1. A *recursive function* is one that calls itself. The following `countDown` function is passed a starting value and proceeds to count down to zero and prints “Blast Off!!”.

```cpp
#include <iostream>
using namespace std;

// prototypes
void countDown(int count);

int main() {
    int startOfCountDown;
    cout << "Enter count down start: ";
    cin >> startOfCountDown;
    cout << endl << "Count Down: " << endl;
    countDown(startOfCountDown);
} // end main

void countDown(int count) {
    if (count == 0) {
        cout << "Blast Off!!" << endl;
    } else {
        cout << count << endl;
        countDown(count - 1);
    } // end if
} // end countDown
```

The `countDown` function, like most recursive functions, solves a problem by splitting the problem into one or more simpler problems of the same type. For example, `countDown(10)` prints the first value (i.e., 10) and then solves the simpler problem of counting down from 9. To prevent “infinite recursion”, if-statement(s) are used to check for trivial *base case(s)* of the problem that can be solved without recursion. Here, when we reach a `countDown(0)` problem we can just print “Blast Off!!”.

a) Trace the function call `countDown(5)` on paper by drawing the run-time stack and showing the output.

```
Enter count down start: 10
Count Down:
10
  9
  8
  7
  6
  5
  4
  3
  2
  1
Blast Off!!
```

b) What do you think will happen if your call `countDown(-1)`?

c) Why is there a limit on the depth of recursion?
2. Write a recursive function, \texttt{power}(x, y) that takes two parameters and returns $x^y$, where $x$ is some number and $y$ is a non-negative integer. Some steps to help you:

- consider an example, say $3^5$, which means $3 \times 3 \times 3 \times 3 \times 3$
- think about how you might calculate the value of $3^5$ recursively, i.e., how you might calculate the value of $3^5$ if you knew the answer to a smaller problem say $3^4$?
- think about what base case(s) are trivial enough that the answer is obvious, i.e., what power(s) of 3 are simple to solve?

a) write the recursive function using some variation of the if-statement

b) Draw the recursion tree for \texttt{power}(3, 5).

3. Some mathematical concepts are defining by recursive definitions. One example is the Fibonacci series:

\[ 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \ldots \]

After the second number, each number in the series is the sum of the two previous numbers. The Fibonacci series can be defined recursively as:

\[
\text{Fib}_0 = 0 \\
\text{Fib}_1 = 1 \\
\text{Fib}_N = \text{Fib}_{N-1} + \text{Fib}_{N-2} \quad \text{for } N \geq 2.
\]

a) Write the recursive function

b) Draw a recursion tree for \texttt{fib}(5).

c) On my office computer, the call to \texttt{fib}(40) takes 22 seconds, the call to \texttt{fib}(41) takes 35 seconds, and the call to \texttt{fib}(42) takes 56 seconds. How long would you expect \texttt{fib}(43) to take?