# **Laboratory practice No. 1: Recursion**

|  |  |
| --- | --- |
| **María Alejandra Moncada**  Universidad Eafit  Medellín, Colombia  mamoncadaa@eafit.edu.co | **Manuela Zapata Mesa**  Universidad Eafit  Medellín, Colombia  mzapatam1@eafit.edu.co |

**1) Project drill**

**1.2** We need to calculate ways to fill a rectangle with smaller rectangles of specific measures. We receive n as a parameter, the rectangle we must fill will be 2 multiplied by that n. The rectangles we must fill them will be size 2x1, our start case is n<=2, because of the number of ways available fill the big rectangle, in that case the function returns n. For the recursive case we will call the function(n) -1 and then add function (n) – 2 because we are splitting the rectangles in smaller rectangles in order to fill the big one.

**2) Codingbat exercises**

**2.1**

* **Factorial:** This function receives a number and must return that number applied to factorial function (3x2x1…) The first thing we did was to start with a case with n>0. This will perform as the recursive function. If n>0 our code will return n! - 1 because as we know when using factorial function is to multiply a number by the same number minus 1, which at the same time calls the one before it and so on recursively. The other case is that if n<0 the code returns 1.
* **Bunny ears:** This problem has the objective to count how many ears there are, receiving the number of bunnies as a parameter. As we know every bunny as two ears, so our stop case is when we don´t have any bunnies, so if the number of bunnies is 0 the code returns 0, but if bunnies number is different to 0 we use recursion to return 2 + bunny ears, which is the recursion of bunnies – 1. We use two because each bunny has two ears.
* **Fibonacci:** Fibonacci´s pattern forms by adding the two prior numbers. Our function receives a number n which is the starting point to develop Fibonacci. Our stop case is that if n<2 the code returns n. We call Fibonacci´s function – 1 and then add Fibonacci´s function – 2 for the n received previously. That is how the problem gets solved.
* **Bunny ears 2:** Essentially the problem is the same as bunny ears, but it specifies that if the number of bunnies is pair they have three ears because they have a lifted paw and that counts as another ear, but if it is not a pair number the bunnies got two ears. We receive as a parameter the number of bunnies. Our stop case is that if we have 0 bunnies the code returns 0. The other case is if the number of bunnies is not a pair number the function returns 2 + bunnies -1. And if it is a pair number of bunnies the function returns 3 + bunnies -1 because of the three ears.
* **Triangle:** The problem presents us with a triangle which has several rows, therefore if it has one row it has 1 column, if it has two rows it has 2 columns and so on. The function receives as a parameter the number of rows and it returns the number of columns depending on the number of rows. If the number of rows is < 2 it has the same number of rows as columns the number of columns is the same as the number of rows, but if the number of rows is > 2 the function returns the number of rows + triangle + number of rows – 1.

**3) Practice for final project defense presentation**

**3.5**

**- Factorial:** T(n) = n T(n-1) + c

**- Bunny ears:** T(n) = T(n-1) + c

**- Fibonacci:** T(n) = T(n-1) + T(n-2) + c

**- Bunny ears 2:** T(n) = 2T(n-1) + c

**- Triangle:** T(n) = T(n-1) + c

***4) Practice for midterms***

* 1. Línea 3: return true;

Línea 4: if (s.charAt(0)==s.chaeAt(s.length()-1)

* 1. *d*
  2. .1 Int res= solucionar(n-a,a,b,c)

***4.3.2*** res= Math.max(res, solucionar(n-b,a,b,c)+1)

* + 1. res =Math.max(res, solucionar(n-c,a,b,c)+1)

***4.5.1***

Linea 3:if(t==0)

Linea 4:return 1

Linea 8:return f1+f2+f3

***4.5.2*** B

***4.6.1*** Linea 10: return 0

***4.6.2*** Linea 12: return (n.charAt(i)-‘0’)+ (n.charAt(i+1)-‘0’)

***4.8.1*** Return 0

***4.8.2*** Int suma= ni +nj

***4.10*** B

***4.11.1*** Return lucas(n-1)+lucas(n-2)

***4.11.2*** C