Introduccion

Para cada ejercicio debe realizar el analisis correspondiente. Esta primera parte es mas para una ayuda personal que entrega. Por ahora, solo debe demostrar que todas las pruebas han sido pasadas correctamente.
Recuerde los pasos:

1. Encontrar las entradas y salidas
2. Entender su relacion
3. Pseudocodigo / diagrama de flujo
4. Implementacion
5. Pruebas

En esta etapa se recomienda analizar la realacion de las entradas y salidas, para luego realizar el pseudocodigo y finalemnte su implementacion

Entregables

En el LMS debe compartir el link de su repositorio github publico a los ejercicios.

Importante: El repositorio de ejercicio debe de ser publico y compartido con el docente.

Se valorara tambien la cantidad de commits que haya realizado durante la resolucion de los ejercicios. Esto es importante, porque hacerlo en un solo commit es irrealista. Trate de que para cada ejercicio realizar el commit correspondiente. O cada dos ejercicios.

La estructura del repositorio:

- ejercicio1.cpp
- ejercicio2.cpp
- ejercicio3.cpp
- etc

Instrucciones

1. Cree un nuevo repositorio en su cuenta de github
2. Vaya realizando los ejercicio de este documento

3. Realice commit por aca ejercicio resuelto. Almacene su solucion en `ejercicio1.cpp`, `ejercicio2.cpp`, etc
4. Happy coding!

Ejercicio 1

Para buscar documentos, debe poder dividir el texto en palabras. Inicialmente, lo haremos de una forma no estandar: escriba su propio algoritmo.
Las palabras están separadas por espacios. Se garantiza exactamente un espacio entre palabras. La línea comienza con una palabra (no un espacio) y termina con el final de la línea (`\n`). Solo se ingresa una línea, el final de la línea también es uno, y alguna palabra siempre termina antes.
Lea toda la línea desde la terminal hasta una nueva línea. Para cada palabra, imprima el índice después del último carácter de la palabra. Recuerde, la indexación comienza desde cero.

Restricciones

Hay un espacio entre las palabras. No hay espacios al principio o al final de la cadena. Hay exactamente una línea en la consulta.

Entrada

```
green parrot
```

Salida

```bash
5
12
```

Los símbolos con estos índices están precedidos por las palabras verde y loro, respectivamente. (Cuente letra por letra comenzando desde 0 y verifique el resultado mostrado arriba)

Ejercicio 2

Ahora escriba las palabras encerradas entre corchetes. Por ejemplo, si la entrada fue un `green parrot`, la salida debería ser:

```
[green]
[parrot]
```

Puedes reutilizar el algoritmo del problema anterior. Allí ibas a lo largo de la línea y verificaste si el caracter era un espacio. Si el carácter no era un espacio, simplemente lo saltabas y seguía adelante. Ahora este caso tendrá que ser procesado.

Cree una variable de cadena, `word`, en la que agregará carácter por carácter de la cadena original, si este carácter no es un espacio. Si se encuentra un espacio, muestre `[`, la palabra acumulada y `]` y asigne una cadena vacía a la variable, `word`, para reiniciar el proceso con la siguiente palabra de la cadena.

Restricciones

Lo mismo que en el ejercicio anterior:

- Hay un espacio entre las palabras.
- Una línea comienza con una palabra y termina con el final de una línea.
- Hay exactamente una línea en la consulta.

Ejercicio 3

Según una encuesta realizada por `Most Accurate Statistics Bureau`, los encuestados hacen un promedio de 10 visitas al refrigerador por noche. Solo una circunstancia puede detenerlos: `Ya se acabo la torta de la heladera`.

Escriba un programa que simule las aventuras nocturnas del ciudadano. Puede averiguar si hay un pastel leyendo el número de `cin`. Solo tomará dos valores:

- 1 - todavía hay torta,
- 0 - se acabó la torta.

Si se encuentra un pastel, imprima la cadena `Om-nom-nom :P`. Si se acabó el pastel, escriba `Sin pastel :(` y finalice el ciclo con una declaración de `break`. Tambien, debe ignorarse ingresar más de 10 unidades, ya que el ciudadano promedio ya se ha quedado sin pastel en este caso.

Ejemplos

Entrada

```bash
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

Salida

```bash
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
```

```
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
```

Cuente la cantidad de `1` en esta entrada y la cantidad de `Om-nom-nom :P`.

Entrada

```bash
1 1 1 0 0 0 0 0 1 1
```

Salida

```bash
Om-nom-nom :P
Om-nom-nom :P
Om-nom-nom :P
No cake :(
```

Entrada

```bash
0
```

Salida

```bash
No cake :(
```

Ejercicio 4

Suponga que tiene un número `no negativo` y tiene que calcular su factorial.

Restricciones

El usuario puede equivocarse e introducir un numero negativo! Su codigo debe lidiar con este caso e indicarle al usuario que esto no es permitido y que lo intente de nuevo.

```
El numero es negativo. Intentelo de nuevo
```

Si se introduce un numero muy grande el programa debe advertirle al usuario que el resultado se demorara en ser calculado y pedirle que lo intente de nuevo. El numero maximo es 14.

```
El numero es muy grande. Intentelo de nuevo
```

Entrada

```bash
5
```

Salida

```bash
120
```

Ejercicio 5

Imprime el mes del calendario dado el primer día y el número de días especificados. Tu respuesta debe lucir aproximadamente así:

```
                    1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
```

Formato de entrada

Se ingresan dos números: n — el número del día de la semana del primer día del mes (un número entero de 1 a 7) y k — el número de días en este mes (un número entero de 1 a 99). . Ten en cuenta que el número de días en el mes no necesariamente tiene que ser el mismo que en el calendario convencional. El dia numero 1 es el lunes. Y el ultimo dia es el domingo.

Formato de salida

Debes imprimir el calendario como se muestra en el ejemplo. Rellena las posiciones vacías en la primera fila con espacios. También separa los números adyacentes con espacios. Siempre separa cada número con dos caracteres. Al final de las líneas antes de un salto de línea no debe haber espacios. La salida debe terminar exactamente con un salto de línea continuo.

Ejercicio 6

Calcule la suma de los dígitos de un número entero no negativo.

Formato de entrada

Se ingresa un único número entero no negativo que no excede de

Formato de salida

Imprime la suma de los dígitos de este número.

Ejemplo

stdin	stdout
59	14

Ejercicio 7

El valor del logaritmo natural $ln2$ se representa como la suma

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\ldots+\frac{(-1)^{n+1}}{n}$$

Escribe un programa que muestre la suma de los primeros `n` términos de esta serie. Intenta también hacerlo sin utilizar la instrucción `if`.

Sugerencia: Use el tipo `double` para trabajar con números de punto flotante.

Formato de entrada

Se ingresa un número entero positivo `n`, que cabe en el tipo `int`.

Formato de salida

El programa imprime la respuesta al problema.

Ejemplo 1

stdin	stdout
3	0.833333

Ejemplo 2

stdin	stdout
2	0.5

Ejemplo 3

stdin	stdout
1	1

Ejercicio 8

Dada una string de letras minúsculas del alfabeto latino y espacios. Compruebe si es un palíndromo sin tener en cuenta los espacios.

Formato de entrada

Se ingresa una sola string. La string puede contener espacios. Puede haber un número arbitrario de espacios consecutivos. La longitud de la string no supera los 100 caracteres.

Formato de salida

Imagina que se eliminan todos los espacios de la cadena. Debes imprimir `YES` si la cadena resultante es un palíndromo, y `NO` en caso contrario.

Ejemplo 1

stdin	stdout
hello world	NO

Ejemplo 2

stdin	stdout
never odd or even	YES

Una cadena vacía se considera un palíndromo.

Ejercicio 9

Dada una cadena. Encuentra la segunda aparición de la letra f en esta cadena e imprime el índice de esta aparición. Si la letra f aparece solo una vez en esta cadena, imprime el número -1, y si no aparece en absoluto, imprime el número -2. Los índices se numeran desde 0.

Ejemplo

stdin	stdout
comfort	-1
coffe	3
car	-2

Ejercicio 10

Se proporcionan dos números naturales en stdin. Imprime en stdout su máximo común divisor.

Ejemplo

stdin	stdout
25 27	1
12 16	4
13 13	13

Ejercicio 11

Escriba un programa que calcule los primeros 10 términos de la secuencia Un tal que:

$$U_0 = 1; U_{n+1} = \frac{U_n}{n+1}$$

Salida

```
U0 = 1
U1 = 1
U2 = 0.5
U3 = 0.166667
U4 = 0.0416667
U5 = 0.00833333
U6 = 0.00138889
U7 = 0.000198413
U8 = 2.48016e-05
U9 = 2.75573e-06
U10 = 2.75573e-07
```

Ejercicio 12

Modifique el programa de arriba para que calcule simultáneamente la secuencia Un y la serie Vn, donde

$$V_n = \sum_{i=0}^{n} U_i$$

Verifique que Vm converga a e = exp(1) = 2.71828

Salida

```
U1 = 1 V1 = 1
U2 = 0.5 V2 = 1
U3 = 0.166667 V3 = 1
U4 = 0.0416667 V4 = 1
U5 = 0.00833333 V5 = 1
U6 = 0.00138889 V6 = 1
U7 = 0.000198413 V7 = 1
U8 = 2.48016e-05 V8 = 1
U9 = 2.75573e-06 V9 = 1
U10 = 2.75573e-07 V10 = 1
```

Ejercicio 13

Encuentra la suma algebraica de la expresión: $1^k + 2^k + 3^k + \ldots + N^k$. El usuario introduce N y la potencia k.

Ejemplo

N = 5
k = 2
Resultado = 55

Ejercicio 14

Se te da un número entero. Tu tarea es encontrar si el número dado es un palíndromo o no.

Ejemplo

stdin	stdout
212	Es palindrome

Ejercicio 15

Se te proporciona un número decimal. Tu tarea es convertirlo en un número binario. Para convertir el número decimal en binario, debemos seguir dividiendo el número por 2 hasta que el cociente sea igual a 0. El resto resultante será el número en binario.

Ejemplo

stdin	stdout
10	1010

Ejercicio 16

Encontrar el cociente de la división y el residuo utilizando solo operaciones de suma y resta.

Ejemplo

stdin	stdout
15 5	3 0

Ejercicio 17

Mostrar los primeros `n` numeros primos.
Ejemplo
n = 4
Mostrar: 2 3 5 7

Ejercicio 18 - 19

Supongamos que jugamos el siguiente "juego" con algún (positivo) entero:

- si puede ser dividido por 3, agreguemos 4 a él;
- si no puede ser dividido por 3 pero puede ser dividido por 4, dividámoslo por 2;
- de lo contrario, restémosle 1.

Repetimos las operaciones anteriores hasta llegar a 0. Concretamente, comenzando con un entero `n0` y aplicando las reglas una vez, obtenemos un nuevo número `n1`. Luego, si `n1` no es nulo, aplicamos las reglas anteriores a él para obtener un nuevo entero `n2`; y así sucesivamente hasta obtener un entero `nk` que sea nulo.

Por ejemplo, si comenzamos desde 7, entonces obtenemos: 6, 10, 9, 13, 12, 16, 8, 4, 2, 1 y 0. El número k de repeticiones es así 11 en este caso.

Como otro ejemplo, si comenzamos desde 1, obtenemos directamente 0, y el número de repeticiones es así k = 1.

Ahora, se te pide que escribas un programa que, para cada entero entre dos enteros solicitados al usuario, imprima el número de iteraciones requeridas para obtener 0 a partir de él.

Por ejemplo, si el usuario solicita ver los resultados entre 1 y 7, el programa mostrará:

```
1 → 1
2 → 2
```

```
3 → 12
4 → 3
5 → 4
6 → 10
7 → 11
```

En el ejemplo anterior, 1 y 7 (en la primera columna) son los límites proporcionados como entrada por el usuario. La segunda columna muestra el número de repeticiones para llegar a 0 con el juego descrito anteriormente, comenzando con el número en la primera columna; por ejemplo, 11 repeticiones comenzando desde 7. Como otro ejemplo, si el usuario solicita enteros entre 99 y 100, el programa mostrará:

```
99 → 18
100 → 17
```

Considere que el valor inicial no puede ser negativo. En este caso mostrar el mensaje:

```
El numero debe de ser positivo y mayor a zero
```

Ademas, el valor final no debe ser menor al valor inicial.

```
El numero de fin no debe de ser menor al inicial
```