Lab 15: Recursion

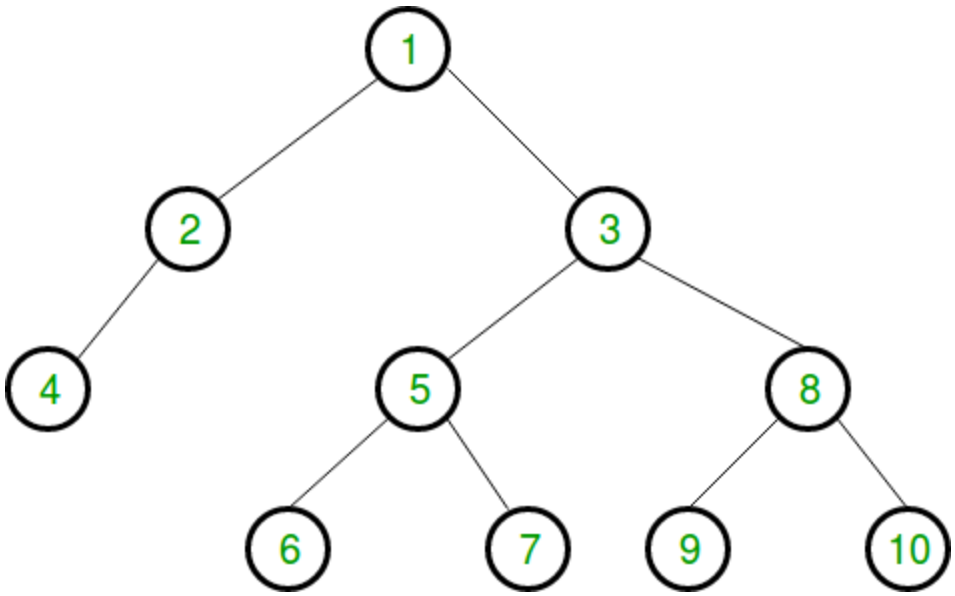
1. **Fibonacci Numbers [20 minutes]:** Code along with the GSI as we write a program to find the nth Fibonacci numbers using recursion. The Fibonacci numbers are a sequence of integers where $a_n = a_{n-1} + a_{n-2}$, starting with $a_0 = 0, a_1 = 1$. The first few numbers are `0, 1, 1, 2, 3, 5, 8, ...` and so on.
   a. First, we will just write a program to print the nth Fibonacci number
   b. Second, we will try to adapt this program to create a list of the first n Fibonacci numbers (rather than just printing the nth)
   c. Finally, we will try redoing (b) but with iteration instead of recursion to see which is easier to write and more efficient.

2. **Printing a Stack in Reverse: [10 minutes]**: Code along with the GSI as we write a program to print the values in an `std::stack<int>` in reverse order (from the bottom to the top). Create a file called `lab13_reverse_stack.cpp` and copy in the following template code. Then, fill out `print_stack_reversed`.

```
#include <iostream>
#include <stack>


void print_stack_reversed(std::stack<int> s) {
  // --- Your code here
  //
}

int main() {
    std::stack<int> s;
    s.push(0);
    s.push(1);
    s.push(2);
    s.push(3);
    s.push(4);
    // 4 is at the top of the stack, we want to print starting at the bottom
    // should print 0,1,2,3,4

    print_stack_reversed(s);

    std::cout << std::endl;
```

```
    return 0;
}
```

3. **Binary Tree Traversal [30 minutes]:** A binary tree is a tree-shaped data structure, with one “root” node, and “child” nodes branching out from the root, and an example is shown in the diagram below.



Copy the following template code into `lab15_tree.cpp`. Then, implement `printLeafNodes` using recursion. The correct output for this example is `4 6 7 9 10`. In the quiz you will be asked more about the order of traversal in this example.

```
#include <iostream>

// A Binary Tree Node
struct Node
{
    Node(int data) : data(data) {}

  int data;

  Node *left = nullptr;
    Node *right = nullptr;
};

// function to print leaf
// nodes from left to right
```

```
void printLeafNodes(Node *root)
{
  // --- Your code here
  // ---
}

int main()
{
  // create the binary tree
  Node n1(1);
  Node n2(2);
  Node n3(3);
  Node n4(4);
  Node n5(5);
  Node n6(6);
  Node n7(7);
  Node n8(8);
  Node n9(9);
  Node n10(10);
  n1.left = &n2;
  n1.right = &n3;
  n1.left->left = &n4;
  n1.right->left = &n5;
  n1.right->right = &n8;
  n1.right->left->left = &n6;
  n1.right->left->right = &n7;
  n1.right->right->left = &n9;
  n1.right->right->right = &n10;

  printLeafNodes(&n1);

  return 0;
}
```

4. **Generating All Sequence of Coin Flips [30 min]**: Write a program to generate & count all possible sequences of heads or tails coin flips of a given length. For example, if n=2 then the possible sequences are:

   `HH HT TH TT` (4)

   And for n=3 the possible sequences are:

   `HHH HHT HTH HTT THH THT TTH TTT` (8)

   What are the possible sequences for n=4? After creating all possible sequences, count the number of sequences (HINT: use `std::count_if`) which have exactly 2

heads (HINT: use `std::count_if` again). Write down the number of sequences for n=4 with exactly 2 heads.