Do not use external libraries such as STL.
- Submissions will be done through the Ninova server. You must submit all your program and header files. You must also submit a softcopy report.
- Each student must work individually for the project. Teamwork is not accepted!

If you have any questions, please feel free to contact Res. Asst. Umut Sulubacak via e-mail (**`sulubacak@itu.edu.tr`**). Make sure you send your e-mails well in advance of the submission deadline.