INF-253 Lenguajes de Programación
Tarea 2: C

Profesor: José Luis Martí – Esteban Daines
Ayudante Cátedras: Juan Pablo Escalona
Ayudante Tareas: Javier Echiburú – Cristian Vallejos

23 de Agosto de 2016

1. SansaStone™

La conocida empresa de videojuegos Stezzard ha desarrollado una nueva idea para un juego de cartas virtual ambientado en la Universidad Técnica Federico Santa María. Para llevar dicha idea a cabo, se le ha pedido a los alumnos de Lenguajes de Programación que diseñen una versión beta del juego.

Para aumentar la velocidad del proceso, Stezzard pidió a los alumnos que trabajaran con un solo tipo de carta, sin embargo que la estructura del mazo del juego permitiera en un futuro agregar nuevos tipos de cartas sin mayores cambios.

2. Reglas del juego

- El programa a implementar consiste en un juego de cartas virtual versión sansana.
- Cada jugador tendrá disponible su propio mazo, el cual contará con 20 cartas.
- Cada vez que se comience una nueva partida el mazo debe estar ordenado aleatoriamente.
- Los jugadores serán tipo Sansano.
- La prioridad es la vida de los Sansanos.
- Cada jugador tendrá 3000 puntos de prioridad inicialmente.
- La prioridad **máxima** es 3000.
- Gana el jugador que deje sin puntos de prioridad al rival.
- Cada inicio de turno se roba una carta del mazo.
- Las cartas se activan justo después de sacarlas del mazo.
- Cuando ambos mazos estén sin cartas el juego termina y gana el Sansano que posea mayor prioridad.
- Al comenzar, el jugador debe elegir un nombre, luego iniciará la partida.
- El enemigo será la computadora, sus jugadas deben ser de manera aleatoria.
- El mazo del jugador tiene las mismas cartas que el mazo rival, sin embargo el orden de éstas es aleatorio.

- El único tipo de carta existente es tipo Curso.
- Existen 2 tipos de jugadas para cada carta tipo Curso:
  - Reprobar: Usa su poder de ataque para quitarle prioridad al enemigo.
  - Aprobar: Usa su poder de defensa para auto curarse la prioridad.
- A continuación irán las cartas tipo Curso del juego con su descripción correspondiente, además tendrá la cantidad de veces que se encuentra dicha carta en el mazo:
  - Matemáticas: Ataca 550/Cura 200 puntos de prioridad. Cantidad: 1
  - Física: Ataca 450/Cura 150 puntos de prioridad. Cantidad: 4
  - LP: Ataca 510/Cura 180 puntos de prioridad. Cantidad: 2
  - Programación: Ataca 110/Cura 300 puntos de prioridad. Cantidad: 6
  - ED: Ataca 470/Cura 160 puntos de prioridad. Cantidad: 3
  - EDD: Ataca 430/Cura 120 puntos de prioridad. Cantidad: 4

3. Requerimientos

- Se debe implementar un programa que permita jugar el juego descrito en la sección anterior.
- El juego se jugará mediante linea de comando, se debe entregar feedback de lo que ocurre en la pelea al jugador mediante texto (incluyendo la prioridad actual de los jugadores). El jugador utilizará los **números** seguidos de un enter para seleccionar opciones.
- El jugador y la computadora serán tipo Sansano, cada uno con un mazo propio.
- Los Sansanos y Cartas deberán ser creados mediante las estructuras dadas en la Sección 4.
- Deben crearse 2 funciones, **usarReprobar** y **usarAprobar**, las cuales se asignarán a la instancia de la estructura CartaCurso mediante punteros a función.
- Se debe implementar una lista enlazada para la creación y el manejo de las cartas disponibles en el mazo de cada jugador. Dado que Stezzard pidió a los alumnos que el mazo de un jugador pudiera admitir futuros tipos de cartas, se debe utilizar un puntero a void hacia el contenido del nodo correspondiente.
- No esta permitido el uso de librerías externas para el manejo de listas.
- Se debe trabajar sobe sistema operativo CentOS presente en los computadores de la Universidad.
- El programa debe incluir un **MAKEFILE** para su compilado.
- Como la lista enlazada no es parte del programa en sí, debe estar en archivos separados, con su respectivo archivo .c y .h.
- Cada definición de estructuras y cabeceras de funciones deben estar en un archivo .h
- Cada función de elementos debe estar en archivos .c
- Se debe liberar la memoria al momento de finalizar el programa.

4. Estructuras Básicas

```
struct CartaCurso{
    char * nombre, * descripcion;
    int ataque, defensa;
    void (*reprobar)(void *, void *); //Los parametros son la carta y el enemigo
    void (*aprobar)(void *, void *); //Los parametros son la carta y el jugador
}

struct Sansano{
    char * nombre;
    int prioridad;
    void * mazo;
}
```

5. Archivos a Entregar

Resumiendo lo anterior, los archivos mínimos a entregar son:

- Sansano.c (funciones de configuración de Sansano).
- Sansano.h (Archivo con estructuras y prototipos de funciones de Sansano).
- CartaCurso.c (funciones de configuración de CartaCurso).
- CartaCurso.h (Archivo con estructuras y prototipos de funciones de CartaCurso).
- main.c (Archivo con programa principal).
- MAKEFILE (Archivo de compilación automática).
- README.txt (Archivo manual con instrucciones de uso del programa).
- lista.c (Archivo con funciones de lista enlazada).
- lista.h (Archivo con estructuras y prototipos de funciones de lista enlazada).

Los alumnos pueden crear más archivos si lo estiman necesario, mientras tengan los nombrados anteriormente entre los enviados.

6. Sobre Entrega

- La revisión se efectuará sobre el sistema operativo CentOS presente en los computadores del laboratorio de la Universidad.
- El código debe venir identado y sin warnings.
- Cada función debe llevar una descripción según lo establecido por el siguiente ejemplo:

```
/******** Funcion: Suma_Enteros ********************
Descripcion: suma dos enteros positivos

Parametros:
n1 entero
n2 entero

Retorno: resultado de la operacion aritmetica de la suma entero
************************************************/
```

- Debe estar presente el archivo MAKEFILE para que se efectúe la revisión.
- Se debe trabajar en grupos de dos personas, las cuales deben pertenecer al mismo curso de malla.
- **La entrega debe realizarse en tarball (tar.gz) y debe llevar el nombre: Tarea2LP_RolIntegrante-1_RolIntegrante-2.tar.gz**
- **El archivo README.txt debe contener nombre y rol de los integrantes del grupo e instrucciones detalladas para la compilación y utilización de su programa.**
- **El no cumplir con las reglas de entrega conllevará -30 puntos en su tarea.**
- La entrega será via moodle y el plazo máximo de entrega es hasta el **lunes 12 de septiembre a las 23:55 hora moodle**.
- Por cada día de atraso se descontarán 30 puntos.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.

7. Calificación

Todos comienzan con nota máxima y se les irá descontando por cada punto omitido (el puntaje que aparece al lado es el máximo de puntos que se les descontara en caso de incumplimiento):

- Código no ordenado (-5 puntos)
- Código no comentado (-5 puntos)
- Mal funcionamiento del juego (-20 puntos)
- No utiliza listas enlazadas (-20 puntos)
- No utiliza estructuras propuestas (-20 puntos)
- No libera toda la memoria utilizada (-20 puntos)
- Warnings (-5 puntos cada uno, con un máximo de -25 puntos)
- No Cumplir Reglas de Entrega (-30 puntos)
- No utiliza punteros a void (-50 puntos)
- No utiliza punteros a función (-50 puntos)
- No compila (-100 puntos)

En caso de existir nota negativa ésta será reemplazada por un 0.