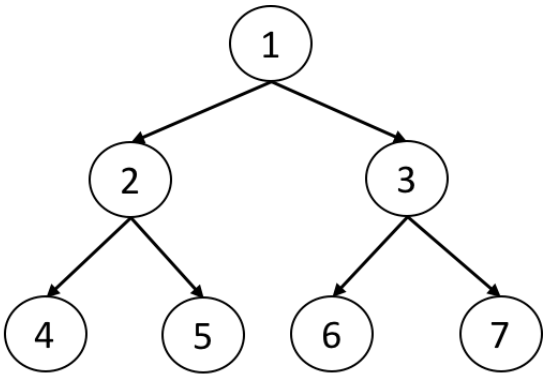
Input:

#4 types of command
node [NodeType] [Type] [Code] [Content]
checkType [NodeNum]
generateCode [NodeNum]
printContent [NodeNum]

Output:

After node command:

Construct a node in the tree, the structure of the tree will look like this (binary tree):



… … …

The node number indicates the input order.

After checkType command:

Starting from node [NodeNum], traverse nodes with depth-first search, for each node:
=> Node [NodeNum]: [NodeType] [Type]

After generateCode command:

Starting from node [NodeNum], traverse nodes with depth-first search, for each node:
=> Node [NodeNum]: [NodeType] [Code]

After printContent command:

Starting from node [NodeNum], traverse nodes with depth-first search, for each node:
=> Node [NodeNum]: [NodeType] [Content]

Comment:

[NodeType] is limited to "VariableRefNode" and "AssignmentNode".
[NodeNum] will not exceed the current number of node commands, and it is guaranteed to be a positive integer.

Please implement your main function in Class Main.
We'll test your program through "java Main inputFile"
e.g java Main sampleInput

Do not read input from System.in or hard code input file, or your program won't pass any test case.

Upload:

Please zip your source code and upload it.
The file name should be Team[teamID].zip. e.g. Team7.zip
The folder structure should be:

```
    unzip Team7.zip
=> [dir] Team7
=>          Team7/Main.java
=>          Team7/*.java (optional)
```

You won't receive any point if you didn't follow the directory structure or main class name or compressed format!

#**SampleInput:**
node VariableRefNode int code1 this_is_a_code
node AssignmentNode assign code2 this_is_seafood
checkType 1
generateCode 1
printContent 2

#**SampleOutput:**
Node 1: VariableRefNode int
Node 2: AssignmentNode assign
Node 1: VariableRefNode code1
Node 2: AssignmentNode code2
Node 2: AssignmentNode this_is_seafood