Carreras: Licenciatura e Ingeniería en Sistemas			
Parcial 2 de: Estructuras de datos y Algoritmos 2			**Código de materia:**
Fecha: 5-12-19	**Id Examen:**	**Acta:**	**Hoja 1** de **1**

Ejercicio 1 (25 puntos)

Un amigo suyo consigue un libro de laberintos. Desea adicionarle al libro una sección con las soluciones de cada laberinto. Como no es muy bueno resolviendo los laberintos, le pide a usted que implemente un algoritmo que resuelva un laberinto del libro.

Los laberintos de este libro no son comunes:

- Se representa el laberinto como una matriz NxM con los siguientes caracteres:
  - S: se puede pasar
  - N: no se puede pasar
  - R: resta salud.
- Se puede mover en 4 direcciones: arriba, abajo, izquierda y derecha.
- Se cuenta con un valor H que indica la salud inicial del jugador y hay un conjunto de posiciones conocidas del laberinto (R) que restan 5 puntos de salud. Cuando la salud es 0, el jugador pierde.
- Se conoce el origen y el destino.
- El laberinto puede tener varios caminos de origen a destino, pero según las reglas, un camino es mejor a otro si es más corto y en caso de tener el mismo largo, si la salud restante es mayor.

a) Describa cómo resolvería este problema utilizando la técnica de diseño de algoritmos **greedy**.
b) Resuelva el problema, usando la técnica de diseño de algoritmos con retroceso (**backtracking**). Se debe retornar un conjunto (array/lista) de las mejores soluciones.

El desarrollo de esta función debe incluir claramente expresados los aspectos centrales de la técnica aplicada como, por ejemplo, estudio de la factibilidad de la solución, poda (si corresponde), construcción de la solución, consideración de las alternativas disponibles, retroceso.
Serán muy importantes en la evaluación aspectos como legibilidad, uso de nombres auto explicativos para variables, funciones, etc.
Se podrá usar función es auxiliares que se deberán implementar.

Para este ejercicio puede usar TADs que deberá especificar, pero no implementar.

Ejercicio 2 (20 puntos)

Dada la siguiente función:

```
int f(int n)
{
	if (n < 4)
	{
		return n;
	}
	else if (n % 2 == 0)
	{
		return f(n - 1) + f(n - 2) + f(n - 3) - f(n - 4);
	}
	else
	{
		return f(n - 2) + f(n - 3) - f(n - 4);
	}
}
```

a) Explique qué problemas tiene la implementación anterior.
b) Implemente utilizando la técnica **memorización**.
c) Implemente utilizando la técnica **programación dinámica**.