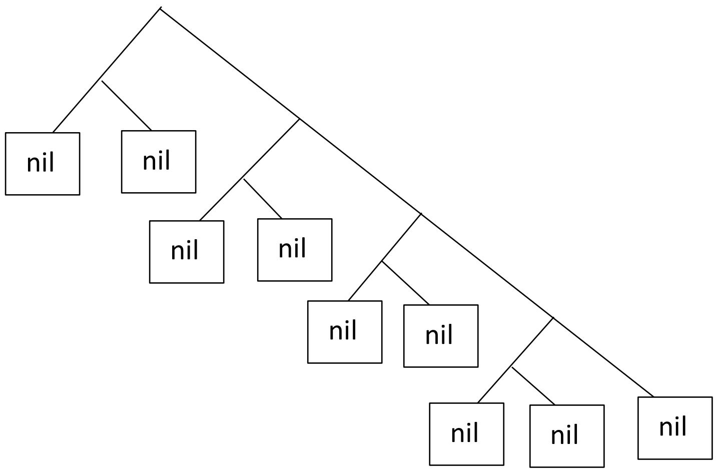
Limits of Computation

Assignment 1

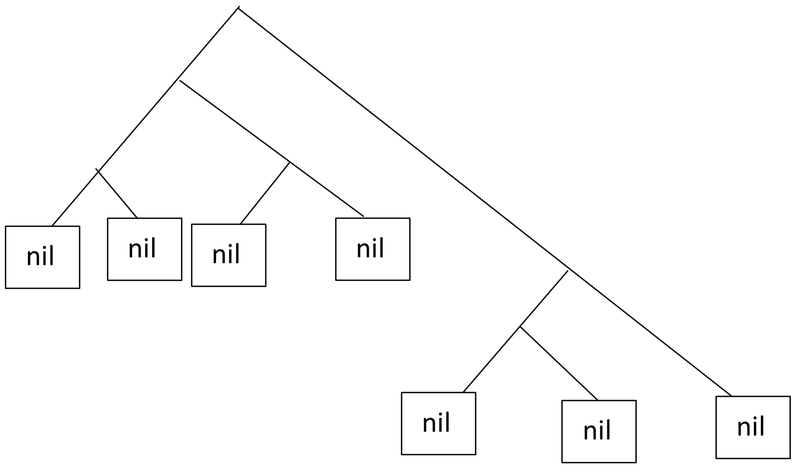
1-) a-) <<nil.nil>.<<nil.nil>.<<nil.nil>.<<nil.nil>.nil>>>>

[1,1,1,1] or [[0],[0],[0],[0]]



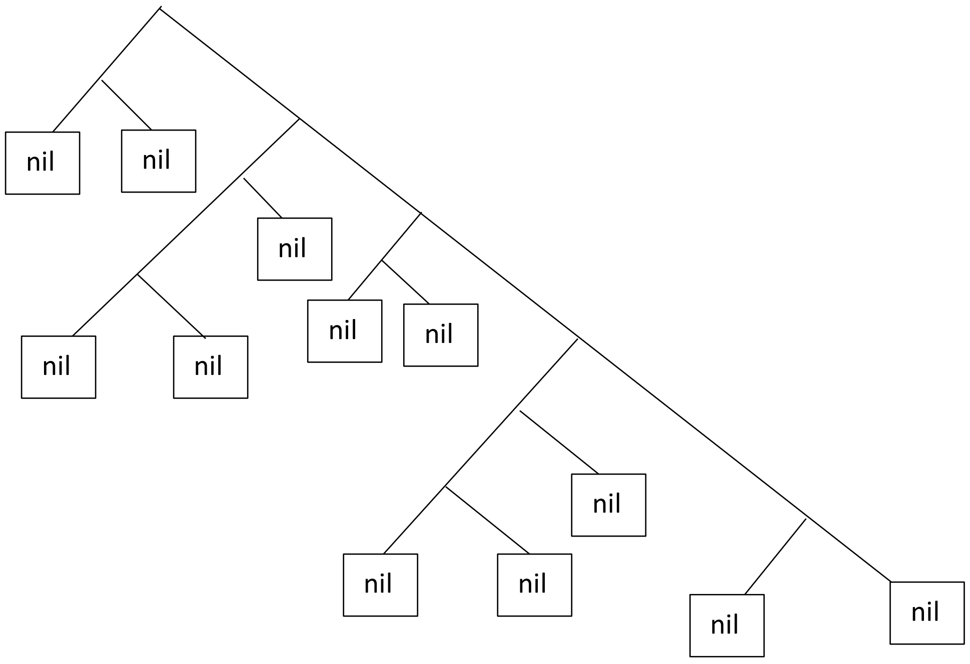
b-) <<<nil.nil>.<nil.nil>>.<<nil.nil>.nil>>

[[1,1],[0]]. It does not give a list of numbers, only a list of lists of numbers.



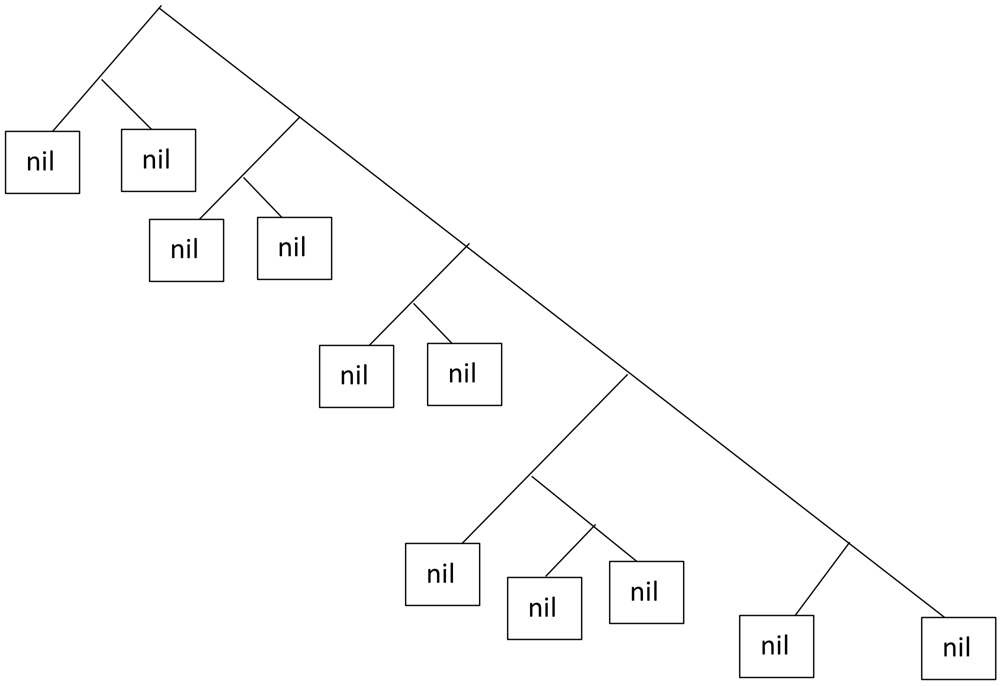
c-) <<nil.nil>.<<<nil.nil>.nil>.<<nil.nil>.<<<nil.nil>.nil>.<nil.nil>>>>>

[[0],[1],[0],[1],[]]. It does not give a list of numbers, only a list of lists of numbers.



d-) <<nil.nil>.<<nil.nil>.<<nil.nil>.<<nil.<nil.nil>>.<nil.nil>>>>>

[1,1,1,2,0] or [[0],[0],[0],[0,0],[]]



2-) As any input d ∈ D will be a list; the program implements a recursive algorithm to find the length of any given d by putting nil for each element. The output will be a list of nil's and represent a number.

3-)

```
[
 [appf, [var]] ,
 [if, [hd, [var]],
          [cons, [quote, nil], [appf, [cons, [tl, [hd, [var]]], [cons, [tl, [hd, [var]]], [quote, nil]]]]],
          [hd, [tl, [var]]] ]
]
```

4-) Yes, F-programs still inherit many properties from while-programs; Having a finite number of instructions. If carried out without problems, always produce the desired result in a finite number of steps. Procedures require no insight or ingenuity. The main difference is, F-programs are recursive, and while-programs are iterative. This requires a different approach to the same problem but still makes F-programs a good alternative choice for "effective procedures".