TRABAJO PRÁCTICO 3:
ORIGEN DE LA CAIDA DE LA ACADEMIA SITH

75.40 Algoritmos y Programación I
Curso: Méndez

24 de mayo de 2017

1. Objetivo

- Que el alumno analice un problema dado, determine los requerimientos pedidos y diseñe una solución acorde.
- Que el alumno se familiarice con el manejo de archivos de texto y binarios, valiéndose de todos los elementos vistos en el curso.
- Que el alumno profundice aún más las bases de las buenas prácticas de programación.

2. Historia

La academia siguió en funcionamiento haciendo caso omiso de las advertencias de Visas Marr e ignorando la plataforma predictiva de batallas de jedi contra sith, ya que los altos mandos estaban convencidos que nunca se llegaría a esa situación pues los jedi no llegaban a completar su entrenamiento al ser "expulsados".
Cada semestre más y más alumnos estaban deseosos de poder asistir a la academia, por lo cual se hizo urgente contar con un sistema de administración del alumnado y todos los recursos necesarios para "priorizar la educación y la libertad de expresión de la Fuerza". Esta tarea cayó una vez más en los alumnos desarrolladores del mítico detector de alineamiento con la Fuerza y el avanzadísimo y poco utilizado simulador de batallas.
El sistema de administración necesita poder dar de alta a nuevos alumnos, "expulsarlos" cuando sea conveniente, y poder modificar los datos de registro de algunos. También debe ser capaz de llevar un detalle del desempeño académico de cada alumno por curso realizado, y poder seleccionar a los alumnos que más podrán ayudar al imperio.
Algunos de los alumnos desarrolladores se dieron cuenta que las medialunas gratis no valían tanta matanza y camuflaron algunas operaciones para poder avisar a los estudiantes que iban a ser "expulsados" con el propósito de salvarles la vida y luego rebelarse contra las autoridades de la academia, iniciándose así una serie de batallas como las que Visas Marr había visualizado.

3. Especificaciones técnicas: Archivos

Se contará con los siguientes archivos respetando las estructuras de sus registros:

- **sistyplan.dat**: un archivo binario que contiene la información de los sistemas y planetas de los alumnos que se van registrando. Este archivo se encuentra ordenado ascendentemente por código de sistema/planeta y cada registro de este archivo contiene la siguiente información:
  1. **Código**: número entero autoincremental que arranca desde 0.
  2. **Descripción**: cadena de hasta 50 caracteres.
  3. **Indicador de tipo**: 'S' si es un sistema, sino 'P' para los planetas.
- **alumnos.txt**: un archivo de texto que contiene la información personal de los alumnos que siguen en la academia. Este archivo se encuentra ordenado ascendentemente por código de alumno y cada línea de este archivo contiene la siguiente información en el orden presentado y separados por punto y coma (;):

1. **Código**: cadena de hasta 10 caracteres.
2. **Nombre**: cadena de hasta 20 caracteres.
3. **Apellido**: cadena de hasta 20 caracteres.
4. **Email**: cadena de hasta 100 caracteres.
5. **Sistema de origen**: código de sistema según el archivo de sistemas y planetas.
6. **Planeta de origen**: código de planeta según el archivo de sistemas y planetas.

- **respxalum.txt**: un archivo de texto que contiene el detalle de los tipos de respuestas que cada alumno expone en las distintas clases, por lo cual un alumno puede tener muchos registros, uno o ninguno. Este archivo se encuentra ordenado ascendentemente por código de alumno y fecha y cada línea de este archivo contiene la siguiente información en el orden presentado y separados por punto y coma (;):

  1. **Código de alumno**: código de alumno según el archivo de alumnos.
  2. **Fecha**: cadena de 8 caracteres que tiene una fecha en formato YYYYMMDD.
  3. **Descripción del curso**: cadena de hasta 50 caracteres.
  4. **Cantidad de respuestas tendientes a la luz**: número entero siempre positivo.
  5. **Cantidad de respuestas tendientes a la oscuridad**: número entero siempre positivo.

4. Especificaciones técnicas: Biblioteca adminsith.h

Se debe desarrollar la biblioteca adminsith.h la cual debe contar únicamente con las siguientes funciones:

```
void agregarSistema(char* descripcion);
void agregarPlaneta(char* descripcion);
void consultarSistemasYPlanetas();
void actualizarAlumnos(char* archivoActu);
int expulsarJedi(int porcentajeMinimo);
void consultarDatosAlumno(char* codigoALumno);
int actualizarRespuestas(); //Opcional para los que vayan por la promocion
```

La implementación de la biblioteca debe tener en cuenta el siguiente comportamiento de cada función:

1. **agregarSistema**: recibirá por parámetro la descripción (no puede ser vacía) del sistema a agregar. El código de sistema/planeta será autoincremental, es decir el nuevo sistema tendrá como código el último sistema/planeta generado +1. Si el archivo no existe debe ser creado en ese momento.

2. **agregarPlaneta**: recibirá por parámetro la descripción (no puede ser vacía) del planeta a agregar. El código de sistema/planeta será autoincremental, es decir el nuevo planeta tendrá como código el último sistema/planeta generado +1. Si el archivo no existe debe ser creado en ese momento.

3. **consultarSistemasYPlanetas**: imprimirá por pantalla código, descripción e indicador de tipo de cada registro almacenado en el archivo de sistemas y planes.

4. **actualizarAlumnos**: recibirá por parámetro el nombre del archivo con las altas y/o modificaciones realizadas a los alumnos. Este nuevo archivo tendrá la misma estructura que el archivo alumnos.txt y también estará ordenado ascendentemente por código. Los alumnos que estén en el archivo de actualizaciones pero no en el original serán agregados (dados de alta), sino se actualizará la información del original con la que se encuentre en el de actualizaciones. Se debe tener en cuenta que no se podrá dar de alta o actualizar información inválida, por ejemplo que no exista el planeta y/o sistema indicado, el código de alumno esté vacío y cualquier otro caso que se considere relevante; los registros en esta situación deben ser ignorados, es decir no deben ser dados de alta o actualizar la información del alumno.

5. **expulsarJedi**: recibirá por parámetro el mínimo porcentaje de respuestas tendientes a la luz que debe tener un alumno para ser considerado jedi y, por ende, expulsado; este porcentaje debe ser un número entre 70 y 100. Cualquier alumno que sea igual o supere dicho porcentaje debe ser eliminado del archivo de alumnos, es decir el archivo debe actualizarse dejando los alumnos que no alcancen dicho porcentaje. La función devolverá la cantidad de jedi expulsados.

6. **consultarDatosAlumno**: recibirá por parámetro el código del alumno del que se quiere consultar los datos. Se debe imprimir por pantalla nombre y apellido, email y las descripciones de sistema y planeta de origen.

7. **actualizarRespuestas**: actualizará el archivo de las respuestas de alumnos eliminando los registros correspondientes al detalle de los alumnos que no existen. Esta función tiene el propósito oculto de modificar las respuestas de los alumnos cuyo porcentaje de respuestas tendientes a la luz sea igual o superior a 70, y devolverá la cantidad de jedi salvados de ser expulsados. (Nota: la forma de modificación de las respuestas elegidas para salvar a los jedi queda a libre elección, pero no debe ser tan obvia como eliminar los registros o dejar las cantidades en 0)

5. Especificaciones técnicas: Programa acadesith.c

Se debe desarrollar un programa llamado acadesith.c cuya función main() se encargará de invocar las funciones de la biblioteca antes mencionada. Para ésto se ejecutará el programa por línea de comando enviando los parámetros necesarios para cada función.
El primer parámetro enviado a este programa debe ser un código identificador de operación (string) según la siguiente convención:

- **-addsi** para llamar a la función *agregarSistema*.
- **-addpl** para llamar a la función *agregarPlaneta*.
- **-consp** para llamar a la función *consultarSistemasYPlanetas*.
- **-updal** para llamar a la función *actualizarAlumnos*.
- **-delje** para llamar a la función *expulsarJedi*.
- **-conal** para llamar a la función *consultarDatosAlumno*.
- **-updre** para llamar a la función *actualizarRespuestas*.

Se debe tener en cuenta que cualquier precondición establecida en las funciones de la biblioteca deben ser validadas en este programa, caso contrario el programa mostrará por pantalla el motivo por el cual no se haya podido invocar la función deseada.

6. Especificaciones técnicas: Compilación

Para compilar el trabajo se debe poder ejecutar la siguiente línea:
gcc adminsith.c acadesith.c -o acadesith -std=c99 -Wall -Wconversion -Werror

7. Condiciones de aprobación

El trabajo debe poder ser compilado sin problemas, deben estar desarrolladas tdas las funciones y comportarse lo más cercano posible a los requerimientos que figuran en el enunciado.
La legibilidad y extensbilidad del código también serán tenidas en cuenta como en los trabajos anteriores. Un trabajo que funciona perfecto pero que es imposible de leer puede ir a reentrega directa.

8. Entrega

Se debe generar un .zip que contenga los archivos adminsith.c, adminsith.h y acadesith.c. Este zip debe ser subido a la página de la materia, en la sección habilitada para tal fin, con límite el día miércoles 14/06 a las 23.59 hs.
Este trabajo práctico tendrá 1 única posibilidad de reentrega con límite a determinar. De manera similar, se debe generar un .zip conteniendo los archivos adminsith.c, adminsith.h y acadesith.c actualizados y subirlo en la página de la materia.