CS 226: Asn1 Coding Hypothesis

Test Datasets (results):

1. Trial 1: Test10.txt

```
Reading result from test10.txt
Black nodes = 8 Red Nodes = 2
Red Node percentage is : 20
-----------------------------------------
```

2. Test100.txt

```
Reading result from test100.txt
Black nodes = 97 Red Nodes = 3
Red Node percentage is : 3
-----------------------------------------
```

3. Test1000.txt

```
Reading result from test1000.txt
Black nodes = 994 Red Nodes = 6
Red Node percentage is : 1
-----------------------------------------
```

4. Sample single trial of randomized insertion into 3 empty RedBlackBSTs with 10^4, 10^5, 10^6 keys (view note 2 for details on the 100 remaining trials)

```
-----------------------------------------
inserting 10^4 Random keys
Black nodes = 7484 Red Nodes = 2516
Red Node percentage is : 25
-----------------------------------------
inserting 10^5 Random keys
Black nodes = 74413 Red Nodes = 25542
Red Node percentage is : 26
-----------------------------------------
inserting 10^6 Random keys
Black nodes = 742680 Red Nodes = 252443
Red Node percentage is : 25
-----------------------------------------
```

Hypothesis:

Consider a Red-Black BST as two being divide into two parts:

part a) ***Odd tree levels/height***

part b) ***even tree levels/height***

Each height level contains half of the tree nodes n, therefore each part (a and b) contain n/2 nodes. All even levels will have half their nodes as red by having each node at that said level containing a left right edge/link (which is the maximum reds we can have without breaking the properties of Red-Black trees). Consequently, we can say on odd height levels we have:

1/2 * n/2 = n/4 red nodes. (and n/4 black nodes)

Looking at the even tree height levels, we can not have any red edges/links on either left or right since they break our property and the leaf leaning pattern. Consequently, we can say on even height levels we have:

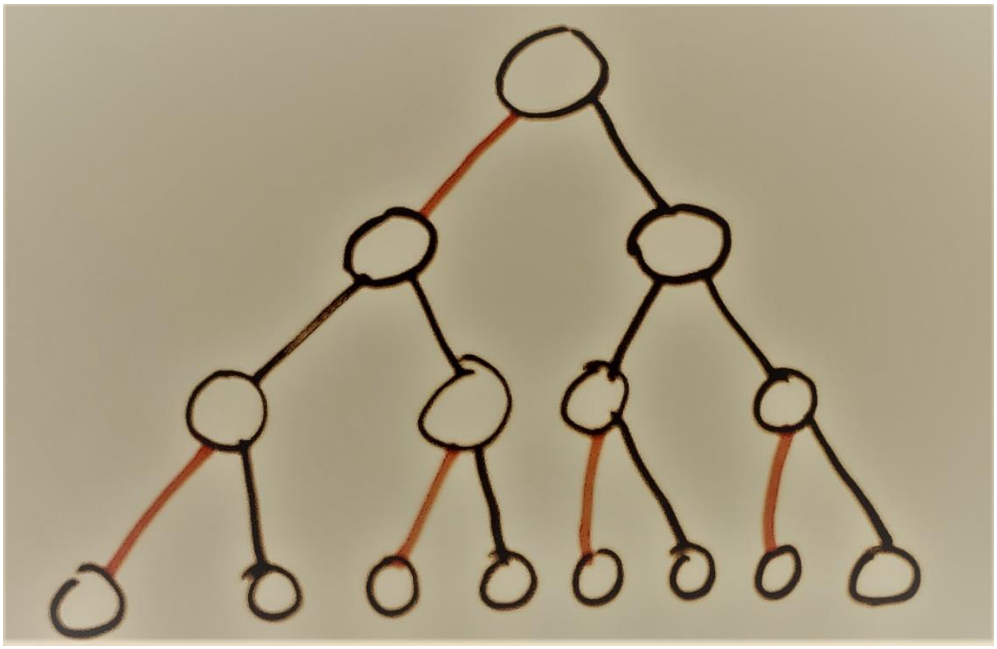
0 * n/2 = 0 red nodes. (and n/2 black nodes)

Total Red nodes = n/4 + 0 = n/4

Total Black nodes = n/4 + n/2 = 3n/4

Adding up both part a and b after deconstructing it (visually) allows us to see that just one quarter (25%) of the tree is actually red.

Red node percent reaches up to 25% and then halves when splitting the root and then gradually increases back to 25.



Note 1: The reason the code generates 26 percent is due to the rounding up of 25.5…. %; therefore, we can further generalize and say that Red-Black BSTs can wiggle around 25%.

Note 2: For the randomized insertions of 100 trials, I submitted a separate txt file “100_results” which I redirected the results into using: RedBlackBST > 100_results.txt

```
C:\Users\Attar\Desktop\CS 226\asns\asn1 coding>java RedBlackBST >100_results.txt
```

No piping sample result:

```
C:\Users\Attar\Desktop\CS 226\asns\asn1 coding>java RedBlackBST
there was an error reading the file
----------------------------------------
inserting 10^4 Random keys
Black nodes = 7467 Red Nodes = 2533
Red Node percentage is : 25
----------------------------------------
inserting 10^5 Random keys
Black nodes = 74506 Red Nodes = 25444
Red Node percentage is : 25
----------------------------------------
inserting 10^6 Random keys
Black nodes = 742733 Red Nodes = 252293
Red Node percentage is : 25
----------------------------------------
inserting 10^4 Random keys
Black nodes = 7434 Red Nodes = 2566
Red Node percentage is : 26
----------------------------------------
inserting 10^5 Random keys
Black nodes = 74630 Red Nodes = 25325
Red Node percentage is : 25
----------------------------------------
inserting 10^6 Random keys
```

This message is because of providing no arguments in args[0] intentionally (since it's not required for the 100 trials) It was caught with the catch statement I wrote which outputs that message. This allows me to not have to comment the file reading part out when not reading files (i.e just trial testing)