Question - 1

SCORE: **40 points**

Imprimir la tabla de contenidos (versión 1)

C++ Standard Input | C++ Standard Output

Como parte de un software de conversión de documentos de un formato a otro, se requiere una rutina que formatee la tabla de contenidos para archivos de texto. La rutina recibe unos parámetros en la primera línea y en las subsecuentes los títulos de las secciones.

Ejemplo de entrada:

```
50 3 .

Programacion procedimental    5
Entrada y salida en C    5
Expresiones y condicionales    11
Ciclos    13
Subrutinas    15
Punteros, arreglos y matrices    23
Cadenas de caracteres de C    51
Registros (estructuras)    78
Programacion orientada a objetos    89
Clases, objetos, atributos, metodos    90
Sobrecarga de operadores (clase String)    107
Programacion generica    124
Arreglo dinamico    124
Lista enlazada    138
Arbol binario    155
Herencia y polimorfismo    192
```

La primera línea indica el ancho total de la página medida en caracteres, seguida del ancho para los números de página, y del carácter de relleno a usar entre los títulos de las secciones y el número de página. La primera línea del ejemplo de entrada indica que se quiere formatear la tabla de contenidos a 50 caracteres de ancho. De esos 50 caracteres, 3 serán usados para los números de página. El espacio sobrante entre el título y los números de página se rellenará con el carácter punto (`'.'`).

Cada una de las subsecuentes líneas contiene un título del documento, y el número de página donde inicia la sección, ambos separados por un tabulador.

Ejemplo de salida:

```
Programacion procedimental.....................
5
Entrada y salida en C.........................
5
Expresiones y condicionales...................
11
Ciclos........................................
13
Subrutinas....................................
15
```

```
Punteros, arreglos y matrices.................
23
Cadenas de caracteres de C...................
51
Registros (estructuras)......................
78
Programacion orientada a objetos.............
89
Clases, objetos, atributos, metodos..........
90
Sobrecarga de operadores (clase String).......
107
Programacion generica........................
124
Arreglo dinamico.............................
124
Lista enlazada...............................
138
Arbol binario................................
155
Herencia y polimorfismo......................
192
```

Se sabe que la longitud de un título de sección nunca supera los 1024 caracteres. Su solución deberá implementar una clase `TableOfContents`, el método `readAndFormat()`, y otros métodos auxiliares.

Para la programación en C++: Puede usar el método `cin.getline()` para leer los títulos de las secciones. Debe obligatoriamente usar entrada y salida de C++ y no de C. Establezca campos y caracteres de relleno al imprimir los títulos de las secciones.

Para la programación en Java: Puede leer los títulos y los números de página fácilmente si establece como delimitador permitido de entrada únicamente a tabuladores y cambios de línea. Java no permite especificar el carácter de relleno de campos, pero se puede lograr fácilmente usando el método `format()` de la clase `String`. Por ejemplo para generar un campo de ancho 10 lleno de puntos, se crea un campo de 10 espacios en blanco y luego se reemplazan esos espacios por puntos:

```
String.format("%10c", ' ').replace(' ', '.')
```

Question - 2

SCORE: **48 points**

Vectores en el plano cartesiano

Class | Instance | Operator Overloading

Sus amigos de secundaria quieren que les ayude con un poco de física. Ellos tienen parcialmente implementado, con algunos errores, un programa para realizar operaciones con vectores en el plano cartesiano. Por favor, ayude a corregirlo y completarlo.

Implemente una clase `Vector` para representar cantidades vectoriales físicas (como la velocidad, la fuerza y el campo eléctrico) en el plano

cartesiano. Un vector consta de magnitud y dirección, y hay varias convenciones para representarlas. Su clase `Vector` debe representar los vectores en el plano cartesiano, con origen siempre en el punto *(0, 0)* y como punto final las coordenadas *(x, y)*, como se aprecia en la Figura 1.

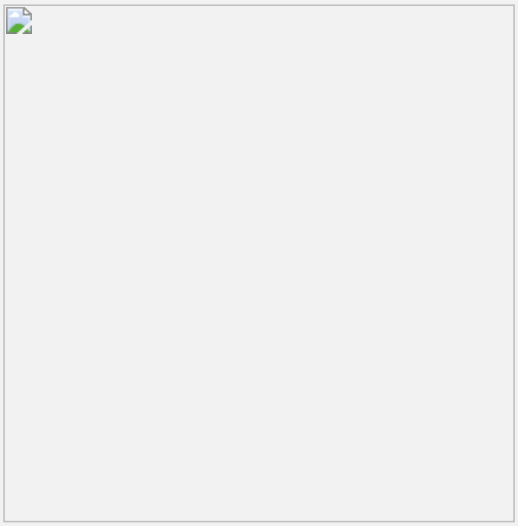

Figura 1. Dos vectores en el plano (en azul), el vector resultante de sumarlos (en lila) y restarlos (en rojo y verde)

Su clase `Vector` debe ser capaz de realizar lo siguiente

1. Almacenar los componentes *(x, y)*, los cuales pueden ser distintos de un vector a otro. Inicializarlos por defecto como el vector nulo *(0, 0)*.

2. Leer un vector de la entrada estándar como una pareja de valores separados por un espacio que representan coordenadas cartesianas. Su método de lectura puede retornar `true` cuando se ha leído un vector distinto al vector nulo.

3. Calcular la norma *r* o magnitud $\left|\vec{V}\right|$ utilizando el Teorema de Pitágoras.

4. Calcular la dirección como una coordenada angular θ en radianes. Sugerencia utilice el método `Math.atan2(y,x)` que retorna θ.

5. Imprimir el vector en la salida estándar como coordenadas cartesianas *(x, y)*.

6. Imprimir el vector en la salida estándar como coordenadas polares *(r, θ)*.

7. Sumar y restar vectores. La suma o la resta de dos vectores $\vec{V_1} = (x_1, y_1)$ y $\vec{V_2} = (x_2, y_2)$ es el vector resultante de sumar o restar sus componentes:

$$\vec{V_1} + \vec{V_2} = (x_1 + x_2, y_1 + y_2)$$
$$\vec{V_1} - \vec{V_2} = (x_1 - x_2, y_1 - y_2)$$

Sus amigos de secundaria quieren indicar en la primera línea, si se quiere imprimir en coordenadas cartesianas (`cartesian`) o polares (`polar`). Luego el programa lee dos vectores en el plano. Por sencillez, los vectores siempre se leen en coordenadas cartesianas. Luego de leer

los vectores, se debe mostrar su magnitud y dirección de acuerdo al sistema de coordenadas indicado en la primera línea.

Finalmente el programa calcula e imprime la resultante de sumarlos y restar ambos vectores como se muestra en el ejemplo de ejecución. Dado que sus amigos quieren usar el programa para revisar ejercicios de libro, haga que siempre que el programa imprima un número entero, se haga sin decimales; y si no es entero, con dos decimales.

Ejemplo de entrada:

```
cartesian

2 5
4 -3
```

Ejemplo de salida:

```
V1 = (2, 5)
V2 = (4, -3)

V1 + V2 = (6, 2)
V1 - V2 = (-2, 8)
V2 - V1 = (2, -8)
```

Question - 3

SCORE: **52 points**

Taquígrafa de tribunal

String | Class | Instance

Su amiga Rosemary ha trabajado durante años como taquígrafa en un tribunal. Por un accidente, tuvo que incapacitarse temporalmente. Al recuperarse y regresar al trabajo, encontró que la van a despedir... su asistente le serruchó el piso. Rosemary está determinada a recuperar su trabajo y necesita su ayuda. Ella quiere probar a su jueza que ella es mucho mejor taquígrafa que su asistente Sharon. Para probarlo, ella retó a su asistente a un duelo, quien no quiso inicialmente, pero la jueza Gina aceptó.

El duelo consiste en lo siguiente. La jueza Gina va a leer un texto de su propia elección a toda velocidad. Ambas taquígrafas deben transcribir el texto en sus estenógrafos sin interrumpir a la jueza. Para que la evaluación sea justa, se piensa en un programa de computadora que pueda contar la cantidad de errores de escritura que cometan Rosemary y Sharon. Los errores se deben contar como la cantidad de cambios que deben hacerse en las transcripciones para que se equiparen al texto original escogido por Gina.

El programa debe recibir en la entrada estándar tres textos. El primero de ellos es el texto literal que leyó la jueza. Después de un separador, denotado por 10 símbolos de igual (`==========`), vendrá el texto que Rosemary logró transcribir. Después de otro separador de 10 símbolos de igual, vendrá como tercer texto, el que Sharon transcribió.

Ejemplo de entrada:

```
Lo vieron salir de casa en el escarabajo rojo.
Apresuradamente, según el
informe. Cruzó la ciudad a toda prisa, llegó al
aeropuerto, embarcó en un
vuelo nacional en el último momento y dejó a sus
perseguidores con un palmo
de narices. El coche y el aparcamiento del
aeropuerto fueron sometidos a una
vigilancia intensiva.

El avión despegó rumbo a Sao Paulo, y debía hacer
cuatro escalas intermedias.

==========

Los vieron salir de casa en el escarsbajo rojo,
apresuradamente, según el
informe. Cruzó la ciudad a toda prisa, llegó al
aeropeurto, embarcó en un
vuelo nacional en el último momemento y dejó a
sus pesreguidores con un palmo
de narices. El coche y el aparcamiento del
aeropuerto fueron sometidos a una
viglincia intensiva.

El avión despegó rumbo a Sao Paulo, y debía hacer
cuatro escalas intremedias.

==========

Lo vieron salir de casa en el escarabajo rojo
apresruadasmente según el
informe. Cruzo la ciudad a toda pirsa. LLegó al
aeropuerto. Embrarco en un
vuelo nacional en el ultimo momento y dejo a sus
perceguidrores con una palma
de narises. El coche y el apartamento del
aeropuerto fueron sometidos a una
vigilansia intenciva.

El avión despego rumbo a San Pablo, y debía haser
cuatro ecscalas intermedias.
```

Ejemplo de salida:

```
Original:    67    333
TranscrA:    14    4.20%
TranscrB:    32    9.61%
```

El programa deberá comparar primero el texto de Rosemary contra el texto original e indicar el número de errores. Luego hacer lo mismo con el texto de Sharon. Los textos se deben comparar palabra por palabra, una a a la vez. Las palabras se separan por espacios en blanco. Usted le explica a Rosemary que el programa podría ser útil para otras situaciones. Por eso acuerdan que en la salida, el programa simplemente indique lo siguiente:

1. La cantidad de palabras y caracteres en el texto original (identificado como `Original`), sin considerar espacios en blanco. Ambas cantidades se separan por tabuladores.
2. La cantidad de correcciones que hay que hacer en el texto de Rosemary (identificado como `TranscrA`) para que se equipare al

texto original, y un porcentaje, calculado como esa cantidad de cambios entre la cantidad de caracteres del texto original.
3. La cantidad de correcciones en el texto de Sharon, identificado como `TranscrB` y el porcentaje de errores que representa. Se calcula igual que en el punto anterior.

Para implementar este programa a Rosemary, usted indaga y encuentra que el algoritmo de Distancia de Levenshtein resulta muy apropiado para calcular la cantidad de modificaciones que hay que realizar en una palabra para que se convierta en otra. También usted encuentra que existen muchas implementaciones reutilizables de algoritmo y decide aprovechar una, por ejemplo, la realizada por el equipo de Apache.

[Opcional: si quiere pasar los últimos cuatro casos de prueba] Usted encuentra una dificultad adicional. Ocurre que es normal que los taquígrafos omitan palabras. Para que los resultados de su programa no se vean afectados por este hecho, usted propone adaptar el programa para comparar cada palabra transcrita contra la correspondiente del texto original y adicionalmente contra la siguiente en el texto original. Si la cantidad de cambios es mayor en la palabra correspondiente que con la palabra siguiente, el programa asume que la taquígrafa la omitió, y toma la longitud de la palabra original como el número de errores para esa palabra.

Question - 4

El bosque encantado

SCORE: **65 points**

Un rey generoso de una isla muy lejana, estaba preocupado porque sus ciudadanos estaban aburridos de la monotonía de la pequeña isla. Tuvo la idea de encantar el bosque que está alrededor de su castillo, de tal forma que cada medianoche el bosque cambie y al día siguiente los ciudadanos encuentren un lugar diferente para recrearse. A su mago le pareció genial la idea, pero no sabe qué reglas incluir en el hechizo para que el bosque se mantenga en equilibrio y no llegue a convertirse en un desierto o una selva impenetrable. Si tuvieran alguna forma de ver el efecto de las reglas a futuro, podrían decidir el hechizo con mayor seguridad...

El rey tiene un mapa del bosque como el que puede verse en el ejemplo de entrada. El mapa ilustra lo que hay en cada metro cuadrado del bosque, a los que les llaman *celdas*. Una celda puede contener un árbol mágico (`'a'`), un trozo de lago encantado (`'l'`), o pradera donde pueden transitar los ciudadanos (`'-'`). Las dimensiones del mapa se encuentran en la primera línea de entrada, indicadas como el número de filas y columnas. Un tercer número indica la cantidad de medianoches en las que quiere probarse el efecto de las reglas para decidir el hechizo. Las reglas que quieren probarse en cada medianoche trabajan en una celda a la vez y consideran las 8 celdas vecinas que tiene alrededor. Son las siguientes.

1. *Inundación*: Si la celda tiene un árbol y al menos 4 vecinos que son lago encantado, entonces el lago ahoga el árbol, y pasa a ser lago encantado.
2. *Sequía*: Si la celda es lago encantado y tiene menos de 3 vecinos que sean lago encantado, entonces el lago se seca y pasa a ser pradera.
3. *Reforestación*: Si la celda es pradera y tiene al menos 3 vecinos árboles, las semillas tendrán espacio para crecer y la celda se convierte en árbol.

4. *Hacinamiento*: Si la celda es un árbol y tiene más de 4 vecinos árbol, el exceso de sombra evita que crezca y entonces pasa a ser pradera.
5. *Estabilidad*: Cualquier otra situación, la celda permanece como está.

Se quiere que el programa que ayude a probar las reglas, imprima el bosque en su estado actual (día 0) y en cada medianoche aplique las reglas anteriores. El programa debe imprimir el estado del bosque en cada amanecer. En el ejemplo siguiente, se solicitó aplicar las reglas en una medianoche, por tanto el programa imprime el día inicial (antecedido por la leyenda `0:`) y el día siguiente a la medianoche (precedido por `1:`).

Ejemplo de entrada:

```
7 7 1

-------
-l--l--
-ll----
-l-----
---laa-
-aa-al-
a-a----
```

Ejemplo de salida:

```
0:
-------
-l--l--
-ll----
-l-----
---laa-
-aa-al-
a-a----

1:
-------
-------
-ll----
-------
----aa-
-aaaa--
aaaa---
```

Dado que el rey y el mago quieren ver el efecto de las reglas a un futuro lejano, una salida de día tras día puede ser abrumadora. Por esto, si el número de medianoches se especifica en negativo, por ejemplo `-1000`, el programa debe imprimir únicamente el estado inicial (día `0:`) y el día final (día `1000:`).

Recuerde diseñar primero una solución antes de implementar. Para su solución en C implemente matrices en memoria dinámica y pase estas por parámetro entre varias subrutinas. Para su solución en Java debe implementar al menos dos clases, una controladora (por ejemplo, `Solution`) y una clase modelo que representa el bosque (`Forest`). El bosque debe realizar al menos las siguientes operaciones:

1. Construir un bosque vacío con dimensiones de filas y columnas arbitrarias dadas por parámetro.
2. Saber la cantidad de filas y columnas que hay en el bosque.

3. Actualizar una celda del bosque en una medianoche. Debe implementarse con un método privado que recibe por parámetros la fila y la columna que se quiere actualizar. El método actualiza la celda en medianoche de acuerdo a las reglas.
4. Actualizar todo el bosque en una medianoche.
5. Este bosque no se imprime en la salida estándar, sino que de acuerdo al las convenciones del lenguaje de hechizos, Java, la clase implementa un método `toString()`. Este método construye y retorna un objeto de tipo String que contiene el estado del bosque dispuesto en filas y columnas. Este resultado puede ser impreso en la salida estándar o usado para otros propósitos.

Sugerencia. Para realizar las actualizaciones utilice dos matrices, una que representa el estado actual del bosque hoy, y otra que tendrá el estado del bosque al día siguiente (después de la medianoche).