Objective of this assignment:

- To design and implement an interview-like programming exercise about binary search trees.

What you need to do:

a) Implement the *Search and Tree-Insert(T,z)* operations on a binary search tree.
b) **Repeatedly** Insert ***n*** user provided numbers in a binary search tree. In other words, you will prompt a user to enter a new number n until the user enters -1. Do not insert the number -1.
c) **Each time** the user provides a number n:
1) Check that this number n is not already in the binary search tree. If it is, tell it to the user and ask for a new number
2) If the number n is not yet in the binary search tree, insert it and print its parent's key $p.key$ and the word *left* (resp. right) if the number n is the left (resp. right) child.
d) When done, print out the height of the binary search tree.

Do not hesitate to ask questions if you have any doubt about the requirements.

You can use **any** programming language as long as:
1) It is already **available** on Tux machines (the teaching assistant will NOT install a "new" programming language).
2) your program compiles and executes correctly on a Tux machine.

Report

- Write a report that will contain the following information:
  - whether the program works or not (this must be just ONE sentence)
  - the directions to compile and execute your program on Tux machines
- Good writing is expected.

The program `BinarySearchTree.java` works as requested above.
Directions to compile and execute on Tux:

Type `javac BinarySearchTree.java` – *this will compile the program*
Type `java BinarySearchTree` - *this will execute the program*
At this point you should be able to see the directions on your Tux.

Example provided below:

```
xterm
Enter a number: 9
Parent: NIL. Node is ROOT.
Enter a number: 3
Parent: 9, 3 is LEFT child.
Enter a number: 10
Parent: 9, 10 is RIGHT child.
Enter a number: 7
Parent: 3, 7 is RIGHT child.
Enter a number: 14
Parent: 10, 14 is RIGHT child.
Enter a number: 14
14 is already in the tree, please enter another number.
Enter a number: -1
The HEIGHT of the tree is 2.
```

What you need to turn in:

- Electronic copy of your source program (standalone)
- Electronic copy of the report (standalone). Submit the file as a Microsoft Word or PDF file.
- Do not use zipped folders

Grading

- Program is worth 90% if it works correctly and meets the requirements
- Quality of the report is worth 10%.
- If the teaching assistant cannot compile/execute your program based on the provided instructions, you will loose 30% of the credit.