1) Implementa en Haskell una función que pida un nombre por pantalla y lo devuelve en mayúsculas y en minúsculas

```
*MasEntradaSalida> mayusculasYminusculas
Dame tu nombre
Alfonso
En mayusculas: ALFONSO
En minusculas: alfonso
```

2) Implementa una función en Haskell que pida dos números enteros y si quieres sumarlos o restarlos y que te devuelva el resultado. Este proceso se repite mientras que la operación introducida no sea la suma o la resta.

```
*MasEntradaSalida> calculadora
Dame el primer operando
7
Dame el segundo operador
8
Quieres sumar ('s') o restar('r')?
d
No tengo esa operacion
Dame el primer operando
7
Dame el segundo operador
8
Quieres sumar ('s') o restar('r')?
s
15
```

3) Implementa una función en Haskell que lea de un archivo de texto su contenido y lo reescribe en otro separando en dos líneas: una para las letras y otra para el resto de caracteres

Archivo de entrada:

```
A mi lo que mas me gusta es tomarme una cervecita al sol con mi amigo @pagoyo1 hablando de #futbol y del disco de AC&DC
```

Archivo de salida:

```
AmiloquemasmegustaestomarmeunacervecitaalsolconmiamigopagoyohablandodefutbolydeldiscodeACDC
@1#&
```

4) Implementa en Haskell una función que te vaya pidiendo líneas por pantalla y te vaya mostrando su longitud hasta que introduzcas una de longitud 0 (vacía)

```
*MasEntradaSalida> lineas
Dame una linea
Me encanta programar con Haskell
32
Dame una linea
Aunque, si te digo la verdad, lo que mas me pone es resolver un ejercicio con diodos Zener
90
Dame una linea
Viva la gente de Software
25
Dame una linea

Adios
```

5) Implementa un programa en Haskell que pida una frase y nos la devuelva con las palabras al revés, aunque sin cambiarlas de sitio. Este proceso se repetirá hasta que se introduzca la frase nula. Se recomienda usar las funciones `words` para separar una frase en palabras y `unwords` para hacer la operación inversa.

```
*MasEntradaSalida> darLaVuelta
Dame una linea
Voy a aprobar PP seguro
yoV a raborpa PP oruges
Dame una linea
Y luego Concurrente
Y ogeul etnerrucnoC
Dame una linea

Adios
```

6) Diseña una función en Haskell que sea llamada con un número por pantalla y luego lo intente adivinar (va diciendo mientras tanto si es mayor o menor)

```
*MasEntradaSalida> adivinar 10
dame un numero
7
dame uno mayor
dame un numero
9
dame uno mayor
dame un numero
12
Dame uno menor
dame un numero
10
Correcto
```

7) Diseña la función anterior pero haciendo uso de la operacion binding (>>=)

8) Implementa en Haskell una función que lea de dos archivos diferentes la secuencia de números enteros (los cuales pueden tener más de un dígito o ser negativos) que contiene cada uno y muestre por pantalla la suma emparejada de cada de ellos, indicando el número de operación por la que va. Las sumas pararán cuando se acabe una de las listas (o las dos a la vez). La instrucción que convierte un `String` en un tipo de dato `T` es `(read x)::T`.

```
read "7"::Int
7
```

```
"1 5 6 8 4 2 (-6) 8 4 1 2 0 3"  (Contenido del archivo1)
"1 1 (-9) (-5) 6 2"             (Contenido del archivo2)
```

```
"Suma 1: 2"
"Suma 2: 6"
"Suma 3: -3"
"Suma 4: 3"
"Suma 5: 10"
"Suma 6: 4"
```

9) Escribe un programa en Haskell que lea un archivo de texto, separe el contenido del mismo por frases (separadas por puntos) y reescriba su contenido en dos archivos diferentes, pero separando de manera alternativa cada una de las frases del texto original en cada uno de los archivos. Para realizar este ejercicio está prohibido hacer uso de la función `split` ni de ninguna de sus variantes