Recursion

A frog stands in front of a flight of n stairs. In one jump, the frog can cover one, two or three steps. In how many ways can the frog cross all the steps? Call it $C(n)$.

For example, if $n = 4$, then all the possibilities for the frog are (1,1,1,1), (1,1,2), (1,2,1), (1,3), (2,1,1), (2,2) and (3,1). Therefore, $C(4) = 7$.

Part 1

Frame a recurrence relation for $C(n)$, and make a straightforward recursive implementation. (Write a recursive function.)

Part 2

Make an efficient (linear-time and constant-space in n) iterative implementation. (Write a non-recursive function.)

Part 3

Suppose you want to compute $C(n,m)$ which stands for the number of ways the frog can cross n steps in exactly m jumps. Derive a recurrence relation for $C(n,m)$, and write a recursive function for it.

Part 4

Make an efficient iterative function to compute $C(n,m)$. You are permitted to use only one local array of size $n + 1$, and some constant number of local variables.

The main() function

- Read n from the user. (Take n no larger than 37.)
- Run the function of Part 1 on n.
- Run the function of Part 2 on n.
- Run the function of Part 3 on n,m for all m in $[0,n]$. Report the sum of all these return values.
- Run the function of Part 4 on n,m for all m in $[0,n]$. Report the sum of all these return values.

For n above 30, you can see how slow your recursive functions are.