Due Date: 22.11.2004, until 23:59.

Submission: https://submit.ceng.metu.edu.tr

You will write a program to implement and compare both binary search tree and red-black tree for insertion and query operations. For comparison purposes you will use a counter variable to keep track of the number of operations performed for insertion or query and you will print it out after each command. Each statement and comparison necessary for the specified operation (insertion or query) is counted as 1. When the operation performed, your program will print the result of the operation and the number of operations performed. Answer to a query might be YES or NO and the answer for an insert operation is always OK. When user wants to quit (by writing “exit”) you will print the average number of operations for insertion and average number of operations for querying and then exit the program.

Here is an example of counting statements and comparisons. Below is a linear search function that searches the array “A” of size “size”, for the key “key”.

```
int Lsearch(int A[], int size, int key)
{
    int op_count = 0;
    int i = 0;
    while(A[i] != key)
    {
        op_count++; /* for comparison A[i] != key */
        i++; op_count++; /* for i++ */
        if (i == size)
            break;
        op_count++; /* for i==size */
    }
    op_count++; /* for the last comparison of while */
    return i;
}
```

Of course you don’t have to increment op_count one by one. If you know that you have m statements inside your loop you can perform op_count+=m at each iterations.

Specifications:

- First of all, stick with the given specifications for I/O. Do NOT add or remove anything. Otherwise you will lose 50 points, no matter how slight your change is.
- You will write in C and your code should be compiled with gcc on a standard Linux machine.
Your program will accept a command line argument that specifies the type of search tree to be used throughout that run. Argument is either bst or rbt, standing for binary search tree or red-black tree, respectively.

You will read from standard input and you will write to standard output (Not stderr). Note that keys to be inserted are integers.

Sample Run:

```
>hw2 bst       % use binary search tree
i 5            % insert 5
OK 4           % 5 is inserted by using 4 operations.
i 14
OK 7
i 32
OK 11
q 40           % query 40 in the tree
NO 5           %40 is not in the tree and 5 operations are performed to decide
q 14           % query 14
YES 4          % 14 is in the tree and found just with 4 operations.
i 20
OK 15
exit
QA 4.5         % QA(query average) = Total query ops/all queries  (9/2 here)
IA 9.25        % IA(insertion average)=Total insertion ops/all insertions (37/4)
```

Note that in the above run, number of operations performed are made-up. Your program does not necessarily print these numbers. The call with red-black tree is similar except the command line argument. ( > hw2 rbt )

Good Luck