Tarea integradora 2 Computación y Estructuras Discretas 1: Venganza Romana

Cliente	Profesores del curso de Computación y Estructuras Discretas 1
Usuario	Monitores del curso y profesores
Requerimientos funcionales	RF0: Interfaz gráfica de usuario RF1: Selección de implementación de grafo RF2: Reproducción de música del juego RF3: Algoritmo de ayuda al usuario : Camino mínimo RF4: Algoritmo de ayuda al usuario : Camino económico RF5: Cálculo de puntaje RF6: Consultar información de un territorio RF7: Atacar territorio RF8: Rendirse
Contexto del problema	"Venganza Romana" es un emocionante juego de estrategia basado en un grafo no dirigido con más de 50 nodos y aristas que representa una parte de la Europa medieval. El jugador comienza en la histórica ciudad de Sicilia con el objetivo final de conquistar la base vikinga. Para lograrlo, deberá conquistar diferentes reinos, ciudades y pueblos, enfrentándose a desafíos estratégicos en cada territorio. El mapa está organizado en niveles, con Sicilia como punto de partida. El jugador avanza a través de pueblos interconectados, cada uno con rutas que conducen a niveles superiores Cada territorio tiene un peso asociado, indicando la dificultad de conquista. El jugador debe tomar decisiones estratégicas para maximizar recursos y minimizar pérdidas de soldados.
Requerimientos no funcionales	RNF 0: El juego debe implementar grafos con lista de adyacencia y con matriz de adyacencia. RNF 1: Se deben usar como mínimo 2 algoritmos de grafos, pero, tener implementados a los 6 (DFS, BFS, Dijkstra, Floyd Warshall, Kruskal, Prim).

Identificador y nombre	*[RF0-* Interfaz gráfica de usuario*]*		
Resumen	El juego requerirá el diseño e implementación de una interfaz gráfica de usuario intuitiva y atractiva. Esta interfaz permitirá a los jugadores interactuar de manera efectiva con el juego, facilitando la comprensión de la información y las decisiones estratégicas a tomar.		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
Resultado o postcondición	El juego reacciona a lo que el usuario hace con la interfaz gráfica		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**

Identificador y nombre	*[RF1-* Selección de implementación de grafo*]*

Resumen	Para demostrar que ambas implementaciones de grafos están bien construidas, se permitirá al usuario seleccionar que tipo de implementación usará para el juego, ya sea matriz de adyacencia o lista de adyacencia		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
	typeGraph	Implementation	*ADJACENCY_MATRIX* *ADYACENCY_LIST*
Resultado o postcondición	Se inicia el juego con la implementación de grafo escogida		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**

Identificador y nombre	*[RF2-* Reproducción de música del juego*]*		
Resumen	Para mejorar la inmersión del usuario al mundo y a su contexto se reproducirá música medieval cuando se inicie el juego.		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**

Resultado o postcondición	La canción medieval se reproduce		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**
	medievalSong	.mp3	

Identificador y nombre	*[RF3-*Algoritmo de ayuda al usuario : Camino mínimo*]*		
Resumen	Una de las ayudas disponibles para el usuario es el camino mínimo para llegar hasta los vikingos, para esto se usará el algoritmo de Dijkstra		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
Resultado o postcondición	Se calcula la ruta más rápida para llegar hasta los vikingos y la muestra en pantalla		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**

	bestRute	ArrayList<Vertex>	

Identificador y nombre	[RF4: Algoritmo de ayuda al usuario : Camino económico]		
Resumen	El otro algoritmo para ayudar al usuario es el de caminos económicos, este muestra las 10 aristas con menor ponderado. Este cálculo se debe de hacer con el algoritmo de Kruskal		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
Resultado o postcondición	Calcula los caminos más cortos y los muestra en pantalla		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**
	lighterWays	ArrayList<Edge>	

Identificador y nombre	[RF5: Cálculo de puntaje]

Resumen	Al finalizar la partida, si ganó, se calcula el puntaje del usuario usando sus tropas restantes y cantidad de territorios conquistados. La fórmula es la siguiente: Pueblos conquistados * 50 + Tropas restantes * 5 Se le restan 100 puntos al usuario si usó Kruskal, 200 si usó Dijkstra y si usó ambos 250.		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
	amountCities	int	*mayor o igual a 0*
	amountTroops	int	*mayor o igual a 0*
Resultado o postcondición	Calcula el puntaje		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**
	points	int	

Identificador y nombre	*[*RF6: Consultar información de un territorio]
Resumen	El usuario debe poder consultar la información de un territorio, dándole la información del peso que conlleva conquistar ese territorio

Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
	city	City	
Resultado o postcondición	Muestra en pantalla la información del territorio		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**
	weight	int	*mayor a 0*

Identificador y nombre	*[*RF7: Atacar territorio]		
Resumen	El usuario debe poder atacar un territorio, conquistando y perdiendo tropas (el peso de la arista). Si el usuario se queda sin tropas, este debe perder, si conquista el territorio vikingo, el jugador gana		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**
	city	City	

Resultado o postcondición	Reduce el ejercito del jugador, conquista el territorio y evalúa si ganó, perdió o continúa el juego		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**
	winMessage	String	*FELICIDADES SOLDADO, HAS LOGRADO NUESTRA VENGANZA AL CONQUISTAR A LOS VIKINGOS DEL NORTE, ERES UN ORGULLO!!* *Tu puntaje final ha sido de: " + score + " puntos,* *por tus " + armyRome + " unidades restantes y " + villages + " pueblos conquistados.* *Pena de " + help+ " puntos por el uso de ayudas en el juego*
	loseMessage	String	*¡Mala decisión, Roma ha caído!*

Identificador y nombre	*[*RF8: Rendirse]		
Resumen	El juego debe ofrecer la opción de rendirse haciendo que pierda automáticamente		
Entradas	**Nombre de entrada**	**Tipo de dato**	**Valores válidos**

Resultado o postcondición	Muestra en pantalla la pantalla de derrota		
Salidas	**Nombre de salida**	**Tipo de dato**	**Formato**