Práctica Calificada 2

1. Diseñe una función que reciba una oración, de no más de 200 caracteres, que contiene tanto palabras como números enteros de hasta 3 cifras. Asuma que las palabras no contienen dígitos numéricos, ni letras tildadas ni caracteres especiales, ni signos de puntuación.
   Luego ejecute lo siguiente:

   Calcule la suma de los números enteros en dicha oración (s).

   Obtenga el resto (r) de la división entera entre dicha suma (s) y el número 5. Luego a dicho resto (r) le sume 1 y obtenga un nuevo número (n).

   Luego determine cuantas palabras en la oración tienen “n” vocales diferentes. Debe imprimir en pantalla un mensaje indicando cuantas palabras tienen “n” vocales diferentes y la función deber retornar la cantidad de palabras calculadas.

   Ejemplo:

   Se recibe la siguiente oración:

   Mis primos tienen casi 40 anios yo tengo 34 y tenemos que jugar pelota con gente de 18 esta bien dificil ganar

   Debe retornar el número 2 (que indica que hay 2 palabras con 3 vocales diferentes)
   El programa principal debe imprimir **Hay 2 palabras con 3 vocales diferentes**
   Nota: s = 40+34+18 = 92; r = 92 mod 5 = 2; n = r+1 = 3;

2. **Enchufe**
   Renato piensa muy a menudo sobre el significado de la vida. Lo hace constantemente, incluso cuando escribe en el editor. Cada vez que empieza a meditar, ya no puede concentrarse completamente y presiona repetidamente las teclas que deberían presionarse solo una vez. Por ejemplo, en lugar de la frase "how are you" puede escribir "hhoow aaaare yyoouu".
   Renato decidió automatizar el proceso de corregir tales errores. Decidió escribir un complemento para el editor de texto que eliminará pares de letras idénticas consecutivas (si las hay en el texto). Por supuesto, esto no es exactamente lo que Renato necesita, ¡pero tiene que empezar por algo!
   Ayuda a Renato y escribe el módulo principal del complemento. Tu programa debe eliminar de una cadena todos los pares de letras idénticas que sean consecutivas. Si después de la eliminación aparecen nuevos pares, el programa también debe eliminarlos. Técnicamente, su funcionamiento debería ser equivalente a lo siguiente: mientras la cadena contenga un par de letras idénticas consecutivas, el par debe ser eliminado. Ten en cuenta que la eliminación de las letras idénticas consecutivas se puede hacer en cualquier orden, ya que cualquier orden lleva al mismo resultado.

   Entrada
   Los datos de entrada consisten en una sola línea para procesar. La longitud de la línea es de 1 a $2 \cdot 10^5$ caracteres inclusive. La cadena contiene solo letras latinas minúsculas.

   Salida
   Imprime la cadena dada después de ser procesada. Se garantiza que el resultado contendrá al menos un carácter.

Ejemplos

input
hhoowaaaareyyoouu
output
wre

input
reallazy
output
rezy

input
abacabaabacabaa
output
a

3. Diseñe un algoritmo que permita recibir 2 números enteros positivos (a y b) de 4 o más cifras y un número entero positivo de una sola cifra (c), estos valores deben ser validados en la entrada de datos. Luego el algoritmo debe combinar los 2 primeros números (a y b) en uno sólo (d) con las siguientes consideraciones:
   Los dígitos del número (d) se encuentren ordenados de forma decreciente de derecha a izquierda.
   En caso existan dígitos iguales en los números (a) y (b) sólo se ponen los dígitos del primer número ingresado (a), es decir, las veces que este dígito aparezca será la cantidad de veces que este dígito aparece en el primer número (a).
   Se debe mostrar por pantalla el número (d) resultante de la mezcla de los números (a) y (b).
   Finalmente, se debe eliminar del número mezclado (d) todas las apariciones del dígito (c).
   Ejemplo 1:
   Se ingresan los números: 145652, 57983272 y 7
   El número mezclado será: 12345567789
   El número resultante será: 123455689

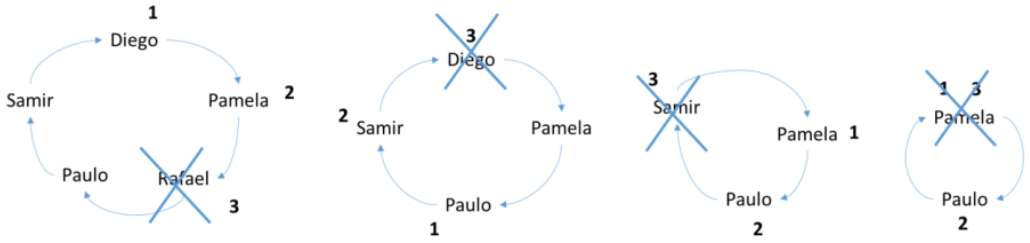
   Ejemplo 2:
   Se ingresan los números: 137288, 2974892 y 3
   El número mezclado será: 123478899
   El número resultante será: 12478899

4. Espías Rusos

   N espías rusos han sido descubiertos y acorralados por las fuerzas de inteligencia enemigas. Los espías no revelarán ningún secreto y para evitar ser torturados se quitarán la vida - uno a uno ingiriendo píldoras suicidas.

   Los espías formarán un círculo, elegirán un número **K > 1**, y contarán desde 1 hasta K en sentido horario. El espía que inicie la cuenta dirá 1 en voz alta, el siguiente espía en orden horario dirá 2, el siguiente dirá 3, y continuarán en forma sucesiva, hasta que el espía que pronuncie el número K ingerirá una píldora suicida y fallecerá en el acto. Luego de ello, la cuenta se reiniciará en 1 desde el siguiente espía en sentido horario, y el proceso se repetirá hasta que todos se suiciden.

   Se sabe, también, que el último espía en quedar vivo podría elegir colaborar con las fuerzas enemigas y sobrevivir, puesto que ningún otro espía podría cuestionar su decisión. Por ejemplo, considerando los siguientes N = 5 espías y K = 3, se observa que si Diego empieza la cuenta, Paulo será el último sobreviviente, y podría decidir colaborar con el enemigo para sobrevivir.



Desarrolle un programa que represente a los N espías como nodos enlazados en forma circular y horaria, y determine el último espía en sobrevivir

Estructura Espia	**Caso de Ejemplo**

`struct Espia {`
`  string nombre;`
`  Espia* siguiente;`
`  Espia(string _nombre) {`
`  nombre = _nombre;`
`  siguiente = NULL;`
`  }`
`};`
	`Espia* pDiego = new Espia("Diego");`
	`Espia* pPamela = new Espia("Pamela");`
	`Espia* pRafael = new Espia("Rafael");`
	`Espia* pPaulo = new Espia("Paulo");`
	`Espia* pSamir = new Espia("Samir");`
	
	`pDiego->siguiente = pPamela;`
	`pPamela->siguiente = pRafael;`
	`pRafael->siguiente = pPaulo;`
	`pPaulo->siguiente = pSamir;`
	`pSamir->siguiente = pDiego;`

Su programa debe incluir las siguientes funciones:

a. Cantidad de espías: Dado un espía inicial cualquiera, determinar la cantidad total de espías, recorriendo los nodos enlazados hasta encontrar algún nodo repetido.

```
int numeroEspias(Espia* espiaInicial)
```

b. Avanzar X pasos: Dado un espía inicial cualquiera, retornar el espía que se ubique X pasos en sentido horario.

```
Espia* avanzar(Espia* espiaInicial, int X)
```

c. Último sobreviviente: Dado un espía inicial cualquiera, y el número K, retornar el último espía sobreviviente.

```
Espia* ultimoSobreviviente(Espia* espiaInicial, int K)
```

5. Dados dos polinomios de grados N y M, se desea almacenar los coeficientes de dichos polinomios y luego mostrar los coeficientes de los productos de estos.

   Ejemplo:

   P(x)=3x^3-4x^2+1 => coeficientes: 3, -4, 0, 1

   Q(x)=-2x^2+x => coeficientes: -2, 1, 0

   Producto:

   P(x)Q(x)=-6x^5+11x^4-4x^3-2x^2+x => Coeficientes: -6, 11, -4, -2, 1, 0

6. Se desea almacenar en arreglos de caracteres los dígitos de dos números del sistema hexadecimal de m y n cifras respectivamente.
   Luego almacenar en otro arreglo de caracteres la suma dicho números y finalmente presentar los dígitos de la suma.

   Nota: No pasar los números a base 10. El producto debe realizar en el sistema Hexadecimal.
   Ejemplo:
   Primer número con n=4: AB10F
   Segundo número con m=6:FF041F
   Suma será: 109B52E

7. Para un conjunto de N empleados se desea almacenar en un arreglo unidimensional el código del empleado (entero positivo de 3 dígitos) y seguidamente el número de horas extras (entero positivo de 2 dígitos) por cada mes en las que tuvo horas extras. Esto significa que no todos los empleados tendrán el mismo número de meses con horas extras. Se pide especificar un programa C++ que permita almacenar la información de los empleados, muestre el arreglo, después calcule el promedio de horas extras y luego elimine del arreglo los datos de los empleados cuyo total de horas extras es superior al promedio. Finalmente muestre el arreglo después de la eliminación. Debe resolverlo utilizando un solo arreglo (utilizar sólo arrays simples, no Contenedores de STL ni algoritmos de STL)

   Por ejemplo, para N=3:
   1er empleado código 200, horas extras: 40 y 40 (tuvo 2 meses con horas extras)
   2do empleado código 150, horas extras: 20, 10 y 20 (tuvo 3 meses con horas extras)
   3er empleado código 300, horas extras: 20, 30, 10 y 20 (tuvo 4 meses con horas extras)

Empleado 1			Empleado 2				Empleado 3					
	Mes			Mes				Mes				
Cod	1	2	Cod	1	2	3	Cod	1	2	3	4	
200	40	40	150	20	10	20	300	20	30	10	20	...

   De donde se tiene: Total de horas extras de los empleados 1, 2 y 3 es 80, 50 y 80 respectivamente. El promedio de las horas extras por empleado es 70,00. Los empleados con código 200 y 300 superan el promedio y deben ser eliminados. Luego de la eliminación solo quedan los datos del empleado con código 150:

150	20	10	10	...

8. Un *doblete* es una pareja de palabras en la que una de ellas se diferencia de la otra por una sola letra; por ejemplo, “booster" y “rooster", o “rooster" y “roaster", o “roaster" y “roasted". Se proporcionará un diccionario de hasta 25.143 palabras en minúsculas en el que ninguna de ellas superarán las 16 letras. A continuación, se mostrarán parejas de palabras. La tarea consistirá en encontrar, para cada pareja de palabras, la secuencia más corta que comience con la primera palabra y finalice con la segunda, de forma que cada palabra y su adyacente formen un doblete. Por ejemplo, si se muestra la pareja “booster" y “roasted", una solución posible sería (“booster", “rooster", “roaster", “roasted"), asumiendo que todas esas palabras estuviesen incluidas en el diccionario.

Entrada
La entrada constará del diccionario seguido de parejas de palabras. El diccionario incluirá varias palabras, una por línea, y finalizará con una línea en blanco. A continuación, aparecerán las parejas de palabras, cada una de ellas en una línea independiente.

Salida
Mostrar, por cada pareja de la entrada, un conjunto de líneas que comience con la primera palabra y finalice con la ´ultima. Cada pareja de líneas adyacentes debe formar un doblete. Si existen varias soluciones posibles que muestren una secuencia de la menor longitud, cualquiera de ellas será válida. Si no hay solución se mostrará la línea: “No solution.". Entre cada dos casos se dejará una línea en blanco.

Ejemplo de entrada
booster
rooster
roaster
coasted
roasted
coastal
postal
booster roasted
coastal postal

Ejemplo de salida
booster
rooster
roaster
roasted
No solution.

9. Progresión Aritmética (5 puntos)

Dado un arreglo X de n números enteros, utilice la mayor cantidad posible de dichos números para formar una progresión aritmética y una progresión geométrica. Su programa debe leer el arreglo X, y debe imprimir ambas progresiones, de modo que la cantidad total de números en ambas sea máxima. En caso de haber múltiples opciones puede imprimir cualquiera.
`void imprimirProgresiones(int X[], int n)`

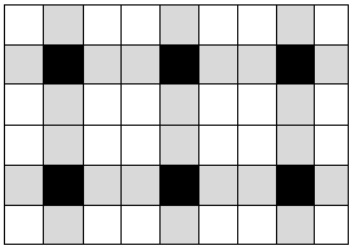
Input	Output
X = {1, 10, 9, 5, 7, 13, 100} X = {5, 5, 5, 5, 5, 1} X = {0, 10, 3, 8, 9, 6} X = {100}	A: 1 5 9 13 - G: 10 100 A: 5 5 5 5 5 – G: 1 A: 6 8 10 – G: 3 9 A: 100 – G:

10. Comprimir Mosaico (7 puntos)

Una imagen en escala de grises puede ser representada por una matriz de 2 dimensiones con valores enteros, donde el valor 0 corresponda al blanco, el valor 255 al negro, y los valores intermedios (1-254) a distintas tonalidades de gris.
A continuación, se presenta una imagen de 6x9 que resulta de la composición de múltiples sub imágenes idénticas de 3x3. Desarrolle un programa que reciba una imagen, y determine cuál es la imagen de menor tamaño tal que múltiples copias de dicha imagen permiten construir la imagen original.

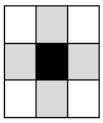
`Imagen descomponerMosaico(Imagen x)`

Input



0	120	0	0	120	0	0	120	0
120	**255**	120	120	**255**	120	120	**255**	120
0	120	0	0	120	0	0	120	0
0	120	0	0	120	0	0	120	0
120	**255**	120	120	**255**	120	120	**255**	120
0	120	0	0	120	0	0	120	0

Output



Nota: Utilice una estructura **Imagen** con los enteros nFilas, nColumnas y una matriz de enteros X.

Consideraciones:

• La tarea es grupal (grupo de 5 personas)

• Cada pregunta vale 2 puntos.

• La fecha límite de entrega es el sábado 14/12/2024 a las 18:00 horas.

• Se debe subir un archivo zip que contenga:

- La solución a cada problema (sólo los archivos .cpp, no son necesarios los .exe). Cada pregunta debe seguir la nomenclatura: **probleman.cpp**, es decir, el primer programa será **problema1.cpp**, el segundo será **problema2.cpp**, etc.
- Puede usar como workspace de inicio el que adjunto: