Assignment 2

Name:Tian Zhou

NETID:tz164

Q1.

The program of Q1 is in the Q1&2 file.
“leanLeft()” function is used to lean the 2-3 tree to the left.
”leanRight()” function is used to lean the 2-3 tree to the left.
“insert()”function is used to insert a node to the tree.
“print()”function is used to print the tree in order. (ascending)

Q2.

This question also use the program in the Q1&2 file.
(i)
Run the program by inserting 10^1, 10^2, 10^3, 10^4, 10^5 random Nodes to the tree. Each value of N was tested 5 times the get the result of average path length:

Table 3-2-1

N		100	1000	10000	100000
logN		2	3	4	5
result of experiment (Path Length)	1	367	5975	93424	1212913
	2	417	6144	93696	1190652
	3	385	5995	89958	1218106
	4	329	6322	90786	1201477
	5	355	6648	93079	1171750
average of result		371	6217	92189	1198980
average for each node		3.71	6.22	9.22	11.99
3*N*(logN-1)		300	6000	90000	1200000
3*(logN-1)		3	6	9	12

As we can see in the table, the estimating value of average path length is **$3*(\log_{10}N-1)$** or **O(lgN)** .

For N-random insertions (assume N is large), the estimation of average path length is **$3*(\log_{10}N-1)$.**

(ii)
N-sorted insertions.
N=20,200,2000,20000

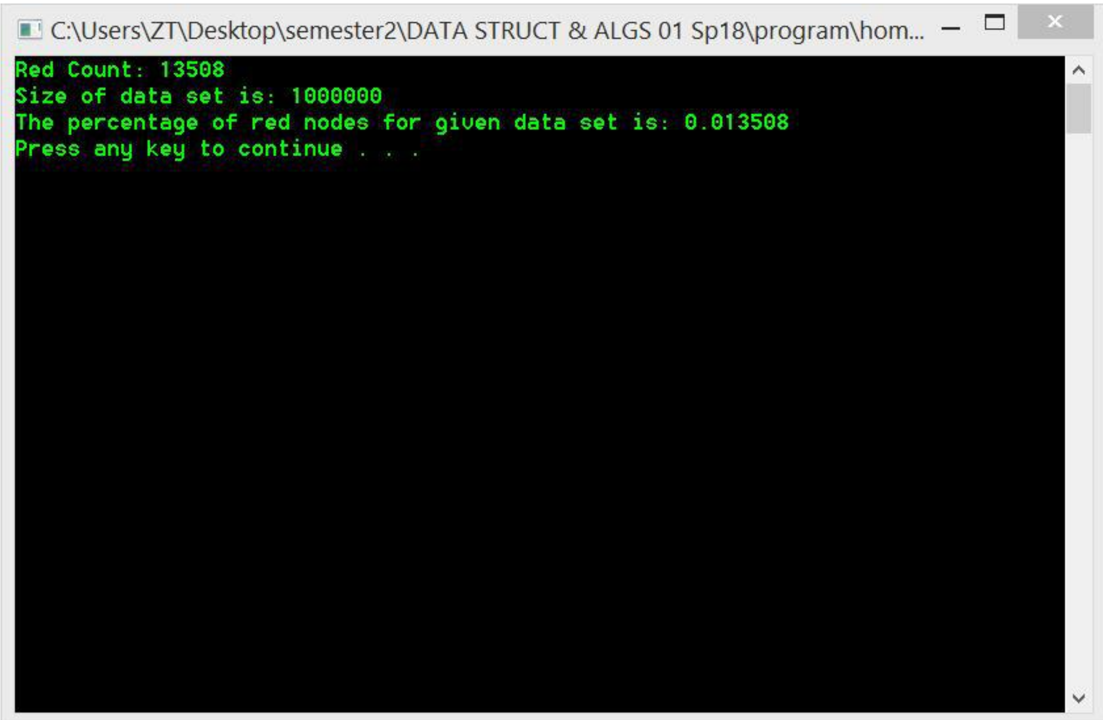
Table 3-2-2

N	20	200	2000	20000
	2*10^1	2*10^2	2*10^3	2*10^4
log(N/2)	1	2	3	4
result(Path Length)	90	9900	999000	99990000
	10^2-10^1	10^4-10^2	10^6-10^3	10^8-10^4
Path Length=N/2*(N/2-1) Average Path Length=1/2*(N/2-1)				

Since the program will hook the new node onto the bottom with a black link when inserting into a 3-node at the bottom, the N-sorted insertions could have a large value of average path length.
As we can see in the table, the estimating value of average path length is 1/2*(N/2-1) or **O(N)** .

Q3.

Compute given red-black tree.
The percentage of red nodes in a given red-black tree is 0.013508.
The size of given data is 1000000.



100 trials for each size of data: N=10^4, 10^5, $2*10^5$, 10^6.

The average of 100 trials result is:

Table 3-3-1

10000		100000	
count	percentage	count	percentage
4864.87	0.486487	46927.09	0.4692709
200000		1000000	
count	percentage	count	percentage
83482.66	0.4174133	188933.13	0.18893313

As we can see in the table, when the size of data grows up, the percentage of red nodes descends.

Q4.

Table 3-4-1

10			100		
std deviation		0.145804664	std deviation		0.08600387
total	average	variance	total	average	variance
21.99	2.199	0.021259	541.312	5.41312	0.007396666
1000			**10000**		
std deviation		0.040609382	std deviation		0.030961996
total	average	variance	total	average	variance
8774.01	8.77401	0.001649122	121647.418	12.1647418	0.000958645

Each size of data has 1000 trials to compute the average length of a path for a random node.The result of average and standard deviation for each size is show in the Table 3-4.

(More specific data of trials is in the “homework3.xlsx” file.)

Depends on the average value, we can assume the average length of a path for size N is approximately : $\log_2 N - 1$

Table 3-4-2

Size	Average	lgN-1
10	2.199	2.322
100	5.413	5.644
1000	8.774	8.966
10000	12.165	12.288

For the standard deviation: It’s obvious that when the size of data is more larger ,the value of standard deviation is more smaller, which means the result of random samples insertion become more stable.

Q5.

The program use the data set in “select-data.txt” file. (already include in the Q5 file)

The implementation of rank() and select() ordered operation for a BST is in the program 3_5 in Q5 file. Total size of data size is 1000000 (10^6).

(i) The value of select(7) for the data set is 1.
(ii) The value of rank(7) for the data set is 6105.

Both rank() and select() functions use recursive to find the key or the rank. Once find the request node, the function will not run right part.