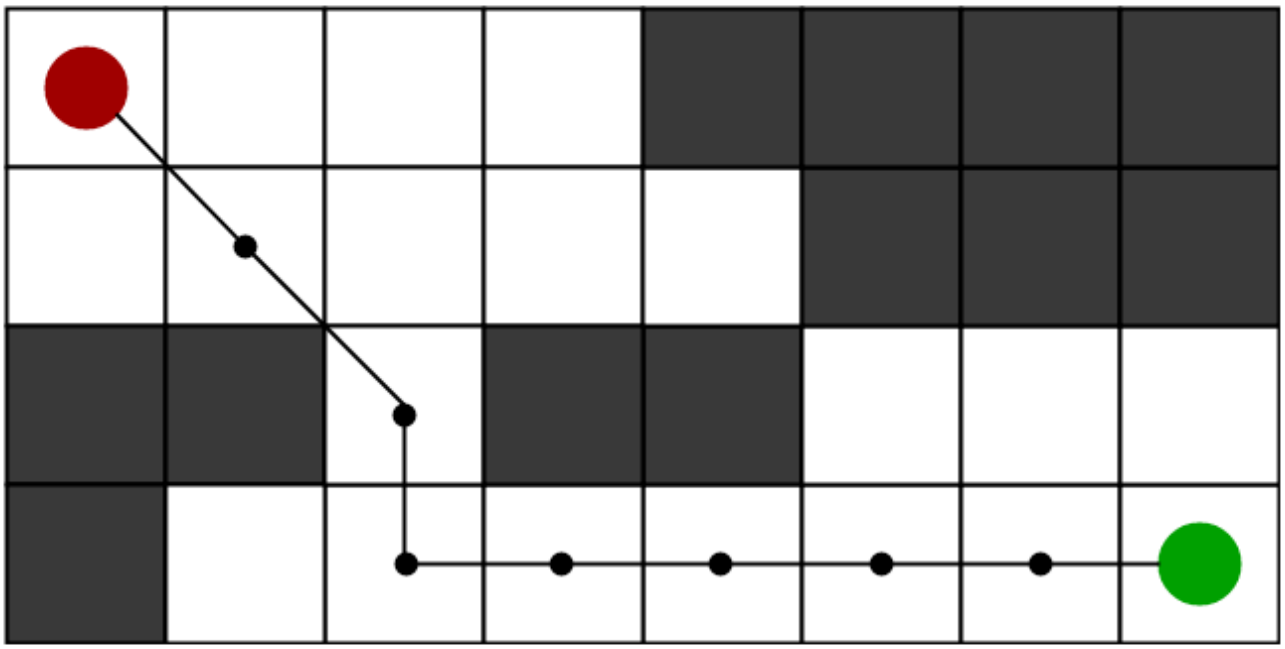
Exercise 1



1. En la funcion main del programa declare una variable, `board`, de tipo vector de vectores de enteros.
2. Asigne la informacion del mapa de arriba. En donde, las casillas blancas deben tener valores de 0, las casillas negras valores de -1, el punto rojo una `x` y el punto verde una `G`. Por ejemplo:

```
{{0, 1, 0, 0, 0, G},
 {x, 1, 0, 0, 0, 0},
 {0, 1, 0, 0, 0, 0},
 {0, 1, 0, 0, 0, 0},
 {0, 0, 0, 0, 1, 0}}
```

Note: No se olvide de incluir la libreria `vector` en su programa

3. Ahora que tiene un `board` almacenado en su programa, necesitará una forma de imprimirlo para poder mostrar los resultados. En este ejercicio, agregará una función `printBoard` para imprimir el tablero una fila a la vez. Para este ejercicio debe utilizar los indices del vector. Cuando haya terminado, la salida impresa debería verse así:

```
010000
010000
010000
010000
000010
```

4. Para el `vector<vector<int>>` `board` definido anteriormente, intente obtener el tamaño de uno de los vectores internos. Por ejemplo, para el vector creado abajo el tamano de uno de sus vectores internos debería ser 4.

```
vector<vector<int>> b = {{1, 1, 2, 3},
                         {2, 1, 2, 3},
                         {3, 1, 2, 3}};
```

5. Ahora intente escribir un bucle doble range-based que imprima todas las entradas del vector 2D `board`.

Exercise 2

El usuario ingresa 40 números enteros. Póngalos en una vector. Luego, imprima el vector en el orden en que se ingresaron los números. Despues, imprima en orden inverso. Cada impresión del vector debe separar los números del vector por un espacio.

Por ejemplo: la matriz era `[3,4,55]` la impresión sería `3 4 55`

Exercise 3

Hay un vector de números enteros. La longitud de los vectores que se van a buscar varía. Un usuario ingresará un número entero y el programa buscará en el vector. Si el valor está en el vector, el programa debe devolver la ubicación del elemento. Si el valor no está en el vector, se notificará al usuario que el valor no está en el vector. Para salir del programa el usuario ingresará `-1`.

Exercise 4

Escriba un programa que acepte valores para una matriz de 4x4 y un vector de tamaño 4. Use el producto escalar para multiplicar la matriz por el vector. Imprima el vector resultante.

Exercise 5

Se dan los valores de temperatura observados durante `N` días sucesivos. Encuentre el número de días (numerados desde cero) con un valor de temperatura por encima de la media aritmética para todos los `N` días.

Se garantiza que la media aritmética de los valores de temperatura es un número entero.

Formato de entradas

Se ingresan el número *N,* luego *N* enteros no negativos: los valores de temperatura en el día 0, 1, ... (*N*−1)-ésimo.

Formato de salidas

El primer número *K* es el número de días en los que el valor de la temperatura está por encima de la media aritmética. Entonces *K* enteros son los números de estos días.

Ejemplo

Entrada

```
5
7 6 3 0 9
```

Salida

```
3
0 1 4
```

Exercise 7

La gente esta parada en una fila, pero nunca se van del principio, sin embargo pueden llegar al final e irse de alli. Además, a veces algunas personas pueden detenerse y comenzar a preocuparse por el hecho de que la cola no avanza.

Asumiremos que las personas en la cola están numeradas por números enteros, comenzando desde 0.

Implemente el procesamiento de las siguientes operaciones en la cola:

- **WORRY** ***i***: marcar a la ***i***ésima persona del frente de la cola como preocupada;
- **QUIET** ***i***: marcar a *i*-ésima persona como calmada;

- **COME *k***: agregar ***k*** personas tranquilas al final de la cola;
- **COME -*k***: eliminar ***k*** personas del final de la cola;
- **WORRY_COUNT**: averiguar el número de personas preocupadas en la cola.
- **La cola está inicialmente vacía.**

Formato de entrada

Número de operaciones `Q`, luego descripciones de las operaciones.

Para cada operación de **WORRY *i*** y **QUIET *i***, se garantiza que existe una persona con el número ***i*** en la cola al momento de ejecutar esta operación.

Para cada operación **COME -*k***, se garantiza que **_k** no es mayor que el tamaño de la cola actual.

Formato de salida

Para cada operación **WORRY_COUNT**, imprime un número entero: el número de personas preocupadas en la cola.

Entrada

```
8
COME 5
WORRY 1
WORRY 4
COME -2
WORRY_COUNT
COME 3
WORRY 3
WORRY_COUNT
```

Salida

```
1
2
```

Hint

Puede usar la funcion count para contar la contidad de `1` en un vector:

```
count(begin(is_nervous), end(is_nervous), true)
```

Exercise 8

Se le proporciona un vector de enteros únicos `salary` donde `salary[i]` es el salario del i-ésimo empleado.

Cree una funcion que acepte este vector sin crear uan copia y esta funcion debe devolver el salario promedio de los empleados excluyendo el salario mínimo y máximo.
Considere que los enteros del vector `salary` son unicos (No se repiten)

Ejemplo 1:
Entrada: `salary = [4000,3000,1000,2000]`
Salida: 2500.00000
Explicacion: El salario mínimo y el salario máximo son 1000 y 4000 respectivamente.
El salario promedio excluyendo el salario mínimo y máximo es (2000+3000) / 2 = 2500

Ejemplo 2:
Entrada: `salary = [1000,2000,3000]`
Salida: 2000.00000
Explicacion: El salario mínimo y el salario máximo son 1000 y 3000 respectivamente.
El salario promedio excluyendo el salario mínimo y máximo es (2000) / 1 = 2000

Exercise 9

Una escuela está tratando de tomar una foto anual de todos los estudiantes. Se les pide a los estudiantes que se paren en una sola fila en orden no decreciente por altura. Supogamos que este orden es representado por un vector de enteros esperado donde `expected[i]` es la altura esperada del i-ésimo estudiante en la fila.

Se le proporciona un vector de enteros `heights` que representa el orden actual en el que se encuentran los estudiantes. Cada `heights[i]` es la altura del i-ésimo estudiante en la fila (índice 0).

Devuelve el número de índices donde `heights[i] != expected[i]`.

La funcion debe ser:

```
int heightChecker(vector<int>& heights);
```

Ejemplo 1:
Entrada: heights = `[1,1,4,2,1,3]`

Salida: 3
Explicacion:

```
heights:  [1,1,4,2,1,3]
expected: [1,1,1,2,3,4]
Los indices 2, 4, y 5 no corresponden.
```

Ejemplo 2:
Entrada: heights = `[5,1,2,3,4]`
Salida: 5
Explicacion:

```
heights:  [5,1,2,3,4]
expected: [1,2,3,4,5]
Todos los indices estan fuera de lugar.
```

Ejemplo 3:
Entrada: heights = `[1,2,3,4,5]`
Salida: 0
Explicacion:

```
heights:  [1,2,3,4,5]
expected: [1,2,3,4,5]
Todos los indices esta en su lugar
```