UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE COMPUTACIÓN
ALGORITMOS Y PROGRAMACIÓN

Proyecto #2 – Star Wars

Luego de que una división de la Resistencia fuese emboscada por una nave Star Destroyer de la Primera Orden, Poe Dameron y Finn tienen que ir a buscar sobrevivientes lo antes posible. Para evitar más perdidas, necesitan de los alumnos de Algoritmo y Programación para crear una simulación que los ayude a evadir a la nave de la Primera Orden y a rescatar a los sobrevivientes del ataque.

El mapa consiste de un espacio de MxN posiciones, cuyos posibles caracteres son los siguientes:

.	Posición Libre
F	Unidad de Combustible
S	Unidad de Escudo
R	Nave de la Resistencia
@	Sobreviviente
X	Star Destroyer
#	Basura Espacial

La simulación consta de varias acciones que se deben realizar según la posición actual de las naves. Esta simulación consta de **I** iteraciones (ingresado por el usuario), donde en cada iteración se mueven las naves una vez, primero la nave de la Resistencia y luego el Star Destroyer.

La nave de la Resistencia cuenta con 4 unidades de **combustible** inicialmente, por cada vez que se mueva gasta una unidad de combustible, pero puede recargar combustible agarrando las unidades '**F**' que encuentre por el mapa. Si la nave tiene 0 unidades de combustible, termina la simulación.

Si la nave de la Resistencia se posiciona sobre un **Sobreviviente** '**@**' lo rescata.

La nave de la Resistencia también empieza la simulación con sus **escudos** al 100%, pero en caso de chocar con basura espacial, se reducen un 25% y se destruye la basura espacial, si la nave recibe un impacto directo del Star Destroyer se reducen en 50%, pero pueden restaurar sus escudos en 10% agarran una unidad '**S**'. Si la nave de la Resistencia recibe un

impacto con sus escudos al 0% la simulación termina. Si la nave tiene sus escudos al 10% y recibe un ataque del Star Destroyer o choca con basura espacial, sus escudos quedan en 0%.

Como el **Star Destroyer** es un crucero de batalla, lo que significa que es considerablemente grande, será representado en la simulación con un tamaño de 2x2 posiciones.

Su rango de ataque depende del estado en el que se encuentre. Si se encuentra en modo de reconocimiento su rango corresponde a las filas y las columnas que ocupe y tiene la capacidad de atacar a la nave de la Resistencia si se detiene en el (solo puede haber un ataque por iteración y el rango puede ser interrumpido por basura espacial), si la nave de la Resistencia es alcanzada por un ataque en este modo, sus escudos se reducirán un 50%, es decir:

.	.	.	.	.	.
.	.	X	X	.	.
.	F	X	X	.	.
.	.	.	.	.	.
.	.	S	#	.	.
.	.	.	R	.	.

Su segundo modo, es el modo de movimiento, durante el mismo el Star destroyer apunta únicamente a la dirección en la que se mueve, y en caso de acertar un golpe contra la nave de la Resistencia, sus escudos bajan completamente (si la nave de la Resistencia tiene sus escudos al 0% en ese momento es destruida). Es decir, si se le asigna la trayectoria 2 al Star Destroyer, su rango será:

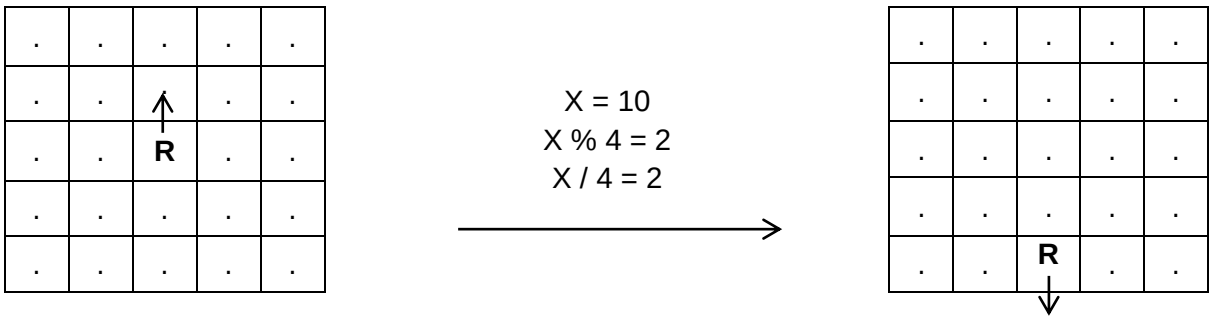
.	.	.	.	.	.
.	.	X	X	.	.
.	F	X	X	.	.
.	.	.	.	.	.
.	.	S	#	.	.
.	.	.	R	.	.

El Star Destroyer se encuentra en modo de reconocimiento en las iteraciones pares y en modo de movimiento en las iteraciones impares. Por lo tanto, solo se moverá en las iteraciones impares.

Si el Star Destroyer choca con una unidad de escudo, combustible, basura espacial o algún sobreviviente, lo destruye, y si se da el caso donde el Star Destroyer se ubique encima

de la nave de la Resistencia, la captura y se finaliza la simulación. Así mismo, se sabe que solo hay un único Star Destroyer cerca, pero puede darse el caso de que se haya ido. Al principio de la simulación se le asignará al Star Destroyer una dirección (según el Sistema de Trayectoria de la Resistencia) y en cada iteración la nave avanzará una posición en esa dirección. En caso de llegar al tope del mapa el Star Destroyer saldrá de la simulación y se asume que se retiró por completo.

La Resistencia desarrolló un **Sistema de Trayectoria** donde, dado un entero **X**, la operación **X** *mod* 4 indica el número de rotaciones de 45° en sentido horario (al principio de todas las simulaciones se asume que se está "viendo hacia arriba", luego las rotaciones se harán desde la última dirección indicada). Y la parte entera de la operación **X** / 4 es el número de posiciones que la nave va a desplazarse con la dirección obtenida (en caso de que el número de desplazamientos haga que la nave salga de la matriz, la nave solo se moverá hasta el extremo de la matriz). Por ejemplo:



Entrada

- La primera línea tendrá dos enteros $4 \leq \mathbf{M} \leq 8$ y $4 \leq \mathbf{N} \leq 8$.
- La segunda línea corresponde a un entero $0 \leq \mathbf{D} \leq 3$ que indica la dirección en la que se moverá el Star Destroyer.
- Luego se introducen **M** líneas de **N** caracteres que corresponden al mapa de la simulación.
- La siguiente línea corresponde a un entero $1 \leq \mathbf{I} \leq 5$ que representa el número de iteraciones.
- Por último, **I** líneas con los enteros que indican los movimientos de la nave de la Resistencia

Salida

Se deberá imprimir cada evento que ocurra en el mapa, con el formato especificado a Continuación (sin las comillas):

- Cada cambio de Iteración: "Iteracion <numero_actual>"
- Si la nave logra moverse de forma exitosa: "La Resistencia logro moverse exitosamente"

- Si la Resistencia rescata a un sobreviviente: “<numero_actual_de_rescates> sobrevivientes rescatados”
- Si la Resistencia recolecta una unidad de combustible: “Total de Unidades de Combustible: <numero_de_unidades_de_combustible_actual>”
- Si la Resistencia recolecta una unidad de escudo: “Escudos al <porcentaje_actual_de_escudos>%”
- Si el escudo de la Resistencia baja: “Escudos al <porcentaje_actual_de_escudos>%, causa: <Impacto con basura espacial/Ataque enemigo>”
- Si la simulación se termina exitosamente (llegó al número **I** de iteraciones): “<numero_actual_de_rescates> sobrevivientes rescatados. Fin de la simulación”
- Si la simulación se detiene antes de terminar todas las iteraciones: “<numero_actual_de_rescates> sobrevivientes rescatados. Fin de la simulación por: <Nave de la Resistencia Capturada/Nave de la Resistencia Destruida/Falta de Combustible>.”
- Si alguna entrada no es válida: “NO VALIDO”

Casos De Prueba

4 4 1 XX#. XX.. .##. @R@. 2 5 10	Iteracion 1 La Resistencia logro moverse exitosamente 1 sobrevivientes rescatados Iteracion 2 La Resistencia logro moverse exitosamente 2 sobrevivientes rescatados 2 sobrevivientes rescatados. Fin de la simulación
4 4 1 XX#. .XX. ###. R@@.	NO VALIDO
4 5 3 R#@.S ....@ XX.#. XX..F 2 17 17	Iteracion 1 La Resistencia logro moverse exitosamente Escudos al 75%, causa: Impacto con basura espacial 1 sobrevivientes rescatados Escudos al 85% Iteracion 2 La Resistencia logro moverse exitosamente 2 sobrevivientes rescatados Total de Unidades de Combustible: 3 Escudos al 35%, causa: Ataque enemigo 2 sobrevivientes rescatados. Fin de la simulación

7 7 2 ..XXF.. S#XX..@ @#F.R#S ..##... ……. F….## @S.#@FS 4 19 19 27 21	Iteracion 1 La Resistencia logro moverse exitosamente Total de Unidades de Combustible: 4 Escudos al 75%, causa: Impacto con basura espacial 1 sobrevivientes rescatados Iteracion 2 Escudos al 25%, causa: Ataque enemigo La Resistencia logro moverse exitosamente Total de Unidades de Combustible: 4 2 sobrevivientes rescatados Iteracion 3 La Resistencia logro moverse exitosamente Escudos al 35% Escudos al 10%, causa: Impacto con basura espacial 3 sobrevivientes rescatados Total de Unidades de Combustible: 4 Escudos al 20% Iteracion 4 Escudos al 0%, causa: Impacto con basura espacial Escudos al 10% 4 sobrevivientes rescatados 4 sobrevivientes rescatados. Fin de la simulación

OBSERVACIONES

1. Puede ser realizado de forma individual o en parejas (alumnos de cualquier sección).
2. Se debe enviar un pdf con el análisis de la solución del problema y el algoritmo pseudo formal (usando la notación vista en teoría).
3. SÓLO se deben usar las herramientas algorítmicas vistas en clase.
4. Se deben realizar las validaciones respectivas en los datos de entrada, con el fin de asegurar el robusto funcionamiento del programa.
5. Las copias serán penalizadas con cero (0) puntos para TODOS los involucrados.
6. El proyecto debe ser implementado con C++.

7. Recuerde **respetar el formato de salida**, en caso contrario contará con una **penalización** de 10 puntos.

FORMATO, LUGAR Y HORA DE ENTREGA

- Tanto el pdf como el archivo cpp deben contener la sección, cédula y nombre del integrante. Adicionalmente ambos archivos deben ser colocados en un zip que indique sección, cédula y nombre (incluyendo ambos integrantes, si es en pareja). Por ejemplo: “C1-123456-PedroPerez.zip” o “C1-123456-PedroPerez_C2-123457_JoseMartinez.zip".

- Enviar la solución del proyecto al correo aypucv@gmail.com. Cualquier proyecto que pase de la fecha y hora de entrega **NO SERÁ CORREGIDO.**

- La entrega es hasta el día 3/2/2020 a las 11:59 PM (GMT -4).