Recursividad: Ejercicios Adicionales

1) Desarrollar una función recursiva que devuelva la cantidad de dígitos de un número entero, sin utilizar cadenas de caracteres:
   a) Modificar el algoritmo visto en clase para que logre contar dígitos en forma recursiva tanto para valores positivos y negativos.
   b) Resolver con cadenas de caracteres.

2) Desarrollar una función recursiva que reciba un número binario y lo devuelva convertido a base decimal. Creando una excepción o generando **ValueError** en caso de recibir un número que no es binario o un número negativo con el mensaje aclaratorio correspondiente a cada error. Resolver de dos formas distintas:
   a) Resolver utilizando **raise**
   b) Resolver utilizando **assert**

3) Crear un programa principal que capture las excepciones generadas por la función y vuelva a solicitar un nuevo número en caso de capture la excepción.

4) Desarrollar una función recursiva para resolver exponente con multiplicaciones sucesivas.

5) Desarrollar una función recursiva para retornar el mínimo de una lista. Resolver:
   a) Utilizando rebanadas.
   b) Utilizando índices.

6) Contar la cantidad de números de una cadena mediante una función recursiva.

7) **Búsqueda Binaria:** desarrollar un algoritmo para realizar un algoritmo de búsqueda binaria en forma recursiva.

Ayuda para resolver el ejercicio:

La solución recursiva se fundamenta en lo siguiente (aplicando el enfoque recursivo obviamente): como clave para la solución recursiva de la búsqueda cabe citar que, en cada paso, se considera el elemento central de la lista (con lo que éste se DIVIDE en dos mitades) y dependiendo de la comparación de x con lista[cen] se continúa el mismo proceso por la mitad correspondiente.

Para poder llevar a cabo esta reducción del problema, es necesario que la representación de datos utilizada admita la misma. Por lo tanto, en este caso será necesario tratar sublistas en vez de una lista, por lo que se considerará como dato de entrada a la función recursiva, una sublista dada por la terna <lista, ini, fin>.

En definitiva, la **solución recursiva** en pseudocódigo al problema puede expresarse como:

```
cen <- (ini +fin) DIV 2
Si (x < lista[cen])
       Entonces      Retorna BusBinRec(v, ini, cen, x)
       Sino          Retorna BusBinRec(v, cen, fin, x)
FinSi
```

Para el **caso base** se debe reducir la sublista hasta que la última lista considerada siempre tenga dos componentes consecutivas. Así pues, el **caso base** se alcanza cuando las posiciones inicial y final de la sublista son consecutivas (ini+1 = fin), en cuyo caso, el resultado de la búsqueda viene dado por el valor de la posición ini. Nótese, que en el caso

extremo de que la lista esté vacío, se cumple inmediatamente la condición del caso base, gracias al hecho de considerar las dos posiciones ficticias lista[0] y lista[n+1].
Por lo tanto, la definición de la función completa, podría ser la siguiente:

Función: BusBinRec

Pre-condición: una sublista de números (enteros), representado por la tripleta de datos <lista, ini, fin>. Como precondición se considera que la lista ya tiene cargados un conjunto de valores válido, que estos valores están ordenados, y que el número de elementos de la lista es cero si fin < ini, y el valor de la expresión fin – ini + 1, en caso contrario. Además, se tiene que x es el elemento a buscar dentro de la sublista.

Pos-condición: la posición i en la que debería encontrarse el elemento x dentro de la sublista, de forma que se cumple que lista[i] <= x < lista[i+1]