BLG35E – Analysis of Algorithms 1, Fall 2017

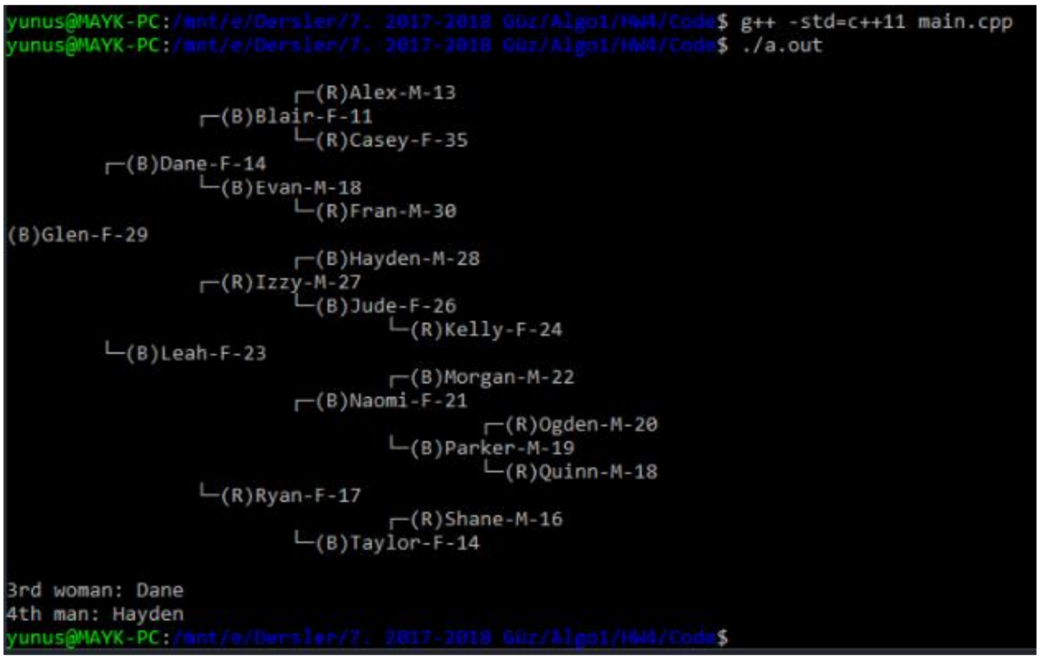
Project 4

Yunus Güngör 150150701

a. Code

Compiling Code: compiled with command: `g++ main.cpp -std=c++11`

Code Output:



```
yunus@MAYK-PC:/mnt/e/Dersler/7. 2017-2018 Güz/Algo1/HW4/Code$ g++ -std=c++11 main.cpp
yunus@MAYK-PC:/mnt/e/Dersler/7. 2017-2018 Güz/Algo1/HW4/Code$ ./a.out

                        ┌─(R)Alex-M-13
                ┌─(B)Blair-F-11
                        └─(R)Casey-F-35
        ┌─(B)Dane-F-14
                └─(B)Evan-M-18
                        └─(R)Fran-M-30
(B)Glen-F-29
                        ┌─(B)Hayden-M-28
                ┌─(R)Izzy-M-27
                        └─(B)Jude-F-26
                                └─(R)Kelly-F-24
        └─(B)Leah-F-23
                                ┌─(B)Morgan-M-22
                        ┌─(B)Naomi-F-21
                                        ┌─(R)Ogden-M-20
                                └─(B)Parker-M-19
                                        └─(R)Quinn-M-18
                └─(R)Ryan-F-17
                                ┌─(R)Shane-M-16
                        └─(B)Taylor-F-14

3rd woman: Dane
4th man: Hayden
yunus@MAYK-PC:/mnt/e/Dersler/7. 2017-2018 Güz/Algo1/HW4/Code$
```

Figure 1: Code output

b. Updating keys and Incrementing ages

1. Updating keys

Updating a person's name in this project, would cause a key update, because names used as keys in this implementation. To change a key, one must perform delete operation on the existing data, and insert the same data with different name to the tree. Insert procedure is explained and implemented in the source code. However, delete procedure only explained in this report. To delete a node in red-black tree, node will be deleted with simple binary tree delete. Simple binary tree delete procedure replaces the deleted node with inorder successor. If deleted node is a leaf, it will be deleted without any further operation, if the deleted node has one child it will be replaced with its child. But if the deleted node has 2 children's, smallest element on right tree (inorder successor) will be exchanged with the deleted node and then deleted node will be removed.

Red black trees, must comply with coloring rules and black height rule. Removing a node with simple binary tree delete will cause an exception. If replaced node or deleted is red, a simple recoloring will solve the exception. However, both of the effected nodes are black, black-height will change, and an exception will occur. In this case sibling of the deleted node will be checked. Consider that, in simple binary tree delete, only leafs are deleted, if node wants to be deleted has two children it will be replaced with inorder successor and newly exchanged leaf will be deleted. If the deleted leaf's sibling and sibling's children are black recoloring will solve the exception. If sibling is black and sibling's both child is red, one rotation will solve the exception, but if sibling is black and only one of sibling's child is red, two rotations and recoloring will be required to solve the exception. If the sibling is red, rotation and recoloring will make sibling and sibling's children black and this case explained above.

2. **Incrementing ages**

   Since age belongs to satellite data on nodes and has no effect on tree order or shape, traversing the tree one by one, and incrementing each node's age one by one would be a sufficient solution.