Week 9: Functions that call themselves

Please make sure to submit your solutions **by next Monday**.

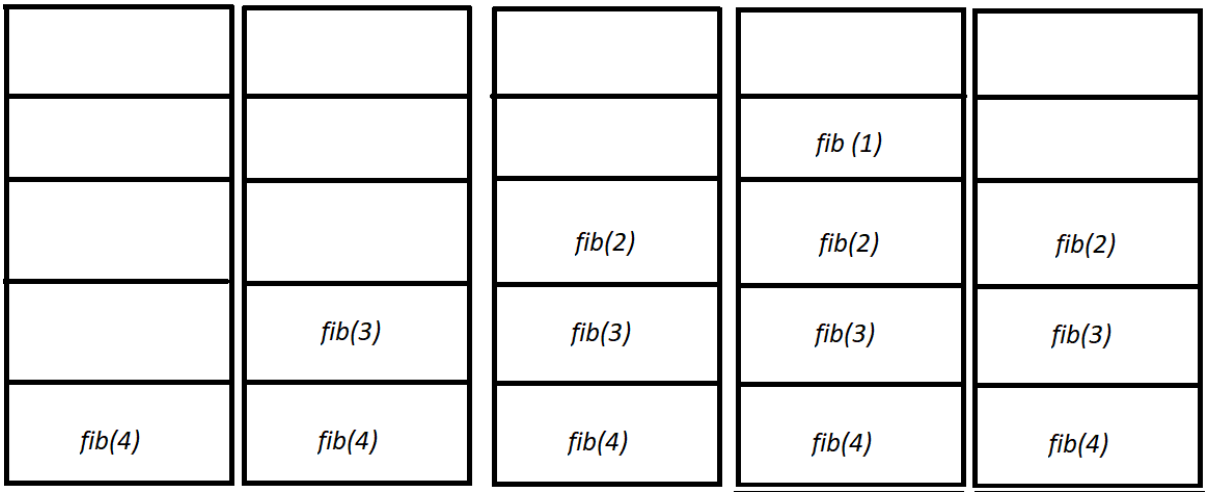
In the beginning of each question, it is described what kind of answer that you are expected to submit. If *Text and code answer* is stated, then you need to submit BOTH some argumentation/description and some code; if just (*Text answer*) or (*Code answer*) then just some argumentation/description OR code. The final answer to the answers requiring text should be **one pdf document** with one answer for each text question (or text and code question). When you hand-in, add a link to your GitHub reposetory in the beginning of your pdf file. Make sure that you have committed your code solutions to that reposetory.

Note: the **Challenge** exercises are *optional*, the others mandatory (i.e. you **have** to hand them in).

Link to repository: https://github.com/Aarhus-University-ECE/assignment-9-Teun0n

Exercises

(1) (Text) We talked about the run-time stack (see e.g. slides from lecture 7). In the lecture, we looked at the Fibonacci numbers and a program to calculate them (`fib.c`). Draw the stach as it evolves when calculating `fib(4)`



(2) (Code) Summing an array kan recursively be described as follows (a is the array, n is the length of the array):

$$sum(a,n) = \begin{cases} a[n-1] + sum(a, n-1), & \text{if } n > 0 \\ 0, & \text{if } n = 0 \end{cases}$$

Implement a recursive function with the signature `int sum(int a[], int n)` that sums the integer array a

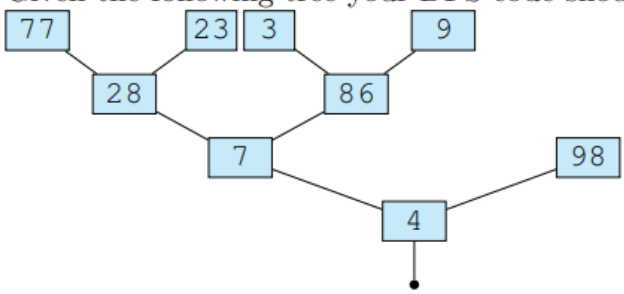
(3) (Code) In the lecture, we looked at an recursive binary search. To use binary search, the elements must be sorted. A recursive search function NOT requiring the elements to be sorted could look like (a is the array, n is the length of the array, x is the element to be found):

$$search(a,n,x) = \begin{cases} true, & \text{if } n > 0 \text{ and } a[n-1] == x \\ search(a, n-1, x), & \text{if } n > 0 \text{ and } a[n-1] != x \\ false, & \text{if } n = 0 \end{cases}$$

Implement a recursive function with the signature:
`bool search(int a[], int n, int x)` that searches the integer array a for the element x.

(4) (Code) Implement depth-first search using a stack in a fashion similar to as presented in the lectures. Your stack should be implemented as a linked list, and your tree

as *tree nodes* that each have an integer as the data item and a left and right child. Given the following tree your DFS code should print the sequence of nodes visited.



The correct output should be: 4, 7, 28, 77, 23, 86, 3, 9, 98