05.12.2020

CENG 311

HOMEWORK 2

REPORT

Arif Burak Demiray-250201022

Table of Contents

1. Implementation Details

The tree is stored in s0 register so I am using s0 when calling the tree. There are a lot of subprocesses but here are the 4 main processes that must be mentioned. More details are in the code file. The main procedure where the menu and other procedures is not mentioned below. They all explained detailed in bst.asm.

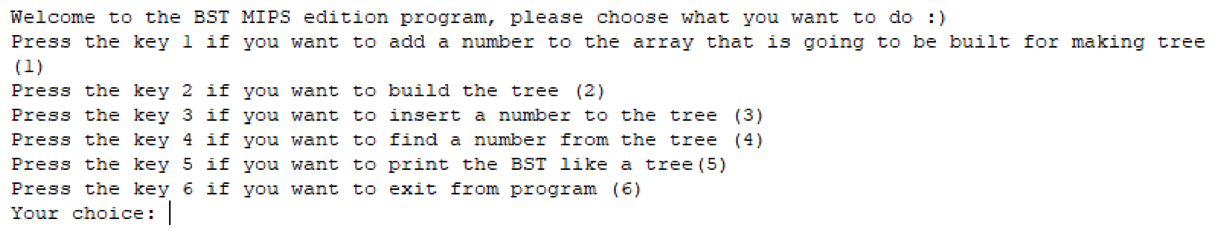
a. **Build Procedure**: In build procedure, I am firstly making root manually to start the process of adding children. Then, I start to look for greatness or lessness. In my implementation, I allow duplicate values. I add them to the left child of the same parent. After making the root, it goes to next element of the list. Look for if it is lesser, adds it to the right child then moves on to next element of the list. Loop breaks when MAXINT value readed which is -9999 for my implementation. When adding child and make root case, it requests 16 byte place from qtsim with sbrk system call. It returns the root address in s0 register. Also, I am storing the size of the BST in s7 register. I am storing parents in t7 register to move iteratively easily. More details are in the bst.asm.

b. **Insert Procedure**: I assumed that the value is stored at a0 register. The root adress is stored at a1 register. I implemented this procedure using the build procedure. In t9 register, I store it with 1. When it enters to build, in some parts I added when t9 = 1. In insert, I need just it should find its place and sits there. So it enters to build from half of the procedure where it checks existing value is greater or lesser or equal. It compares their values then goes to setLeft or setRight procedures. In that procedures, it looks if the parents left or right child empty. If not it changes the parentery to its child. If empty, it takes place from system call and adds it to current parent's child. Then when t9 = 1 it returns from insert. If t9 not 1 it continues to build the tree. More details are in the bst.asm

c. **Find Procedure**: I assumed that the value is stored at a0 register. The root adress is stored at a1 register. It returns 1 in v0 and 0 in v1 if value not found. It returns 0 in v0 if value found and returns address of the node in v1. I used build procedure to implement find procedure. I store 2 in t9 to show we are using build procedure to find an element. It again enters to build from half. Where looking for lessness and greatness. When t9 is 2 it just changes parents until finds the element, if it can not find returns values mentioned above. Because I am allowing duplicate values as left child. I am only checking equality in setLeft procedure. In setRight, it only changes the parent to its right child and looks for if it is zero or not. If it is zero returns. It checks this conditon also in setLeft. Addition, it checks equality in setLeft procedure. More details are in the bst.asm

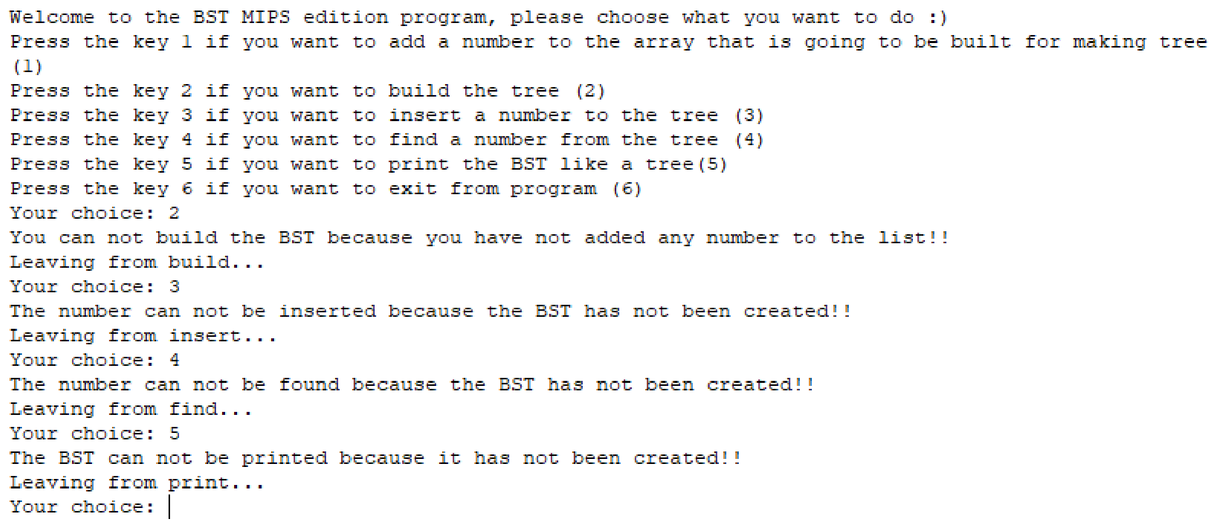
d. **Print Procedure**: This is the last main procedure. In print implementation, I used queues. Which I mentioned in references, the algorithm is not same but I got inspired from it. Firstly, I implemented a queue via linked list thanks to sbrk system call. There are a 8 byte node implementation. First 4 byte is for head’s or element’s addresses. The other 4 byte is for next element. Simply, it adds the parent to queue, then prints it. Then adds the child of it to queue in order of left child and right child. If one of them or both of them are zero, it adds X value to queue which represented by -9999 in my implementation. After adding children to queue, it dequeues one element which is the tail of the queue. The process continues until I printed the all elements in the BST. It controls for the last row’s X values. If the row’s expected count number is not same as the counted one, it continues to print X. It checks for the new line with expected count = counted count. But differently, it divides the counted one by expected one to look more clear way. Because, it checks new line firstly, then other processes continues. And , it checks for “-“ char and “ “ char top rint them or not. If counted count is odd it prints “-“,otherwise it is even, then it prints “ “. Simply detailed version of the print and main processes are. More detailed informations are in bst.asm.

2. Sample Executions



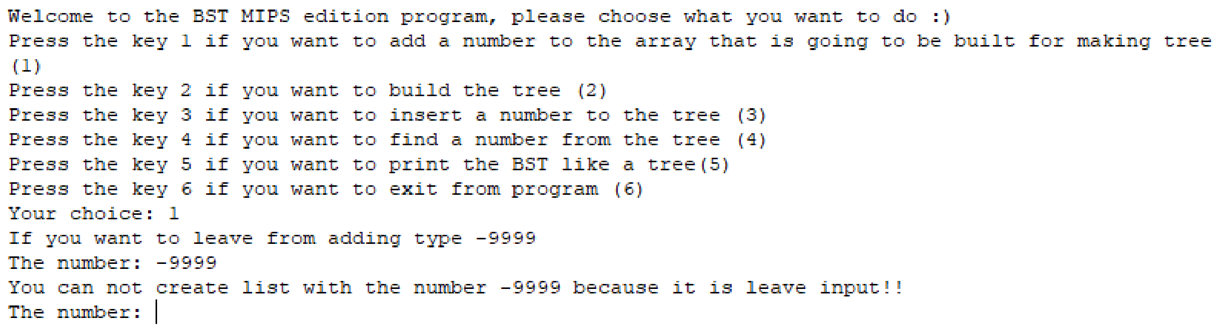
```
Welcome to the BST MIPS edition program, please choose what you want to do :)
Press the key 1 if you want to add a number to the array that is going to be built for making tree
(1)
Press the key 2 if you want to build the tree (2)
Press the key 3 if you want to insert a number to the tree (3)
Press the key 4 if you want to find a number from the tree (4)
Press the key 5 if you want to print the BST like a tree(5)
Press the key 6 if you want to exit from program (6)
Your choice: 
```

Here are the main menu for the program



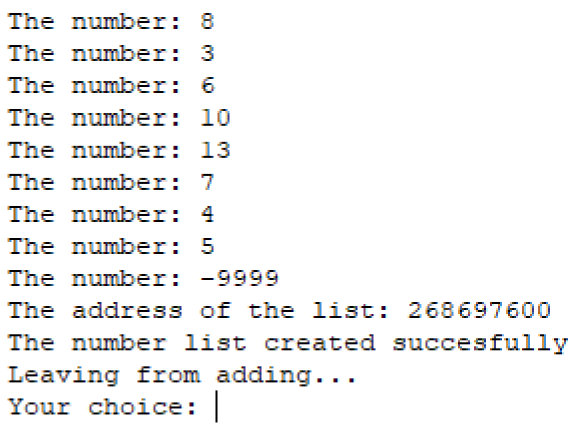
```
Welcome to the BST MIPS edition program, please choose what you want to do :)
Press the key 1 if you want to add a number to the array that is going to be built for making tree
(1)
Press the key 2 if you want to build the tree (2)
Press the key 3 if you want to insert a number to the tree (3)
Press the key 4 if you want to find a number from the tree (4)
Press the key 5 if you want to print the BST like a tree(5)
Press the key 6 if you want to exit from program (6)
Your choice: 2
You can not build the BST because you have not added any number to the list!!
Leaving from build...
Your choice: 3
The number can not be inserted because the BST has not been created!!
Leaving from insert...
Your choice: 4
The number can not be found because the BST has not been created!!
Leaving from find...
Your choice: 5
The BST can not be printed because it has not been created!!
Leaving from print...
Your choice: 
```

Here is the execution if the user did not give list and trys to do other operations



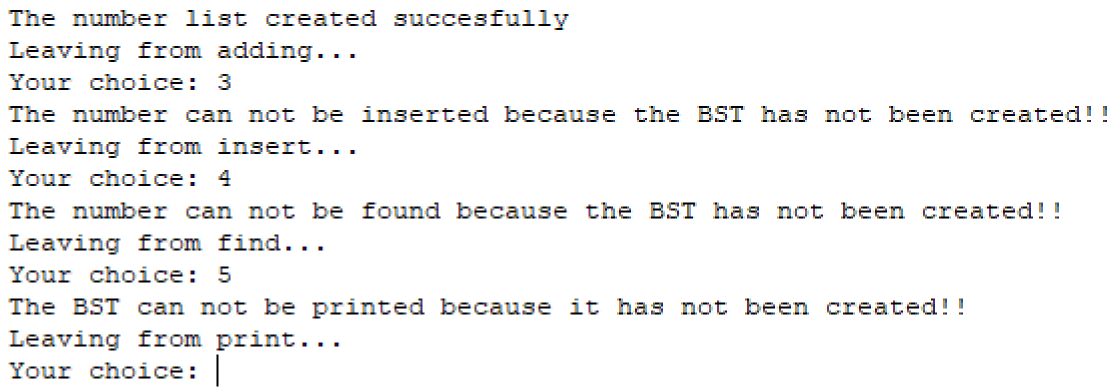
```
Welcome to the BST MIPS edition program, please choose what you want to do :)
Press the key 1 if you want to add a number to the array that is going to be built for making tree
(1)
Press the key 2 if you want to build the tree (2)
Press the key 3 if you want to insert a number to the tree (3)
Press the key 4 if you want to find a number from the tree (4)
Press the key 5 if you want to print the BST like a tree(5)
Press the key 6 if you want to exit from program (6)
Your choice: 1
If you want to leave from adding type -9999
The number: -9999
You can not create list with the number -9999 because it is leave input!!
The number: 
```

Here is the if user gives first input as -9999



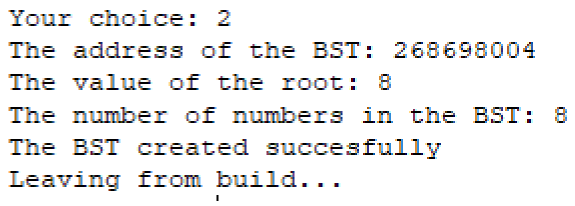
```
The number: 8
The number: 3
The number: 6
The number: 10
The number: 13
The number: 7
The number: 4
The number: 5
The number: -9999
The address of the list: 268697600
The number list created succesfully
Leaving from adding...
Your choice: |
```

Here is the creation of the list



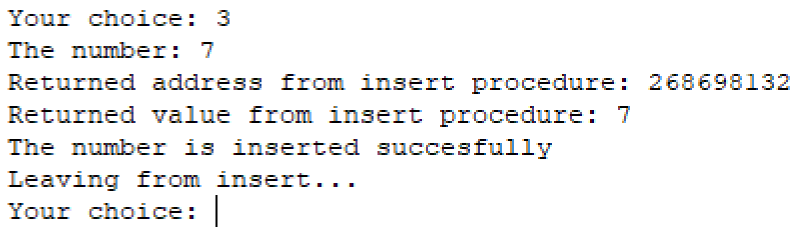
```
The number list created succesfully
Leaving from adding...
Your choice: 3
The number can not be inserted because the BST has not been created!!
Leaving from insert...
Your choice: 4
The number can not be found because the BST has not been created!!
Leaving from find...
Your choice: 5
The BST can not be printed because it has not been created!!
Leaving from print...
Your choice: |
```

The user must create the BST before doing insert,find and print.



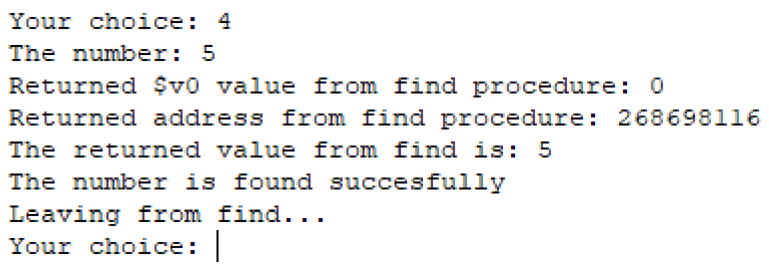
```
Your choice: 2
The address of the BST: 268698004
The value of the root: 8
The number of numbers in the BST: 8
The BST created succesfully
Leaving from build...
```

Here is the creation of the BST.



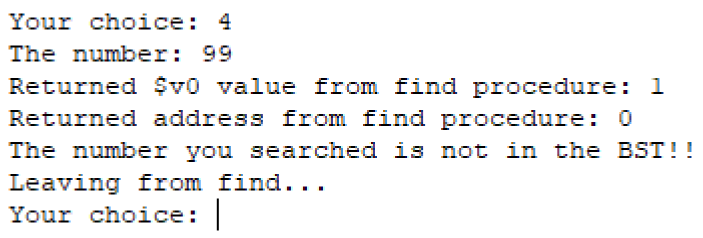
```
Your choice: 3
The number: 7
Returned address from insert procedure: 268698132
Returned value from insert procedure: 7
The number is inserted succesfully
Leaving from insert...
Your choice: |
```

Here is the insertion of and element to the list



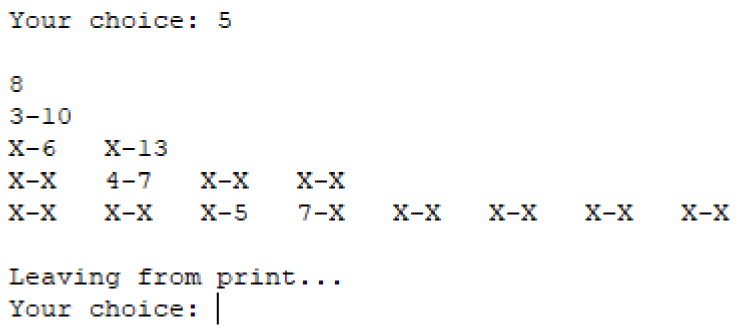
```
Your choice: 4
The number: 5
Returned $v0 value from find procedure: 0
Returned address from find procedure: 268698116
The returned value from find is: 5
The number is found succesfully
Leaving from find...
Your choice: |
```

Here is finding an element from list true case



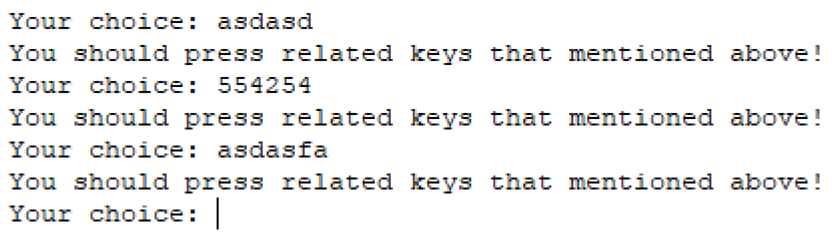
```
Your choice: 4
The number: 99
Returned $v0 value from find procedure: 1
Returned address from find procedure: 0
The number you searched is not in the BST!!
Leaving from find...
Your choice: |
```

Here is the finding an element false case



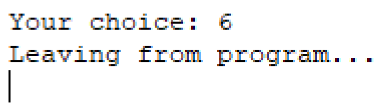
```
Your choice: 5

8
3-10
X-6   X-13
X-X   4-7   X-X   X-X
X-X   X-X   X-5   7-X   X-X   X-X   X-X   X-X

Leaving from print...
Your choice: |
```

Here is printing the BST



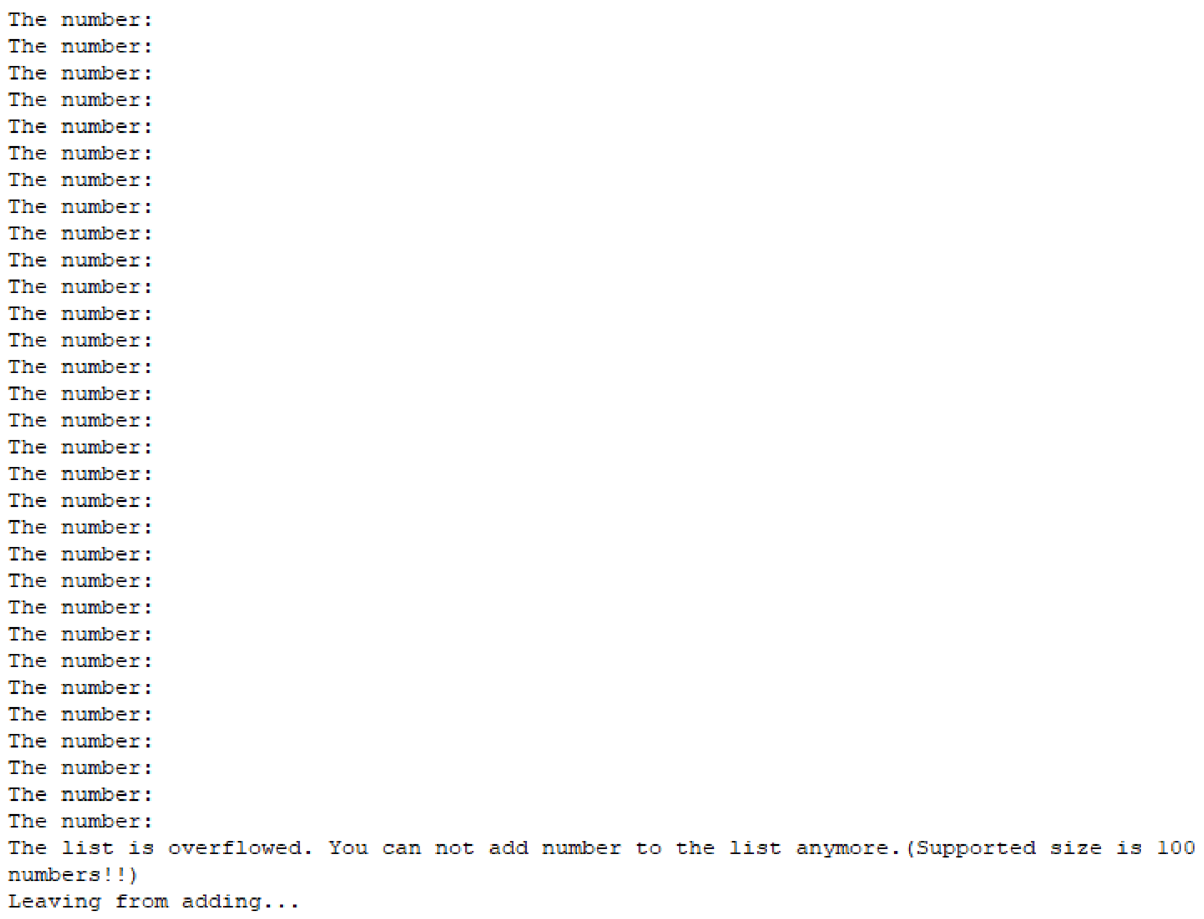
```
Your choice: asdasd
You should press related keys that mentioned above!
Your choice: 554254
You should press related keys that mentioned above!
Your choice: asdasfa
You should press related keys that mentioned above!
Your choice: |
```

Here is the user inputs unsupported prompt



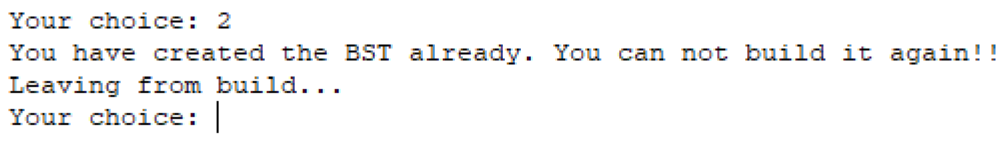
```
Your choice: 6
Leaving from program...
|
```

Here is leaving from program



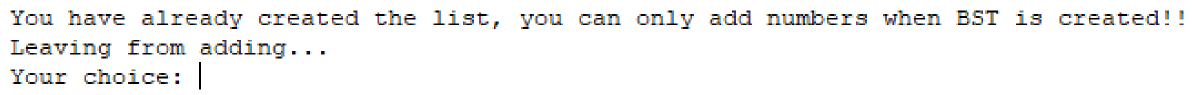
```
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The number:
The list is overflowed. You can not add number to the list anymore.(Supported size is 100
numbers!!)
Leaving from adding...
```

Here is the user can not input over 100 values



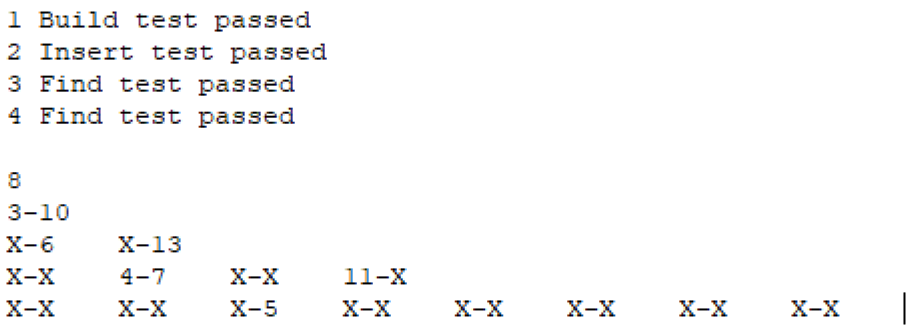
```
Your choice: 2
You have created the BST already. You can not build it again!!
Leaving from build...
Your choice:
```

Here is the user can build tree just one time



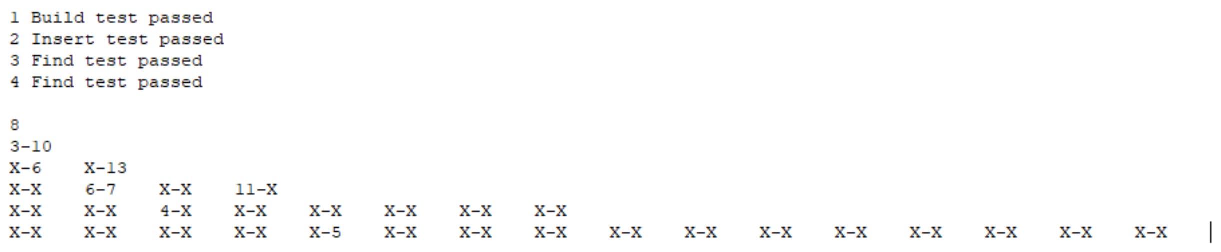
```
You have already created the list, you can only add numbers when BST is created!!
Leaving from adding...
Your choice:
```

Here is if the user trys to add number via 1 after he/she created the list



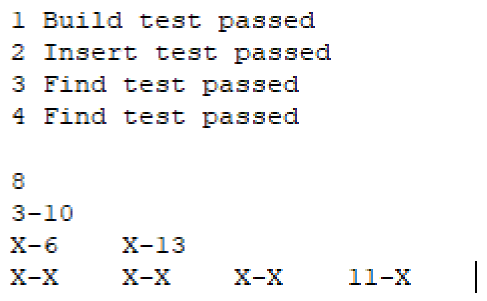
```
1 Build test passed
2 Insert test passed
3 Find test passed
4 Find test passed

8
3-10
X-6    X-13
X-X    4-7    X-X    11-X
X-X    X-X    X-5    X-X    X-X    X-X    X-X    X-X
```

Here is the test.asm with firstList



```
1 Build test passed
2 Insert test passed
3 Find test passed
4 Find test passed

8
3-10
X-6    X-13
X-X    6-7    X-X    11-X
X-X    X-X    4-X    X-X    X-X    X-X    X-X    X-X
X-X    X-X    X-X    X-X    X-5    X-X    X-X    X-X    X-X    X-X    X-X    X-X    X-X    X-X    X-X    X-X
```

Here is test.asm with secondList



```
1 Build test passed
2 Insert test passed
3 Find test passed
4 Find test passed

8
3-10
X-6    X-13
X-X    X-X    X-X    11-X
```

Here is the test.asm with thirdList

References

https://www.geeksforgeeks.org/level-order-tree-traversal/