authors: "Tomas Ockier", "Aran Oliveras" date: 18-04-2024

Part A: Recursion

1. Indicate in the table the sequence of calls that are made until the result is obtained, the parameters and the result of each recursive call. >The first line of the table corresponds to the first recursive call to the function.

Call	V	n	Result
1	[3, 5, 7, 4, 6]	5	35746
2	[3, 5, 7, 4, 6]	4	3574
3	[3, 5, 7, 4, 6]	3	357
4	[3, 5, 7, 4, 6]	2	35
5	[3, 5, 7, 4, 6]	1	3
6	[3, 5, 7, 4, 6]	0	0

2. Implement an iterative function that does the same thing as this recursive function.

```
int FooIt (int v[], int n)
{
    int res = 0;
    for (int i=0; i<n; i++)
    {
        res = res*10 + v[i];
    }
    return res;
}
```

3. How many recursive calls are made to find the solution? How many iterations of the loop in the iterative version? Which solution seems more efficient to you? Why?

There are 6 calls until we reach to the base case. In the iterative version there are also 6 iterations of the loop. generally iterative is better in terms of velocity and memory, that is because we don't need to make a copy of the variables everytime, wich will make our stack to be full of memory.

Part B: Hashing

Exercise 1

Question A

The table `C` is the one that implements the function $h(x) = x \mod 10$, for a key x. Because we handle the collision using linear probing, an element x will be placed in the first empty cell after $h(x)$.

Question B

The method apply in order to resolve collision is chaining, where keys with the same result are appended to a list located in the corresponding position. All of them could result in the table `D`, because the order between keys that have the same result is not altered between the four different possibilities.

Exercise 2

The functions have been implemented in file `hash.c`. Similar loops to the ones in the other data structures that we learned have been used.