UNIVERSITAT JAUME I

1.º Ingeniería Informática
1.º Matemática Computacional
EI1003/MT1003 Programación I
Práctica 4

Ejercicio 1

Escribe un programa que lea repetidamente líneas de texto, las vaya almacenando en una lista hasta la primera cadena vacía leída (que no ha de almacenar) y, finalmente, muestre en ese mismo orden todas las líneas no vacías leídas, junto con sus longitudes. El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo cadenas de caracteres, o vacío para acabar...
Nueva cadena: hola, buenos días
Nueva cadena: 1024
Nueva cadena: 4*2=8
Nueva cadena: Mala SUERTE
Nueva cadena:
Cadenas leídas:
  Cadena de longitud 17: =>hola, buenos días<=
  Cadena de longitud 4: =>1024<=
  Cadena de longitud 5: =>4*2=8<=
  Cadena de longitud 11: =>Mala SUERTE<=
```

Ejercicio 2

Modifica una copia del programa anterior para que el bucle que va mostrando las cadenas almacenadas en la lista la recorra en orden inverso. Un ejemplo de ejecución del programa es:

```
Ve introduciendo cadenas de caracteres, o vacío para acabar...
Nueva cadena: hola, buenos días
Nueva cadena: 1024
Nueva cadena: 4*2=8
Nueva cadena: Mala SUERTE
Nueva cadena:
Cadenas leídas (desde la última hasta la primera):
  Cadena de longitud 11: =>Mala SUERTE<=
  Cadena de longitud 5: =>4*2=8<=
  Cadena de longitud 4: =>1024<=
  Cadena de longitud 17: =>hola, buenos días<=
```

Ejercicio 3

Escribe un programa que lea repetidamente números enteros hasta el primer negativo leído (que debe descartar). Los no negativos que vaya leyendo debe distribuirlos entre dos listas distintas: una para los números pares y otra para los impares. Cada lista ha de respetar el orden de introducción de los números y ambas deben ser mostradas tras finalizar el bucle de lectura. Un ejemplo de ejecución del programa es:

```
Ve introduciendo números enteros positivos, o un número negativo para acabar...
Nuevo número: 25
Nuevo número: 33
Nuevo número: 202
Nuevo número: 7
Nuevo número: 14
Nuevo número: 0
Nuevo número: 1
Nuevo número: -9
Números pares: [202, 14, 0]
Números impares: [25, 33, 7, 1]
```

Ejercicio 4 (Obligatorio)

Escribe una función sin parámetros, `leer_lista_enteros`, que lea repetidamente números enteros y los vaya almacenando en una lista hasta recibir, en vez de un número entero, una cadena vacía. La función debe devolver como resultado la lista de enteros formada.

Utiliza la función anterior para escribir un programa que lea una lista de enteros y después cree una nueva lista en la que cada elemento sea el cuadrado del número que ocupe esa misma posición en la lista leída. Finalmente, ambas listas deben ser mostradas como ilustra el siguiente ejemplo:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 11
Nuevo número: -9
Nuevo número: -25
Nuevo número: 101
Nuevo número: 0
Nuevo número: 2
Nuevo número:
Cuadrados de los números leídos: [121, 81, 625, 10201, 0, 4]
Números leídos: [11, -9, -25, 101, 0, 2]
```

Ejercicio 5 (Obligatorio)

Escribe un programa que, tras llevar a cabo mediante `leer_lista_enteros` una lectura de enteros análoga a la del anterior, vaya pidiendo al usuario posiciones que eliminar de la lista hasta que quede vacía. Cada posición debe expresarla el usuario como un entero no negativo inferior a la longitud que la lista tenga en ese momento. Si el entero leído no cumple las condiciones exigidas, el programa debe mostrar el mensaje "Posición incorrecta".

Para borrar elementos de la lista, escribe y utiliza una función, `borrar_elemento`, que reciba como parámetros una lista y la posición que se quiera borrar. Si la posición es no negativa e inferior a la longitud de la lista, la función borrará el elemento y devolverá `True`; en caso contrario, dejará la lista intacta y devolverá `False`.

El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 4
Nuevo número: -5
Nuevo número: 2
Nuevo número: -6
Nuevo número: -6
Nuevo número: 1
Nuevo número: 2
Nuevo número: 2
Nuevo número:
¿Qué posición debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 7
¿Qué posición debo eliminar de [4, -5, 2, -6, -6, 1, 2]? 1
¿Qué posición debo eliminar de [4, 2, -6, -6, 1, 2]? 6
Posición incorrecta
¿Qué posición debo eliminar de [4, 2, -6, -6, 1, 2]? 0
¿Qué posición debo eliminar de [2, -6, -6, 1, 2]? 0
¿Qué posición debo eliminar de [-6, -6, 1, 2]? -1
Posición incorrecta
¿Qué posición debo eliminar de [-6, -6, 1, 2]? 3
¿Qué posición debo eliminar de [-6, -6, 1]? 1
¿Qué posición debo eliminar de [-6, 1]? 0
¿Qué posición debo eliminar de [1]? 0
La lista ha quedado vacía
```

Ejercicio 6

Modifica una copia del programa anterior para que, en el bucle de eliminación de elementos, lo que especifique el usuario no sea una posición de la lista, sino el valor de uno de sus elementos. Si hay varios elementos iguales, se eliminará el primero de ellos. Y el mensaje para peticiones del usuario irrealizables por el programa será ahora "Número no encontrado".

En tu nueva versión, la función `borrar_elemento` buscará en la lista recibida como primer parámetro la posición ocupada por el valor recibido como segundo y, si la encuentra, procederá a borrarla y devolverá `True`; en caso contrario, devolverá `False`.

El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 4
Nuevo número: -5
Nuevo número: 2
Nuevo número: -6
Nuevo número: -6
Nuevo número: 1
Nuevo número: 2
Nuevo número: 2
Nuevo número:
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 7
Número no encontrado
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 1
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 2, 2]? 2
¿Qué número debo eliminar de [4, -5, -6, -6, 2, 2]? 2
¿Qué número debo eliminar de [4, -5, -6, -6, 2]? 2
¿Qué número debo eliminar de [4, -5, -6, -6]? -6
¿Qué número debo eliminar de [4, -5, -6]? -5
¿Qué número debo eliminar de [4, -6]? 4
¿Qué número debo eliminar de [-6]? 2
Número no encontrado
¿Qué número debo eliminar de [-6]? -6
La lista ha quedado vacía
```

Ejercicio 7

Modifica una copia del programa anterior para que, si el usuario pide la eliminación de un valor que comparten varios elementos de la lista, el realmente eliminado por el programa sea el último de ellos. Debes cambiar únicamente el cuerpo de la función `borrar_elemento`.

Un ejemplo de ejecución del programa es:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 4
Nuevo número: -5
Nuevo número: 2
Nuevo número: -6
Nuevo número: -6
Nuevo número: 1
Nuevo número: 2
Nuevo número: 2
Nuevo número:
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 7
Número no encontrado
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 1
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 2, 2]? 2
```

```
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 2]? 2
¿Qué número debo eliminar de [4, -5, 2, -6, -6]? 2
¿Qué número debo eliminar de [4, -5, -6, -6]? -6
¿Qué número debo eliminar de [4, -5, -6]? -5
¿Qué número debo eliminar de [4, -6]? 4
¿Qué número debo eliminar de [-6]? 2
Número no encontrado
¿Qué número debo eliminar de [-6]? -6
La lista ha quedado vacía
```

Ejercicio 8

Modifica una copia del programa anterior para que, si el usuario pide la eliminación de un valor que comparten varios elementos de la lista, todos ellos sean eliminados. Como en el ejercicio anterior, solamente debes cambiar el cuerpo de la función `borrar_elemento`.

Un ejemplo de ejecución del programa es:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 4
Nuevo número: -5
Nuevo número: 2
Nuevo número: -6
Nuevo número: -6
Nuevo número: 1
Nuevo número: 2
Nuevo número: 2
Nuevo número:
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 7
Número no encontrado
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 1, 2, 2]? 1
¿Qué número debo eliminar de [4, -5, 2, -6, -6, 2, 2]? 2
¿Qué número debo eliminar de [4, -5, -6, -6]? 2
Número no encontrado
¿Qué número debo eliminar de [4, -5, -6, -6]? -6
¿Qué número debo eliminar de [4, -5]? -5
¿Qué número debo eliminar de [4]? 4
La lista ha quedado vacía
```

Ejercicio 9

Escribe un programa que, tras llevar a cabo mediante `leer_lista_enteros` una lectura de enteros análoga a la de los últimos programas, vaya pidiendo al usuario nuevos valores enteros que interpretar como posibles resultados de sumar elementos de la lista original (o una cadena vacía para acabar la ejecución del programa). Concretamente, por cada valor introducido, el programa debe localizar todos los pares de elementos consecutivos cuya suma coincida con el resultado buscado. El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 2
Nuevo número: 8
Nuevo número: 2
Nuevo número: -10
Nuevo número: -15
Nuevo número: 5
Nuevo número: 5
Nuevo número: 0
Nuevo número:
```

```
Lista leída: [2, 8, 2, -10, -15, 5, 5, 0]
Ve contestando con números enteros, o con una cadena vacía para acabar...
¿Qué suma he de buscar en dos posiciones consecutivas? 8
¿Qué suma he de buscar en dos posiciones consecutivas? -8
  2 + -10 = -8
¿Qué suma he de buscar en dos posiciones consecutivas? 10
  2 + 8 = 10
  8 + 2 = 10
  5 + 5 = 10
¿Qué suma he de buscar en dos posiciones consecutivas?
¡Adiós!
```

Ejercicio 10 (Obligatorio)

Escribe un programa que, tras llevar a cabo mediante `leer_lista_enteros` una lectura de enteros análoga a la de los últimos programas, modifique la lista leída de la siguiente forma:

- Primero, determinará cuál es el menor elemento de la lista. En el caso de que la lista tenga más de un elemento igual al menor, se eligirá su primera aparición.
- Después, si ese menor elemento no coincide con el primero de la lista, intercambiará las posiciones de ambos (es decir, el primer elemento de la lista pasará a ocupar la posición que ocupaba el menor, mientras que ese menor pasará a ser el primero de la lista).

Para ello, define:

- Una función, `posición_menor`, que reciba una lista y devuelva la posición del menor de sus elementos. Puedes asumir (en esta función, no en el programa) que la lista no está vacía.
- Un procedimiento, `intercambiar`, que reciba una lista y dos posiciones, `i` y `j`, e intercambie los elementos en las posiciones `i` y `j`. Si `i` es igual a `j`, la lista no debe variar. Puedes asumir que tanto `i` como `j` son posiciones válidas.

El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 5
Nuevo número: 0
Nuevo número: -4
Nuevo número: 3
Nuevo número: -7
Nuevo número: 1
Nuevo número: -7
Nuevo número:
Lista leída: [5, 0, -4, 3, -7, 1, -7]
Modificada:  [-7, 0, -4, 3, 5, 1, -7]
```

Por otra parte, si la lista leída es la vacía, el programa no ha de hacer prácticamente nada:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número:
Lista leída: []
Modificada:  []
```

Ejercicio 11 (Obligatorio)

Escribe un procedimiento, `ordenar_lista`, que reciba como parámetro una lista de enteros y, haciendo uso de la función `posición_menor` y el procedimiento `intercambiar`, la modifique para que sus elementos queden ordenados de menor a mayor. Para ello, puedes utilizar el siguiente método de ordenación:

- En un primer paso, intercambia el primer elemento de la lista con el menor de ellos, como se hace en el ejercicio anterior.
- En el segundo paso, habrá que intercambiar el segundo elemento de la lista con el menor de los elementos a partir del segundo.
- En el tercer paso, habrá que intercambiar el tercer elemento de la lista con el menor de los elementos a partir del tercero.
- Y así sucesivamente.

De este modo, la lista acabará quedando ordenada.

Deberás modificar la función `posición_menor` para que tenga un parámetro adicional que indique la posición desde la que debe encontrar el menor elemento (puedes asumir que esa posición es válida). El procedimiento `intercambiar` no varía.

Escribe un programa que, tras llevar a cabo mediante `leer_lista_enteros` una lectura de enteros análoga a la de los últimos programas, utilice `ordenar_lista` para ordenar la lista leída y mostrarla por pantalla como muestra el siguiente ejemplo de ejecución:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 5
Nuevo número: 0
Nuevo número: -4
Nuevo número: 3
Nuevo número: -7
Nuevo número: 1
Nuevo número: -7
Nuevo número:
Lista leída: [5, 0, -4, 3, -7, 1, -7]
Lista ordenada: [-7, -7, -4, 0, 1, 3, 5]
```

Tu programa debe funcionar también para listas con un solo elemento como:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 5
Nuevo número:
Lista leída: [5]
Lista ordenada: [5]
```

E incluso para listas vacías:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número:
Lista leída: []
Lista ordenada: []
```

Ejercicio 12 (Obligatorio)

Escribe un programa que, primero, lleve a cabo mediante `leer_lista_enteros` una lectura de enteros análoga a la de los últimos programas y, después, cree una nueva lista que esté formada por los números que aparecen más de una vez en la de enteros leídos. En la nueva lista, cada número que deba aparecer en ella lo hará una sola vez y, además, el orden respetará el de primeras apariciones en la lista original. También ha de calcularse la suma de los elementos de la nueva lista.

Para hacerlo, define y utiliza dos nuevas funciones: la función `repetidos_lista` debe recibir una lista y devolver otra lista con los elementos que aparezcan más de una vez en la recibida y la función `suma_lista` debe devolver la suma de los elementos de la lista que reciba como parámetro. Un ejemplo de ejecución del programa es:

```
Ve introduciendo números enteros, o una cadena vacía para acabar...
Nuevo número: 4
Nuevo número: 15
Nuevo número: 0
Nuevo número: 0
Nuevo número: -3
Nuevo número: 15
Nuevo número: 77
Nuevo número: -3
Nuevo número: 15
Nuevo número:
Números leídos más de una vez (suman 12): [15, 0, -3]
Todos los números leídos: [4, 15, 0, 0, -3, 15, 77, -3, 15]
```

Ejercicio 13 (Obligatorio)

Escribe un programa que sirva para calcular, en unas elecciones, el porcentaje que cada candidatura ha obtenido sobre el total de votos a candidaturas. El programa debe proceder del siguiente modo:

- Primero, debe leer los nombres de todas las candidaturas.
- Después, por cada nombre de candidatura anteriormente leído, debe solicitar al usuario el número de votos obtenido por dicha candidatura.
- Finalmente, ha de calcular y mostrar, con dos decimales, los correspondientes porcentajes.

El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo candidaturas, o vacío para acabar...
Nueva candidatura: PA
Nueva candidatura: PB
Nueva candidatura: PC
Nueva candidatura: PD
Nueva candidatura:
Votos para PA: 14222
Votos para PB: 9012
Votos para PC: 27530
Votos para PD: 1569
   27.18% de los votos a candidaturas para PA
   17.22% de los votos a candidaturas para PB
   52.61% de los votos a candidaturas para PC
    3.00% de los votos a candidaturas para PD
```

Ejercicio 14

Escribe un programa que sirva para calcular, en el recorrido de una carrera de fondo, cierta información sobre los desniveles a los que se han de enfrentar los corredores. El programa debe proceder del siguiente modo:

- Primero, debe leer la altura sobre el nivel del mar de cada punto kilométrico del recorrido. Para ello, tienes que crear y utilizar una función, `leer_alturas`, que lea las alturas de cada punto hasta que se introduzca una cadena vacía y devuelva como resultado una lista con las alturas leídas.
- Después, debe construir una lista donde, por cada kilómetro de la carrera, se almacene el desnivel al que los corredores se enfrentan en ese kilómetro. Crea y utiliza otra función, `calcular_desniveles`, que reciba como parámetro la lista de alturas y devuelva la correspondiente lista de desniveles.
- También debe calcular cuál es el kilómetro con mayor desnivel de subida. Crea y utiliza una nueva función, `posición_máximo`, que devuelva la posición del máximo elemento de la lista que reciba como parámetro.

- Finalmente, ha de mostrar las listas construidas y la información referente al kilómetro con mayor desnivel de subida.

El tipo de salida deseado se muestra en el ejemplo siguiente:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1: 1000
Altura en metros en el punto kilométrico 2: 950
Altura en metros en el punto kilométrico 3: 1025
Altura en metros en el punto kilométrico 4: 1050
Altura en metros en el punto kilométrico 5: 1100
Altura en metros en el punto kilométrico 6: 975
Altura en metros en el punto kilométrico 7: 1025
Altura en metros en el punto kilométrico 8: 1075
Altura en metros en el punto kilométrico 9: 1125
Altura en metros en el punto kilométrico 10: 1175
Altura en metros en el punto kilométrico 11: 1250
Altura en metros en el punto kilométrico 12: 1200
Altura en metros en el punto kilométrico 13:
Lista de alturas: [1000, 1000, 950, 1025, 1050, 1100, 975, 1025, 1075, 1125, 1175, 1250, 1200]
Lista de desniveles: [0, -50, 75, 25, 50, -125, 50, 50, 50, 50, 75, -50]
Kilómetro con mayor desnivel de subida:
  Entre los puntos kilométricos 10 y 11
  Desnivel de 75 m
```

Obsérvese en el ejemplo anterior que, en caso de empate a desnivel, se escoge el kilómetro más próximo a la meta. Por otra parte, el siguiente ejemplo muestra el comportamiento deseado si ningún kilómetro es de subida:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1: 1000
Altura en metros en el punto kilométrico 2: 950
Altura en metros en el punto kilométrico 3: 925
Altura en metros en el punto kilométrico 4: 900
Altura en metros en el punto kilométrico 5: 900
Altura en metros en el punto kilométrico 6:
Lista de alturas: [1000, 1000, 950, 925, 900, 900]
Lista de desniveles: [0, -50, -25, -25, 0]
Ningún kilómetro es de subida
```

Y también hay que prever que el usuario no llegue a introducir al menos dos puntos kilométricos:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1:
Recorrido no válido, con menos de dos puntos
```

Lógicamente, cuando se da este caso, el programa no debe llamar ni a `calcular_desniveles` ni a `posición_máximo`.

Ejercicio 15 (Obligatorio)

Escribe un programa similar al anterior pero que, en vez de calcular el kilómetro con mayor desnivel de subida, proporcione en su salida información referente a tramos de subida. Considera que un tramo de subida es una sucesión de kilómetros consecutivos en el recorrido donde todos y cada uno de ellos presentan un desnivel estrictamente positivo. La nueva información que se desea calcular y mostrar es, por cada tramo de subida detectado:

- Los puntos kilométricos que limitan ese tramo.

- Su longitud.
- Su desnivel.

Debes leer los datos utilizando la función `leer_alturas` que definiste en el ejercicio anterior y debes crear un nuevo procedimiento, `mostrar_tramos_subida`, que reciba la lista de alturas y muestre la información deseada. Para escribir este procedimiento, te resultará cómodo escribir también un procedimiento, `mostrar_tramo`, que reciba el número de tramo, su inicio, longitud y desnivel y muestre esa información por pantalla. Un ejemplo de ejecución del programa es:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1: 1000
Altura en metros en el punto kilométrico 2: 950
Altura en metros en el punto kilométrico 3: 1025
Altura en metros en el punto kilométrico 4: 1050
Altura en metros en el punto kilométrico 5: 1100
Altura en metros en el punto kilométrico 6: 975
Altura en metros en el punto kilométrico 7: 1025
Altura en metros en el punto kilométrico 8: 1075
Altura en metros en el punto kilométrico 9: 1125
Altura en metros en el punto kilométrico 10: 1175
Altura en metros en el punto kilométrico 11: 1175
Altura en metros en el punto kilométrico 12: 1200
Altura en metros en el punto kilométrico 13:
Lista de alturas: [1000, 1000, 950, 1025, 1050, 1100, 975, 1025, 1075, 1125, 1175, 1175, 1200]
Tramo de subida número 1:
  Entre los puntos kilométricos 2 y 5
  Longitud de 3 km
  Desnivel de 150 m
Tramo de subida número 2:
  Entre los puntos kilométricos 6 y 10
  Longitud de 4 km
  Desnivel de 200 m
Tramo de subida número 3:
  Entre los puntos kilométricos 11 y 12
  Longitud de 1 km
  Desnivel de 25 m
```

Otro ejemplo es:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1: 1000
Altura en metros en el punto kilométrico 2: 950
Altura en metros en el punto kilométrico 3: 925
Altura en metros en el punto kilométrico 4: 900
Altura en metros en el punto kilométrico 5: 900
Altura en metros en el punto kilométrico 6:
Lista de alturas: [1000, 1000, 950, 925, 900, 900]
Ningún kilómetro es de subida
```

Y otro más:

```
Ve introduciendo alturas sobre el nivel del mar, o una cadena vacía para acabar...
Altura en metros en el punto kilométrico 0: 1000
Altura en metros en el punto kilométrico 1:
Recorrido no válido, con menos de dos puntos
```

Ejercicio 16

Escribe una función, `es_dígito`, que reciba como parámetro un carácter y devuelva como resultado `True` si se trata de un dígito y `False` en caso contrario.

Escribe otra función, `secuencias_dígitos`, que reciba como parámetro una cadena y, usando la función `es_dígito`, devuelva como resultado una lista de cadenas formada por cada una de las secuencias de dígitos consecutivos que aparecen en la cadena dada.

Escribe un programa que lea una cadena y, haciendo uso de la función anterior, muestre una lista con las secuencias de dígitos que aparecen en la misma. Un ejemplo de ejecución del programa es:

```
Introduce una cadena: 10, 15, 20, 25, 30, 35, ...
La lista de secuencias obtenida es ['10', '15', '20', '25', '30', '35']
```

Otro ejemplo es:

```
Introduce una cadena: 12*x-5*y=8.75
La lista de secuencias obtenida es ['12', '5', '8', '75']
```

Ejercicio 17 (Obligatorio)

Escribe una función, `evaluación_test`, que reciba como parámetros dos cadenas que representen la plantilla de corrección y las respuestas de un examen de tipo test, donde cada respuesta se representa mediante un único carácter y las preguntas no contestadas se representan mediante asteriscos. La función debe devolver como resultado una lista con tres números enteros que representen, en este orden, las cantidades de respuestas correctas, de respuestas erróneas y de preguntas no contestadas. Si las longitudes de las cadenas dadas son distintas, la función debe devolver `None`.

Modifica una copia del programa del ejercicio 23 de la práctica 3 para que use la función definida. Recuerda que el programa que allí se te pedía debía funcionar como muestra el siguiente ejemplo:

```
Introduce la plantilla de respuestas correctas: cbbaabcdcc
Ve introduciendo respuestas de alumnos, o vacío para acabar...
Nuevas respuestas: cbababcd
La cadena introducida es de longitud 8 (se esperaba 10)
Nuevas respuestas: cbababcd**
Resultados: 6 BIEN; 2 MAL; 2 NS/NC
Nuevas respuestas: a*a*a*aaaa
Resultados: 1 BIEN; 6 MAL; 3 NS/NC
Nuevas respuestas: *bbbbbbbbb
Resultados: 3 BIEN; 6 MAL; 1 NS/NC
Nuevas respuestas: cbba*bcdcc
Resultados: 9 BIEN; 0 MAL; 1 NS/NC
Nuevas respuestas: exit
La cadena introducida es de longitud 4 (se esperaba 10)
Nuevas respuestas:
Alumnos corregidos: 4
```

Además, por cada cadena de respuestas introducida por el usuario, el programa debe llamar *una única vez* a `evaluación_test`.

Ejercicio 18

En los siguientes ejercicios vas a utilizar matrices. Para facilitar que puedas trabajar con ellas leyéndolas de fichero, te proporcionamos vía *Aula Virtual* algunos ficheros de texto y un módulo `modulomatrices.py` que vas a probar en este ejercicio. Así pues, copia y ejecuta el programa que aparece a continuación:

```
from modulomatrices import leerMatrizEnteros, mostrarMatriz

nombreFichero = input('Introduce el nombre de un fichero: ')
matriz = leerMatrizEnteros(nombreFichero)
mostrarMatriz(matriz)
```

He aquí un ejemplo de ejecución, resultante de introducir `matriz1.txt` como nombre de fichero:

```
Introduce el nombre de un fichero: matriz1.txt
-153    -3    94    27   -81   107     7
  74   170  -185  -174  -162    -7   193
  91  -215  -159  -241   -31    90   -79
  77   173   -29   205   -23   -20   -22
 195  -112   133    83    -6   223    78
```

Ejercicio 19

Escribe una función, `sumar_elementos`, que reciba como parámetro una matriz de enteros y devuelva como resultado la suma de todos sus elementos.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz, utilice la función anterior para calcular la suma de todos sus elementos y muestre el resultado.

Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz1.txt
La suma de todos los elementos en la matriz es 318
```

Ejercicio 20

Escribe una función, `sumar_fila`, que reciba como parámetros una matriz de enteros y un número de fila y devuelva como resutado la suma de los elementos de esa fila.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz, utilice la función anterior para calcular la suma de cada fila y que muestre los resultados. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz1.txt
La suma de los elementos en la fila 0 es -2
La suma de los elementos en la fila 1 es -91
La suma de los elementos en la fila 2 es -544
La suma de los elementos en la fila 3 es 361
La suma de los elementos en la fila 4 es 594
```

Ejercicio 21

Escribe una función, `sumar_columna`, que reciba como parámetros una matriz de enteros y un número de columna y devuelva como resutado la suma de los elementos de esa columna.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz, utilice la función anterior para calcular la suma de cada columna y que muestre los resultados. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz1.txt
La suma de los elementos en la columna 0 es 284
La suma de los elementos en la columna 1 es 13
La suma de los elementos en la columna 2 es -146
La suma de los elementos en la columna 3 es -100
La suma de los elementos en la columna 4 es -303
La suma de los elementos en la columna 5 es 393
La suma de los elementos en la columna 6 es 177
```

Ejercicio 22

Escribe una función, `sumar_diagonal`, que reciba como parámetro una matriz de enteros y devuelva como resultado la suma de los elementos de la diagonal principal cuando la matriz es cuadrada y `None` cuando no lo sea. La función debe contar con *un único bucle*, no es correcto emplear dos.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz cuadrada, utilice la función anterior para calcular la suma de los elementos de la diagonal principal y escriba el resultado. Si la matriz no es cuadrada, se debe mostrar un mensaje de error. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz1.txt
Error. Se requiere una matriz cuadrada
```

Otro ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz2.txt
La suma de los elementos en la diagonal principal es 12
```

Ejercicio 23 (Obligatorio)

Escribe una función, `producto_diagonal_secundaria`, que reciba como parámetro una matriz de enteros y devuelva como resultado el producto de los elementos en la diagonal secundaria de la matriz cuando la matriz sea cuadrada y `None` cuando no lo sea. La función debe contar con *un único bucle*, no es correcto emplear dos.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz cuadrada, utilice la función anterior para calcular el producto de los elementos de la diagonal secundaria y escriba el resultado. Si la matriz no es cuadrada, se debe mostrar un mensaje de error. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz2.txt
El producto de los elementos en la diagonal secundaria es -143528678807602838400
```

Un ejemplo de ejecución con una matriz no cuadrada es:

```
Introduce el nombre de un fichero: matriz1.txt
Error. Se requiere una matriz cuadrada
```

Ejercicio 24 (Obligatorio)

Escribe una función, `sumar_encima_diagonal`, que reciba como parámetro una matriz de enteros y devuelva como resultado la suma de los elementos situados encima de la diagonal principal de la matriz (sin incluir los elementos de la diagonal) cuando la matriz sea cuadrada y `None` cuando no lo sea.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz cuadrada, utilice la función anterior para calcular la suma de los elementos situados encima de la diagonal principal y escriba el resultado. Si la matriz no es cuadrada, se debe mostrar un mensaje de error. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz2.txt
La suma de los elementos por encima de la diagonal principal es 850
```

Un ejemplo de ejecución con una matriz no cuadrada es:

```
Introduce el nombre de un fichero: matriz1.txt
Error. Se requiere una matriz cuadrada
```

Ejercicio 25

Escribe una función, `sumar_debajo_diagonal`, que reciba como parámetro una matriz de enteros y devuelva como resultado la suma de los elementos situados debajo de la diagonal principal de la matriz (sin incluir los elementos de la diagonal) cuando la matriz sea cuadrada y `None` cuando no lo sea.

Escribe un programa que lea el nombre de un fichero que contenga los datos de una matriz cuadrada, utilice la función anterior para calcular la suma de los elementos situados debajo de la diagonal principal y escriba el resultado. Si la matriz no es cuadrada, se debe mostrar un mensaje de error. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de un fichero: matriz2.txt
La suma de los elementos por debajo de la diagonal principal es 239
```

Ejercicio 26

En los siguientes ejercicios vas a utilizar matrices para representar imágenes. Cada elemento de la matriz representará un punto de la imagen y almacenará un valor entero entre 0 (negro) y 255 (blanco). Los valores intermedios representarán los diferentes niveles de gris. Para facilitar que puedas trabajar con matrices que representan imágenes, te proporcionamos vía *Aula Virtual* algunos ficheros de texto y un módulo `moduloimagen.py` que vas a probar en este ejercicio. Así pues, copia y ejecuta el programa que aparece a continuación:

```
from moduloimagen import leerImagen, mostrarImagen

nombreFichero = input("Introduce el nombre de la imagen: ")
matriz = leerImagen(nombreFichero)
mostrarImagen(matriz)
```

He aquí un ejemplo de ejecución, resultante de introducir `tux.txt` como nombre del fichero de imagen:

```
Introduce el nombre de la imagen: tux.txt
```

Y la siguiente imagen se mostrará por pantalla:

De no ser así, ten en cuenta que `moduloimagen.py` hace uso, a su vez, de `easycanvas.py`, es decir, necesitas tener ambos módulos para poder utilizar el primero.

Ejercicio 27

Escribe un procedimiento, `aplicar_filtro_negativo`, que reciba com parámetro una imagen y la modifique aplicando un filtro de negativo a la imagen. Un *filtro de negativo* sustituye el valor c de cada punto por su valor opuesto (es decir, por $255 - c$).

Escribe un programa que lea el nombre de un fichero con una imagen y la muestre en pantalla tras aplicarle el procedimiento anterior.

He aquí un ejemplo de ejecución:

```
Introduce el nombre de la imagen: tux.txt
```

Y, como resultado de aplicar a `tux.txt` un filtro de negativo, debe obtenerse esta nueva imagen por pantalla:

Ejercicio 28 (Obligatorio)

Escribe una función, `nueva_matriz`, que reciba como parámetros tres números enteros que representan las dimensiones (número de filas y número de columnas) de una matriz y un valor inicial. La función debe construir y devolver una matriz de las dimensiones dadas en la que todos sus elementos contengan el valor dado.

Escribe otra función, `binarizada`, que reciba como parámetros una matriz que representa una imagen y un número entero que representa un umbral y devuelva como resultado una nueva matriz que representa la imagen binarizada a partir del umbral dado. Para crear la nueva matriz, `binarizada` debe hacer uso de `nueva_matriz`. Una *binarización* modifica la imagen de modo que todos los valores menores o iguales que c quedan en negro (0) y el resto queda en blanco (255).

Escribe un programa que lea el nombre de un fichero con una imagen y un número entero c, mediante la función anterior binarice la imagen y después la muestre por pantalla.

He aquí un ejemplo de ejecución:

```
Introduce el nombre de la imagen: tux.txt
Introduce el umbral:170
```

Y esta imagen es la que se debe mostrar por pantalla:

Ejercicio 29 (Obligatorio)

Escribe un procedimiento, `reflejar_horizontal`, que reciba como parámetro una imagen y la modifique para obtener su reflexión horizontal. Una *reflexión horizontal* modifica la imagen para que quede tal y como se vería reflejada en un espejo. Para ello, es necesario invertir los elementos de cada fila de la matriz.

Escribe un programa que lea el nombre de un fichero con una imagen y la muestre en pantalla tras aplicarle el procedimiento anterior.

Al aplicar a la imagen `tux.txt` una reflexión horizontal, debes obtener la nueva imagen que muestra este ejemplo de ejecución, donde Tux ahora mira hacia el otro lado:

```
Introduce el nombre de la imagen: tux.txt
```

Ejercicio 30 (Obligatorio)

Escribe un procedimiento, `reflejar_vertical`, que reciba como parámetro una imagen y la modifique para obtener su reflexión vertical.

Escribe un programa que lea el nombre de una imagen y, utilizando el procedimiento definido, muestre su reflexión vertical. He aquí el correspondiente ejemplo de ejecución sobre la imagen `tux.txt`:

```
Introduce el nombre de la imagen: tux.txt
```

Ejercicio 31

Escribe una función, `promedio_entorno`, que reciba como parámetros una imagen, un número de fila y un número de columna y devuelva como resultado el valor medio en el entorno del correspondiente punto, esto es, el promedio de los nueve puntos formados por el punto indicado por la fila y columna dadas y sus ocho puntos vecinos. Supón que los valores de fila y columna dados serán válidos y que nunca se encontrarán en los bordes de la imagen, es decir, siempre existirán los ocho vecinos del punto dado.

Escribe un programa que lea el nombre de una imagen, un número de fila y un número de columna y, utilizando la función definida, muestre con dos decimales el promedio en el entorno del punto dado. Un ejemplo de ejecución del programa es:

```
Introduce el nombre de la imagen: tux.txt
Introduce un número de fila: 52
Introduce un número de columna: 15
El promedio en el entorno del punto (52,15) es 57.22
```

Y he aquí otro ejemplo de ejecución del mismo programa:

```
Introduce el nombre de la imagen: tux.txt
Introduce un número de fila: 35
Introduce un número de columna: 30
El promedio en el entorno del punto (35,30) es 240.67
```

Ejercicio 32 (Obligatorio)

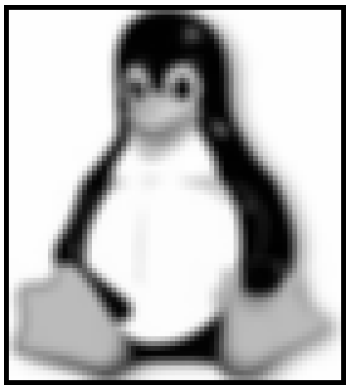
Escribe una función, `difuminada`, que reciba como parámetros una imagen y un número entero y devuelva como resultado una nueva imagen que cumpla lo siguiente:

- Todos los puntos del borde deben contener el valor del entero dado.
- Cada uno de los otros puntos (es decir, los que no están en el borde) debe contener el promedio calculado en el entorno del correspondiente punto de la imagen dada, redondeado al entero más cercano.

Para ello, `difuminada` debe hacer uso de las funciones anteriormente definidas `nueva_matriz` (del ejercicio 28) y `promedio_entorno` (del ejercicio anterior).

Escribe un programa que haga uso de `difuminada` en su programa principal y cuyo comportamiento consista en leer el nombre de una imagen y un número entero y, después, mostrar el resultado de difuminar esa imagen utilizando, para rellenar los bordes de la nueva imagen, el entero dado. He aquí un ejemplo de ejecución del programa:

```
Introduce el nombre de la imagen: tux.txt
Introduce el valor del borde: 0
```



Ejercicio 33 (Obligatorio)

Escribe una función, `girada_izquierda`, que reciba como parámetro una imagen y devuelva como resultado una nueva imagen que se corresponda con la imagen dada girada hacia la izquierda (en sentido antihorario). Esta función debe a su vez utilizar la función `nueva_matriz`, definida en el ejercicio 28.

Escribe un programa que lea el nombre de una imagen y, utilizando la función `girada_izquierda` en su programa principal, muestre el resultado de girar esa imagen hacia la izquierda. He aquí el correspondiente ejemplo de ejecución sobre la imagen `tux.txt`:

```
Introduce el nombre de la imagen: tux.txt
```