INF-134 Estructuras de Datos, 2018-2

Tarea 1

Profesora: Natalia González.
natalia.gonzalezg@usm.cl

Ayudantes:
Camilo Saldías (camilo.saldias.12@sansano.usm.cl)
Alejandro Vilches (alejandro.vilches@sansano.usm.cl)

Fecha de entrega: 14 de Octubre, 2018
Plazo máximo de entrega: 5 días.

1. Reglas del Juego

La presente tarea debe hacerse en grupos de 3 personas. Toda excepción a esta regla debe ser conversada con los ayudantes. No se permiten de ninguna manera grupos de más de 3 personas. Las tareas deben compilar en los computadores que se encuentran en el laboratorios LDS – B-032 (Fedora 17 de 64 bits). Debe usarse el lenguaje de programación C. Se recomienda compilar en el terminal usando `gcc archivo.c -o output -Wall`. Recordar que una única tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobar el curso.

2. Objetivos

Comprender y familiarizarse con las estructuras y tipos de datos básicos que provee el Lenguaje de Programación C. Entre los conceptos mas importantes, se encuentran:

- Paso de parámetros por valor.
- Paso de parámetros por referencia.
- Asignación de memoria dinámica.
- Manipulación de punteros.
- Manejo de E/S.
- Recursividad

Además se fomentará el uso de las buenas prácticas y el orden en la programación de los problemas correspondientes.

3. Problemas a Resolver

Problema 1: Piezas en Jaque

Su trabajo es escribir el programa `ajedrez.c` que lea una configuración de un tablero de ajedrez y responda si existe un rey bajo ataque ("en jaque"). Un rey está en jaque si está amenazado por una pieza del

contrincante (es decir, está en una casilla que puede ser tomada por la pieza de un oponente en su próximo movimiento). Las piezas blancas serán representadas por letras mayúsculas; por otro lado, las piezas negras serán representadas por letras minúsculas. Las piezas blancas atacan desde el lado inferior del tablero, mientras que las negras desde el lado superior.

Para aquellos que no posean conocimiento del ajedrez, el movimiento de cada pieza es el siguiente:

- Peón (p o P): solo se puede mover hacia adelante, una casilla a la vez, pero come piezas en diagonal.
- Caballo (c o C): Tiene un movimiento especial, es la única pieza que puede saltar por sobre otras. El caballo se mueve en forma de L. (ver en el ejemplo)
- Alfil (a o A): Se puede mover cualquier número de casillas en diagonal (avanzando o retrocediendo).
- Torre (t o T): Se puede mover cualquier número de casillas verticalmente u horizontalmente (avanzando o retrocediendo).
- Reina (q o Q): Se puede mover cualquier cantidad de casillas, en cualquier dirección (diagonal, horizontal o vertical, avanzando o retrocediendo).
- Rey (k o K): Se puede mover sólo 1 casilla a la vez, en cualquier dirección (diagonal, horizontal o vertical, avanzando o retrocediendo).

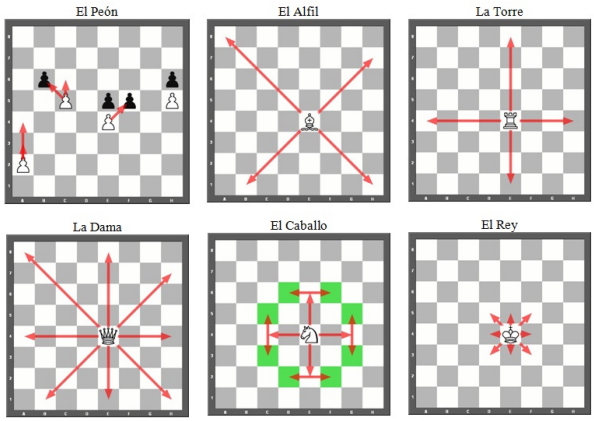

Figura 1: Ejemplo del movimiento de las piezas de ajedrez

Formato de Entrada

La entrada consiste en un archivo `tableros.txt`, el cual consta de un número entero N el cual enumera la configuración de tablero de ajedrez a revisar. Cada tablero será de 8×8, donde un " . " representa un espacio vacío. Las letras en mayúscula o minúscula, como se mencionó anteriormente, representan las piezas. La entrada termina cuando se recibe un valor de $N = 0$. En cada configuración aparecerán ambos reyes, tanto el blanco como el negro.

Ejemplo de Input

```
1
..k.....
ppp.pppp
........
```

```
.T...A..
........
........
PPPPPPPP
K.......
2
tcaqkact
pppppppp
........
........
........
........
PPPPPPPP
TCAQKACT
3
tcaqk.ct
ppp..ppp
....p...
...p....
.aPP....
.....C..
PP..PPPP
TCAQKA.T
0
```

Formato de Salida

Para cada configuración de tablero, se debe entregar la siguiente respuesta (imprimiendo a pantalla), según corresponda:

```
Tablero #N: Rey Blanco en Jaque
Tablero #N: Rey Negro en Jaque
Tablero #N: No hay Rey en Jaque
```

Ejemplo de Output

```
Tablero #1: Rey Negro en Jaque
Tablero #2: No hay Rey en Jaque
Tablero #3: Rey Blanco en Jaque
```

Problema 2: Riesgo Académico

Una institución educacional ha decidido organizar la información académica de sus alumnos. Para ello, utiliza la siguiente definición de curso:

```
typedef struct{
    char sigla[6];
    unsigned int nota;
} curso;
```

Además, se cuenta con la siguiente definición de un alumno:

```
typedef struct{
    char nombre[30];
    unsigned int rol;
    curso cursosRealizados[50];
    int numCursos;
} alumno;
```

Donde el campo `numCursos` indica la cantidad de cursos tomados por el alumno a lo largo de su carrera. Notar que, de acuerdo a esta definición, el alumno no puede tomar más de 50 cursos. Adicionalmente, el rol del alumno no cuenta con puntos ni guiones, y las calificaciones de cada curso se realizan en una escala de 0 a 100.

La institución educacional, para simplificar el sistema, ha decidido almacenar estos datos en un único archivo binario con el siguiente formato:

- El primer elemento del archivo es un `int n`.
- Le siguen `n` componentes de tipo `alumno`.

La institución educacional está altamente preocupada por sus alumnos, sobre todo por los alumnos que se encuentran en riesgo académico. La institución considera que un alumno se encuentra en riesgo académico si el promedio de notas del alumno, a lo largo de toda su carrera, es menor a 55.

Por lo anterior, usted deberá crear una función `riesgoAcademico`, la cual recibe como parámetro el nombre del archivo binario con los datos de los alumnos, y retorna un arreglo con los datos de los alumnos que se encuentran en riesgo académico, la cual estará contenida en el programa `infoAcademica.c`.

Formato de Entrada

La entrada de datos será a través de un único archivo binario, el cual contiene la información de todos los alumnos de la institución educacional de la forma explicada anteriormente. El nombre de este archivo será ingresado como argumento al momento de ejecutar su programa, de la siguiente forma:

```
./infoAcademica "alumnos.dat"
```

Para leer el nombre del archivo pasado como argumento, usted debe poder aceptar parámetros en la función `main` de su programa, lo que se realiza de la siguiente forma:

```
int main ( int argc, char *argv[] )
```

Donde `argc` es la cantidad de argumentos entregados, y el arreglo `argv` contiene los argumentos entregados. Por defecto, el primer argumento (en la posición 0 del arreglo `argv`) es el nombre del programa, por lo que el valor por defecto de `argc` es 1.

Notar que en el caso de que el archivo indicado como argumento no exista, el programa debe indicar por pantalla '`Archivo no existe`' y terminar su ejecución.

Formato de Salida

El programa debe imprimir en el archivo '`riesgo_academico.txt`' el resultado de invocar a la función riesgoAcedemico con los datos provistos en el archivo de entrada. El archivo debe contener el nombre y rol de todos los alumnos que se encuentren en situación de riesgo académico. Los datos de cada alumno en situación de riesgo académico deben estar en una única línea en el archivo.

4. Entrega de la Tarea

La entrega de la tarea debe realizarse enviando un archivo comprimido llamado

`tarea1-apellido1-apellido2-apellido3.tar.gz`

(reemplazando sus apellidos según corresponda) al sitio de AULA del curso, a más tardar el día 14 de Octubre de 2018, a las 23:55:00 hs (Chile Continental), el cual contenga:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. ¡Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, y qué programó cada integrante del grupo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias.

4.1. Pre-Entregas

Se dará la opción de realizar 2 pre-entregas, los días 28 de septiembre y 5 de octubre, donde se corregirá exclusivamente compilación y ejecución (detalles de estructuración del código fuente, orden, etc. serán revisados luego de la entrega final). La pre-entrega debe realizarse via Aula enviando un archivo comprimido llamado

`tarea1_PRE-ENTREGA-apellido1-apellido2-apellido3.tar.gz`

(Si no se indica claramente que es "Pre-Entrega" se considerará como entrega final).

4.2. Entrega Temprana

Se dará la opción de realizar entrega de la versión final de la tarea de manera adelantada, bonificando su nota de la siguiente manera:

1. Entregas hasta el 28/09 a las 23:55:00 hrs via Aula: Bonificación de +10 pts en la nota de la tarea.
2. Entregas hasta el 05/10 a las 23:55:00 hrs via Aula: Bonificación de +5 pts en la nota de la tarea.

5. Restricciones y Consideraciones

- Por cada día de atraso en la entrega de la tarea se descontarán 10 puntos en la nota.
- El plazo máximo de entrega es 5 días después de la fecha original de entrega.
- Las tareas deben compilar en los computadores que se encuentran en los laboratorios LDS (Fedora 17 de 64 bits). **Las tareas que no compilen no serán revisadas y serán calificadas con nota 0**.
- Por cada *Warning* en la compilación se descontarán 5 puntos.
- Si se detecta **COPIA** la nota automáticamente será 0 (CERO), para todos los grupos involucrados. El incidente será reportado al Jefe de Carrera.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

6. Consejos de Programación

El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota.

Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
*   TipoFunción NombreFunción
******
*   Resumen Función
******
*   Input:
*       tipoParámetro NombreParámetro : Descripción Parámetro
*        .......
******
*   Returns:
*       TipoRetorno, Descripción retorno
*****/
```

Por cada comentario faltante, se restarán 5 puntos.

Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por **cada bloque mal indentado, se quitarán 10 puntos**.