Exercise Sheet, Week 5

Trees in Mem (continue)

In Java, in order to represent a binary tree which stores integers, one can define a class

```
class Node {
    Node left;
    int value;
    Node right;
}
```

In the last tutorial, we said that we can represent trees in `Mem`. Each node then takes 3 locations in memory. In some sense, we mimicked Java's representation because whenever you write

```
node = new Node();
```

this corresponds to our:

```
node = allocate_memory(3);  // then node stores just an *address*, e.g. number 693
```

Furthermore,

```
node.left  = null;  // empty
node.value = 123;
node.right = node2;

// assume that node2 has
// been created earlier
```

corresponds to

```
Mem[node]   = END;  // empty
Mem[node+1] = 123;
Mem[node+2] = node2;

// assume that node2 has
// been created earlier
```

The concrete representation in `Mem`, might look like as follows. For example, if `node = 693` and the second node stores `555` and has no children, we might have:

i	Mem[i]
693	END
694	123
695	698
696	949193
697	419399
698	END
699	555
700	END
701	149939

Exercises:
Solve the following in the `Mem` representation. You can first write the solution to the exercises in Java and then translate them the same way as above. *Hint: Use recursion, stacks or queues.*

5-$\frac{1}{2}$. Write a function `int size(int root)` which computes the number of nodes in the tree.

5. Write a function `int sum(int root)` which computes the sum of all numbers stored in the nodes of the tree.

6. What is the time complexity of your function `sum` from (5)? Express the time complexity with respect to $n =$ the **size** of the tree.

   Does it make sense to express the time complexity in terms of the tree's height?

7. **Bonus:** Write a function `int maxLessThan(int root, int x)` which finds the largest value stored in the binary search tree which is $\leq x$.