**Crea una carpeta con tu nombre en el escritorio y almacena en ella todos los ejercicios. Cada ejercicio deberá realizarse en un proyecto independiente.**

**Se valorará en todas las preguntas: las estructuras de datos elegidas, los módulos diseñados, la claridad, los comentarios y por supuesto que se ejecute correctamente haciendo lo indicado.**



**1º Alea Iacta Est.**

Son famosas las formaciones que el antiguo ejército romano utilizaba para entrar en batalla. En esas formaciones, los legionarios se agrupaban en una figura geométrica (normalmente un rectángulo) y protegían **tanto los flancos como la parte superior utilizando escudos.** Los legionarios que ocupaban **posiciones interiores cubrían la parte superior colocando el escudo sobre su cabeza**, mientras que **los que ocupaban los flancos llevaban dos y hasta tres escudos**: uno para proteger la parte superior y uno o dos escudos (si estaban en la esquina) para proteger los laterales. En esta formación todos los legionarios quedaban protegidos por los escudos y eran muy difíciles de vencer.

Cuenta la historia que existió un general que estableció que la mejor figura para la formación no era la rectangular sino la **cuadrada**, de forma que el número de filas y columnas de legionarios coincidía.

El problema al que se enfrentaba este general era decidir en cuántas formaciones (y de qué tamaño) debía separar su ejército para que:

* No quedará **ningún legionario fuera** de una formación (aunque admitía formaciones de un único legionario; en este caso ese legionario debía portar 5 escudos!!!!).
* Se minimizará el número de escudos necesarios para protegerlos.

Nuestro general, después de hacer muchos cálculos, decidió que la mejor manera de que estas dos condiciones se cumpliesen era **comenzar haciendo el cuadrado más grande posible con sus legionarios. Con los que le quedasen libres volvía a repetir la operación, y así hasta que no quedasen legionarios que formar**.

Por ejemplo, si el número de legionarios en el ejército era 35, la manera utilizada por el general para hacer la formación consistía en un cuadrado de 25 legionarios (5x5), otro de 9 (3x3) y otro de 1 (1x1):



Esta formación requería un total de 71 escudos.

Se pide un **programa modular** que **pida el número total de legionarios** y nos **calcule** **los cuadrados que hay que ir haciendo con su dimensión**, así como el **número total de escudos** necesarios en total.

*(3 puntos)*

**2º Cierre de seguridad.**

Vamos a realizar la simulación del **mecanismo de un cierre de seguridad**. Dicho cierre consta de 5 dígitos**. Cada dígito de cada posición** se obtiene de la **suma de otros dos dígitos** de una serie de 50 números (entre 1 y 99 generados al azar) que se pueden elegir para cada posición. La forma de pedir la combinación para entrar es un **grupo de 3 números** (se deben pedir 5 grupos). En dichos grupos el primer número indica la posición inicial, el segundo dígito el desplazamiento hacia la izquierda desde dicha posición (tendremos el primer dígito a sumar) y el segundo número el desplazamiento a la derecha desde la posición actual (obtendremos el segundo dígito a sumar); se debe comprobar que los desplazamientos son correctos, en el caso de que algún desplazamiento no se pueda hacer se asumirá un 0 para ese valor. Para obtener el segundo y sucesivos números de la combinación se procede de la misma forma.

**Se deben pedir los 5 grupos de 3 números y devolver la combinación resultante. Diseñado todo de forma modular.**

**Ejemplo:**

Números entre los que elegir (generados al azar).

| 2 | 36 | 18 | 63 | 12 | 9 | 74 | … | 81 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |

**Primer grupo:**

| 2 | 1 | 3 |
| --- | --- | --- |

El primer número de la combinación será: (uno a la izquierda desde la posición 2 (18): 36 + 3 a la derecha desde la posición actual (la del 36) 12 = 48).

**Segundo grupo:**

| 4 | 4 | 2 |
| --- | --- | --- |

Segundo dígito: 24 + 18 = 42.

**Segundo grupo:**

| 0 | 3 | 6 |
| --- | --- | --- |

Como desde la posición 0 no se puede ir tres posiciones a la izquierda, se asume 0 para el tercer dígito.

Así con los 2 grupos restantes.

**Finalmente se muestra la combinación resultante**.

*(3,5 puntos)*

**3º Gestionando cadenas.**

Vamos a realizar un programa que muestre el siguiente menú hasta que se pulse la opción de salir:

**1.- Almacenar cadena.**

**2.- Buscar cadena.**

**3.- Buscar cadenas que comiencen por…**

**4.- Reemplazar cadena por otra.**

**5.- Listar todas las cadenas.**

**6.- Buscar expresión.**

**7.- Salir.**

Se deben almacenar un conjunto de ***Strings*** 20. Para cada opción se realiza lo siguiente:

**1.- Almacenar cadena.** 🡪 Se pide una cadena por teclado y se almacena al final de la lista.

**2.- Buscar cadena.** 🡪 Se pide una cadena por teclado y se busca en la lista. Caso de estar se devolverá su posición, caso de repetirse se devolverá la primera y caso de no estar se devolverá -1.

**3.- Buscar cadenas que comiencen por…** 🡪 Parecido al punto anterior. Se pide una cadena y se devolverá cuantas cadenas hay en la lista que empiecen por esa expresión. Se indicara también la posición que ocupa, caso de repetirse se devolverá la última posición.

**4.- Reemplazar cadena por otra.** 🡪 Se piden dos cadenas por teclado; la primera se reemplazará por la segunda en todas las cadenas de la lista que la contengan.

**5.- Listar todas las cadenas** 🡪 Realizará exactamente eso: listar todas las cadenas almacenadas hasta el momento.

**6.- Buscar expresión.** 🡪 Se debe pedir una expresión o conjunto de palabras y se debe comprobar que todas esas palabras están en la lista de palabras almacenadas. Indicar simplemente si esto ocurre o no.

**7.- Salir** 🡪 Sale del programa.

El programa debe ser modular, es decir: para cada opción se debe diseñar, al menos, un módulo.

*(3,5 puntos)*

“GGggnngnrrrnnggnr hhr hgn gngnngrrrh grgrnnhrn”

**Chewbacca** – “El Imperio contraataca”