UCT. Ingeniería Civil Informática.
Programación II (Estructuras de Datos.)

Guía de Ejercicios de Recursividad.

Para enfrentar esta guía de ejercicios debemos tener en cuenta para plantear la solución a cada problema la siguiente metodología general:

1- Definir caso(s) base.
2- Hacer llamadas recursivas con instancias del problema con menor tamaño.

A continuación se proponen algunos ejercicios para los que podrán encontrar una solución recursiva más adelante en este documento.

Ejercicios Propuestos:

1- Cree una función que multiplique dos enteros mediante sumas recursivas.
2- Cree una función que eleve un número a un potencia entera.
3- Cree una función que implemente el algoritmo de Euclides para el cálculo del máximo común divisor entre dos números**.**
4- Cree una función que dado un vector diga si es o no palíndromo.
5- Cree una función que dado un número en decimal lo convierta en un *String* con su representación en binario.
6- Cree una función que dado un String con la representación en binario de un número lo convierta en decimal.
7- Cree una función que permita ordenar una vector utilizando el algoritmo de ordenamiento por selección.
8- Cree una función que imprima en pantalla la secuencia de solución del problema de Las Torres de Hanoi a partir de la cantidad de discos.

Ejemplo de solución a los ejercicios:

1- Cree una función que multiplique dos enteros mediante sumas recursivas.

```
public static int productoSumaRecursiva(int a, int b){
    if (b == 1) {
        return a;
    } else {
        return a + productoSumaRecursiva(a , b-1);
    }
}
```

2- Cree una función que eleve un número a un potencia entera.

```
public static int potenciaEnteraRecursiva(int k, int n){
    if (n == 0) {
```

```
        return 1;
    } else {
        return k * potenciaEnteraRecursiva(k , n-1);
    }
}
```

3- Cree una función que implemente el algoritmo de Euclides para el cálculo del máximo común divisor entre dos números**.**

```
public static int mcd(int a, int b){
    if(a > b){
        if ((a % b) == 0){
            return b;
        } else {
            return mcd(b, (a % b));
        }
    } else {
        if ((b % a) == 0){
            return a;
        } else {
            return mcd(a, (b % a));
        }
    }
}
```

4- Cree una función que dado un vector diga si es o no palíndromo.

```
public static boolean palindromo(int[] A, int desde, int hasta){
    if ( desde >= hasta ) {
        return true;
    } else {
        if (A[desde] == A[hasta]) {
                return palindromo( A, desde +1, hasta -1);
        } else {
            return false;
        }
    }
}
```

5- Cree una función que dado un número en decimal lo convierta en un *String* con su representación en binario.

```
public static String intToBin(int n){
    if (n == 0) {
        return "0";
    } else if( n == 1 ) {
        return "1";
    } else {
        return intToBin(n/2) + n % 2;
    }
}
```

6- Cree una función que dado un *String* con la representación en binario de un número lo convierta en decimal.

```
public static int binToDec(String s){
    if (s.equals("0")) {
        return 0;
    } else if(s.equals("1")) {
        return 1;
    } else {
        int primero = Integer.parseInt(s.substring(0, 1));
        return (primero * (int)Math.pow(2, s.length()-1)) +
         binToDec(s.substring(1));
    }
}
```

7- Cree una función que permita ordenar una vector utilizando el algoritmo de ordenamiento por selección.

```
public static int[] ordenaSeleccionR(int[] A, int desde, int hasta){
    if (desde == hasta) {
        return A;
    } else {
        int posicionMinimo = indiceMinimo(A, desde, hasta);
        int valorMinimo = A[posicionMinimo];
        A[posicionMinimo] = A[desde];
        A[desde] = valorMinimo;
        return ordenaSeleccionR(A, desde + 1, hasta);
    }
}
```

8- Cree una función que imprima en pantalla la secuencia de solución del problema de Las Torres de Hanoi a partir de la cantidad de discos.

```
public static void hanoiTowers(int n, String origen, String auxiliar,
String destino){
    if (n == 1) {
        System.out.println("Mover disco de " + origen + " a " +
destino);
    } else {
        hanoiTowers(n-1, origen, destino , auxiliar);
        hanoiTowers(1, origen, auxiliar, destino);
        hanoiTowers(n-1, auxiliar, origen, destino);
    }

}
```