Ratones Paranoicos

Ahora que los ratones inversores (¿se acuerdan del parcial de objetos?) han consolidado su fortuna, han empezado a preocuparse por resguardar con contraseñas la información de sus cuentas, por lo que nos han pedido que implementemos un sistema para evaluar su nivel de seguridad.

Sabiendo que se tienen las siguientes funciones:

```
isAlfa x = elem x ['a'..'Z']
isUpper x = elem x ['A'..'Z']
isDigit x = elem x ['0'..'9']

-- devuelve los segmentos finales de una lista
-- Ej: tails "abc" == ["abc", "bc", "c",""]
tails :: [a] -> [[a]]
```

```
--convierte un carácter a su código ASCII
fromEnum :: Char -> Int

--convierte un código ASCII en carácter
toEnum :: Int -> Char
```

y un diccionario común y corriente:

```
diccionario = ["aaron", "abaco", "abecedario", "baliente", "beligerante"] -- etc.
```

Se pide resolver los siguientes problemas, sin usar recursividad salvo indicación en contrario:

A. Un requisito es una función que me dice si un password cumple una cierta restricción. Implementar los siguientes requisitos:

1. Empezar con cierta letra
   ```
   > empiezaConLetra 'p' "papafrita"
   True
   ```
2. Tener al menos un número
   ```
   > tieneAlMenosUnNumero "papafrita"
   False
   ```
3. Tener exactamente cierta cantidad de mayúsculas
   ```
   > tieneXMayusculas 2 "PaPaFrita"
   False
   ```
4. Ser indeducible (cuando no contiene ninguna palabra del diccionario). No hace falta chequear mayúsculas.
   ```
   Main> esIndeducible "papafritabaliente"
   False
   ```

   Resolverlo empleando las siguientes funciones, que también hay que desarrollar:
   `empiezaCon`, que me dice si un string empieza con otro.
   ```
   > empiezaCon "papafrita" "papa"
   True
   ```

   `contieneA`, que me dice si un string contiene a otro, al principio ó en cualquier lado. Acá se puede emplear recursividad (no es obligatorio)
   ```
   > contieneA "miMamaMeMima" "Mama"
   True
   ```

B. Existen aplicaciones en las que los ratones usuarios pueden registrarse.

Cuando un usuario se registra en ella, se valida que la password que eligió cumpla con todos los requisitos de seguridad que la aplicación disponga. Luego, para guardar la password, la encripta con su propio método de encriptado.

Una aplicación se representa con una terna `(lista usuarios, requisitos de password, método de encriptado)`

Los usuarios, en tanto, se representan como un par `(nombre, passwordEncriptado)`

Por ejemplo:

```
bancoNacion = ([("juan","sdsdiulwd"), ("robby","sxwrhotxhlhh"), ...], [tieneAlMenosUnNumero, esIndeducible], textoHash)
```

1. Saber si puedo usar un password para cierta aplicación. Esto ocurre cuando cumple todos los requisitos

   ```
   Main> puedoUsar "qwertyuiop1" bancoNacion
   True -- es indeducible y tiene un número.
   Main> puedoUsar "12baliente" bancoNacion
   False --tiene una palabra del diccionario, no es indeducible
   ```

2. Encriptamiento. Encriptar es convertir un string en **otro string,** ilegible, para que nadie pueda saber cuál es el password original.
   a. Método césar N: Convierte cada letra en la número N siguiente, según el código ASCII

      ```
      > cesar 3 "CaBe0"
      "FdEh3" -- por ejemplo, F en el codigo ASCII esta tres después que C
      ```

   b. Método textoHash: convierte un string en la **representación textual** de su hash. El hash de un string se calcula como la sumatoria de los códigos ASCII de sus dígitos. Ejemplo:

      ```
      > textoHash "hola"
      "420" -- los ASCIIs de "hola" son 104,111,108 y 97, que suman 420
      ```

3. Hacer una función `registrarse`, que recibe un nombre de usuario, una password, y una aplicación, y agrega el usuario a la lista de usuarios de la aplicación, si es que puede usar la password. Asumir que el usuario es nuevo y no se encuentra en la la lista de usuarios. ¡Recordar encriptar la password!
4. Completar las siguientes definiciones **sin escribir funciones auxiliares**:
   a. `paradigma = . . .`
      Paradigma! es una aplicación que aún no tiene usuarios, usa encriptación `cesar` con `N = 4`, y cuyos requisitos son que los passwords tengan más de 6 caracteres y no empiecen con ‘t’.
   b. `facebutt = . . .`
      Facebutt es una aplicación que acepta cualquier password, no encripta las passwords y tiene infinitos usuarios, de la forma
      `("usuario1", textoHash "laPasswordDificil1") , ("usuario2", textoHash "laPasswordDificil2")`, etc.
      ¿Puedo usar `puedoUsar` con esta aplicación? ¿Y `registrarse`?