Algorithmics	Student information	Date	Number of session
	UO:269546	10-03-22	3_1
	Surname: Fernández Arias		
	Name:Sara		

Escuela de Ingeniería Informática
Universidad de Oviedo

Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

Activity 1. Basic recursive models

Class Division1.java complexity

```
public class Division1 {
    public static long rec1 (int n) {
        long cont = 0;
        if (n<=0) cont++;
        else {
            for (int i=1;i<n;i++)
                cont++ ;  //O(n)
            rec1(n/3);
        }
        return cont;
    }
```

Since the number of subproblems is 1 and the size of the problem is divided by three each recursive call we have that a=1 and b=3 respectively.

Also , the complexity of the program without taking into account the recursive calls is O(n), therefore k will be 1.

Applying divide and conquer by division scheme , we have that a< a=b^k and therefore the complexity will be O(n^k). The complexity of the divide and conquer program is O(n).

Class Division2.java complexity

```
public class Division2 {
    public static long rec2 (int n) {
        long cont = 0;
        if (n<=0) cont++;
        else {
            for (int i=1;i<n;i++)
                cont++ ;  //O(n)
            rec2(n/2);
            rec2(n/2);
        }
        return cont;
    }
```

In this case the number of recursive calls is 2 , then the number of subrproblems , a is equal to 2. Each time a recursive call is produced , the problem size is divided by two. Then , b=2.

The complexity of the program without taking into account the recursive calls is O(n) , therefore k=1;

Applying the divide and conquer division scheme, provided that a=b^k since 2=2, the complexity of the program will be O(n^k *logn) then O(n logn) will be the total complexity of the program.

Algorithmics	Student information	Date	Number of session
	UO:269546	10-03-22	3_1
	Surname: Fernández Arias		
	Name:Sara		

Class Division3.java complexity

```
public class Division3 {
    public static long rec3 (int n) {
        long cont = 0;
        if (n<=0)
            cont++;
        else {
            cont++ ; // O(1)
            rec3(n/2);
            rec3(n/2);
        }
        return cont;
    }
```

In this case the number of recursive calls is 2, then a=2. Each recursive call the problem's size is divided by 2 , therefore, b=2. The complexity of the program without taking into account the recursive calls is O(1) =0(n^0).That means that k=0. Applying divide and conquer division scheme, since a> a=$b^k$the complexity of this implementation would O(n $^{log_b a}$) then it is O($n^{\log_2 2}$) then O(n^1) which is equal to O(n).

Class Division4.java complexity

I provided the following implementation to class Division4.java:

```
public static long rec4 (int n) {
    long cont = 0;
    if (n<=0) cont++;
    else {
        for (int i=1;i<n;i++)
            for (int j=1;j<n;j++)
                cont++ ;  //O(n^2)  -> k=exp of the complexity of the overall scheme excluding recursive calls,
        rec4(n/3);//subproblem 1
        rec4(n/3);//subproblem 2
        rec4(n/3);//subproblem 3
        rec4(n/3);//subproblem 4
    }
    return cont;
}
```

Since it was asked for the program to have a number of subproblems a equal to 4, and a complexity of O(n^2) , by using divide and conquer by division.

In order to reach that complexity , a<b^kmust happen. Therefore either b>a or k!=1.Since I divide the size of the problem by 3 each time, making that b=3. Since b<a, I've chosen to make the complexity of the program (without taking into account the recursive calls) O(n^2) by adding two nested for loops. Then a<b^k since 4<3^2=9 . Ando so , making the complexity O(n^k), O(n^2).

Algorithmics	Student information	Date	Number of session
	UO:269546	10-03-22	3_1
	Surname: Fernández Arias		
	Name:Sara		

Class Substraction1.java complexity

```
public static long rec1(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        cont++;  // O(1)=O(n^0)
     rec1(n-1);
    }
    return cont;
}
```

This implementation uses a divide and conquer strategy by subtraction having a number of subproblems, ,a , equal to 1.

A unit is subtracted from the problem's size each recursive call, then b=1.

The complexity of the program without taking into account the recursive calls is $O(n^0)$=O(1) Therefore , k=0.

Applying the division schema, since a=1, the complexity of the program will be $O(n^{k+1})$ so, $O(n^{0+1})$ = $O(n^1)$

Class Substraction2.java complexity

```
public class Subtraction2 {
    public static long rec2(int n) {
        long cont = 0;
        if (n<=0)
            cont++;
        else {
            for (int i=0;i<n;i++)
                cont++; // O(n)
            rec2(n-1);
        }
        return cont;
    }
```

The number of subproblems is 1, since there's only one recursive call.

Each time the complexity of the problem is decreased in one unit , so b=1.

The complexity of the program without taking into account the recursive call is O(n). Therefore k=1.

Applying the division schema, since a=1, the complexity of the program will be $O(n^{1+1})$ so, $O(n^2)$.

Algorithmics	Student information	Date	Number of session
	UO:269546	10-03-22	3_1
	Surname: Fernández Arias		
	Name:Sara		

Class Substraction3.java complexity

```
public class Subtraction3{
    public static long rec3(int n) {
        long cont = 0;
        if (n<=0)
            cont++;
        else {
            cont++;  //O(1)
            rec3(n-1);
            rec3(n-1);
        }
        return cont;
    }
```

Two recursive calls are made, so a=2. Each time the complexity of the problem is decreased in one unit, so b=1. The complexity without taking the recursive calls into account is $O(n^0)$. That makes k=0.

Applying the division schema, since a>1, the complexity of the program will be $O(a^{n/b})$ so, $O(2^{n/1}) = O(2^n)$.

Class Substraction4.java complexity

For the class Substraction4.java, I provided the following implementation:

```
public static long rec4(int n) {
    long cont = 0;
    if (n<=0)
        cont++;
    else {
        for (int i=0;i<n;i++)
            cont++; // O(n)
        rec4(n-2);//subproblem 1
        rec4(n-2);//subproblem 2
        rec4(n-2); //subproblem 3.
    }
    return cont;
}
```

Since it was asked for the program to have a complexity of $O(3^{\frac{n}{2}})$ by using divide and conquer by subtraction , we need a > 1 , a=3. Also we need b to be equal to two. The value of k is not decisive this time.

For that I made three recursive calls in order to make three subproblems and so a=3. Then in each call I made it subtract two to the size of the problem , so that b=2.