TRABAJO PRÁCTICO 2015

PREGUTNADOS

El trabajo práctico de este año consistirá en el desarrollo de una versión alternativa del popular juego de preguntas y respuestas ***Preguntados*** (https://preguntados.com/). Esta versión se adaptará para facilitar su construcción en el marco de la cátedra ***Algoritmos y Estructuras de Datos***. En lugar de contemplar las categorías tradicionales (arte, ciencia, historia, entretenimiento, deportes y geografía), se trabajará utilizando como categorías las materias del 1º cuatrimestre de 1º año de ISI: ***Física I, Matemática Discreta, Algoritmos y Estructuras de Datos, Arquitectura de Computadoras*** y ***Análisis Matemático I***.

PREGUTNADOS: METODOLOGÍA DE TRABAJO Y PAUTAS DE ENTREGA

La aplicación se construirá progresivamente a lo largo del año por medio de *etapas de desarrollo*.
En cada etapa la cátedra proveerá un enunciado que detalle el trabajo a realizar, el cual podrá estar acompañado por archivos, prototipos, etc. que deberán ser utilizados/consultados durante la construcción de la aplicación.

1. Conformación de grupos

El trabajo es grupal y debe estar conformado por exactamente 3 personas (de la misma o de distinta comisión).
No se admitirán grupos con otra cantidad de integrantes. Una vez formados los grupos, se debe informar a la cátedra los integrantes por medio de una respuesta en el foro de consultas de la materia al tema: ***Conformación Grupos TP***. Una vez enviada esta información la cátedra asignará a cada grupo un número. Este número será el que deberá utilizarse al momento de realizar las entregas de cada una de las etapas.
Nota: No se recibirán entregas sin número de grupo.

2. Documentación a presentar

Las entregas asociadas a cada una de las etapas de desarrollo se realizarán ***exclusivamente*** por medio de la tarea ***Trabajo Práctico 1***, creada para tal fin en el campus virtual de la materia. Los archivos asociados a cada entrega deben comprimirse en un archivo ZIP o RAR nombrado bajo la estructura:

Gn-Em-2015

donde:

- ***n*** es el número de grupo asignado por la cátedra.
- ***m*** es el número de etapa.

Cada entrega la realizará ***sólo uno de los integrantes del grupo***, indicando claramente en un documento llamado ***integrantes.txt*** (contenido dentro del archivo comprimido) el nombre de cada uno de los integrantes y su email.
Nota: No se aceptará bajo ninguna circunstancia entregas fuera de término.

3. Consultas

Las consultas referentes a los enunciados de cada una de las etapas se atenderán por medio del foro. Se dispondrán horarios en los cuales los profesores atenderán las inquietudes y/o problemas de cada uno de los grupos.

PREGUTNADOS: ETAPA DE DESARROLLO Nº 1

El objetivo de esta ***primera etapa*** es interiorizarse sobre el juego y definir el conjunto de preguntas que luego se implementarán como parte de la aplicación. Además, se solicita definir la presentación y el logo de la aplicación.

1. Investigación

A fin de conocer más sobre el juego, sus comienzos y su funcionamiento, se solicita elaborar un informe que respete las siguientes secciones:

- *Carátula*: Indicando claramente el número de grupo, sus integrantes, sus direcciones de correo y la comisión a la que pertenecen.
- *Contenido*: Detallando (como mínimo):
  - *Información sobre su creador*:
    - Nombre, edad, nacionalidad, etc.
    - Qué estudió y dónde.
    - Nombre de su empresa, cuántos empleados tiene, otros productos lanzados, productos en desarrollo, etc.
  - *Información sobre el juego*:
    - En qué año se creó Preguntados.
    - En qué plataformas está disponible.
    - Alcance del juego a nivel mundial, negocios surgidos a partir de él.
    - Información que refleje el éxito que obtuvo el juego en la comunidad.
  - *Información sobre el funcionamiento del juego*:
    - Información que se solicita cuando una persona quiere convertirse en usuario de la aplicación.
    - Explicar el funcionamiento del juego: contra quienes se puede jugar, quién elabora las preguntas, cómo se juega, quién es el ganador, etc.
- *Referencias*: Incluyendo las páginas web, artículos periodísticos, etc. de donde se haya obtenido la información.

2. Elaboración de preguntas

Teniendo en cuenta que el juego funciona con una extensa cantidad de preguntas, su elaboración se realizará en forma conjunta. En base a las categorías mencionadas en el enunciado del trabajo práctico, cada grupo deberá generar dos preguntas (con sus correspondientes opciones) para cada categoría. Los grupos deberán enviar las preguntas respondiendo un mensaje al tema ***Base de Preguntas TP1*** en el foro de consultas de la materia creado para tal fin.

Cada pregunta será cargada por un docente como una nueva fila de la planilla compartida disponible en https://docs.google.com/spreadsheets/d/1fgzEH45wrSOoOwo5ix5c3uT-Fhp-Y2bR4_Y64voUHTo/edit?usp=sharing. Para cada pregunta el grupo indicará:

- Nº de grupo.
- Categoría de la pregunta.
- Pregunta.
- Opción 1.
- Opción 2.
- Opción 3.
- Opción 4.
- Nº de opción correcta.

Nota: No se aceptarán preguntas repetidas ni similares a las ya definidas, por lo cual antes de enviar las preguntas el grupo deberá revisar el contenido de la planilla compartida.

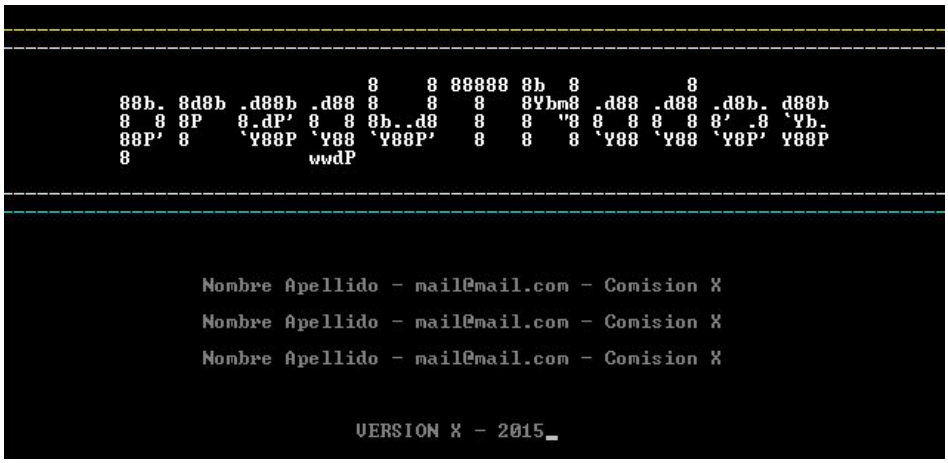
3. Codificación de la presentación y diseño del logo

Con el objetivo de comenzar a implementar el juego, cada grupo deberá desarrollar la pantalla inicial de su aplicativo (presentación) junto con el logo del programa. El diseño de cada uno de estos aspectos quedará a cargo del grupo, debiendo respetar lo siguiente:

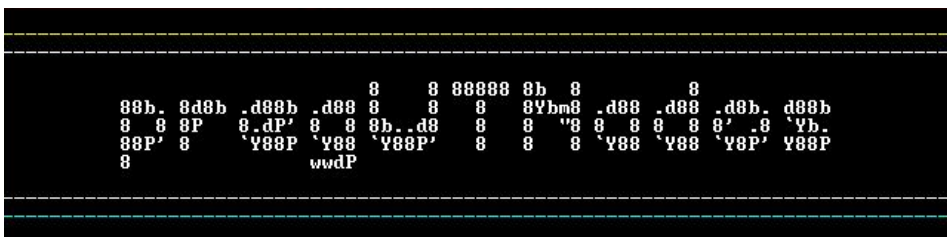
1. La presentación deberá contener (como mínimo):
    - Logo del juego.
    - Nombre, apellido, email y comisión de los integrantes del grupo.
    - Número de versión y año.
2. El logo del juego deberá hacer referencia al nombre del juego.

Tanto la presentación como el logo deberán implementarse haciendo uso de funciones.

Ejemplo de presentación:



Ejemplo de logo:



4. Documentos a entregar

En esta etapa, se solicita ***entregar***:

1. El ***informe*** desarrollado en respuesta al ***punto 1*** en formato ***PDF***.
2. El ***código C++*** *junto al* ***archivo ejecutable*** implementado en respuesta al ***punto 3***.

Además se deberá enviar un mensaje al foro en respuesta al tema ***Base de Preguntas TP*** con las preguntas propuestas por el grupo. Un docente será el encargado de colocar esta información en la planilla compartida luego de verificar que cumpla con los requisitos solicitados.

~~***Fecha máxima de entrega: 03/07/2015***~~

Etapa SUPERADA

PREGUTNADOS: ETAPA DE DESARROLLO Nº 2

El objetivo de esta *segunda etapa* es implementar el entorno del juego, definiendo la estructura general del mismo y sus secciones. Además se desarrollará el registro de usuario, el inicio de sesión y la creación de las partidas.

1. Menú principal

Luego de la presentación del juego, el usuario deberá visualizar un menú llamado "Menú principal". Este menú deberá mostrar las siguientes opciones:

1. *Registrarse*

   Al ingresar a esta opción el usuario tendrá la posibilidad de crear una nueva cuenta. Para esto el programa le solicitará nombre de usuario y contraseña. Podrán existir como máximo 100 cuentas.

   - **Nombre de usuario: Quedará definido por una cantidad mínima de 6 caracteres y máxima de 10, los cuales podrán ser letras, números y/o símbolos del conjunto {+,-,/,*,?,¿,!,¡}. Deberá ser único para cada usuario registrado, debiendo:**
     a. **Comenzar con una letra minúscula.**
     b. **Tener al menos 2 letras mayúsculas.**
     c. **Tener como máximo 3 dígitos.**

     Ejemplos de nombres de usuario incorrectos: AbC123 (no cumple a), pTS!1234 (no cumple c), g178Mci (no cumple b),mARtin123gomez (tiene mas de 10 caracteres).

     Ejemplos de nombres de usuario correctos: mARtin12, jo97!AR.

   - Contraseña: Quedará definida por una cantidad mínima de 6 caracteres y máxima de 8, los cuales deberán ser letras y números. Su conformación no podrá darse al azar, sino que deberá respetar lo siguiente:
     a. No debe tener más de 3 caracteres numéricos consecutivos.
     b. No debe tener 2 caracteres consecutivos que refieran a letras alfabéticamente consecutivas (ascendentemente). Este criterio es válido tanto para letras mayúsculas, minúsculas o combinación de ambas.
     c. Debe tener por lo menos una letra mayúscula.

Ejemplos de contraseñas mal formadas: Ach32 (la cantidad debe ser entre 6 y 8 caracteres), doriT1234 (no cumple a), aBuel123 (no cumple b), alejo123 (no cumple c).

Ejemplo de contraseñas bien formadas: Achus32, 125Af89, Alejo123, DORITO45, 4AC2SA.

Nota: Tanto en el nombre de usuario como en la contraseña deben distinguirse mayúsculas y minúsculas.

Una vez ingresado un nombre de usuario correcto y una contraseña válida, se le solicitará al usuario que confirme la contraseña (reingreso). Si la cadena reingresada coincide con la original, se informará el éxito del registro de usuario y se preguntará si desea iniciar sesión. En caso afirmativo, se ejecutará la secuencia de operaciones asociada a la opción "Iniciar sesión" del menú principal. En caso contrario, el programa volverá al menú principal. Por otro lado, si el reingreso de la contraseña no coincide con la cadena original, se informará el fracaso de la operación de registro y el programa volverá al menú principal.
El almacenamiento de los registros de usuario se implementará de forma gradual a lo largo de las siguientes etapas del trabajo práctico. En esta etapa se solicita implementar el registro de usuarios de forma tal que exista un único conjunto de variables asociadas a la creación de cuentas de usuario. La información almacenada en estas variables se sobreescribirá cada vez que se ejecute la opción "Registrarse" y se tenga éxito en la creación de la nueva cuenta. Si el registro de un nuevo usuario fracasa, los datos precedentes deben mantenerse intactos (si existiesen).
Nota: En esta etapa, la información asociada a las cuentas de usuario se perderá una vez que finalice la ejecución del programa. Por este motivo, siempre será necesario crear una cuenta para utilizar el juego.

2. *Iniciar Sesión*

Al ingresar a esta opción el programa solicitará nombre de usuario y contraseña. Si los datos ingresados coinciden con la información de alguna de las cuentas registradas, se procede a mostrar un mensaje de bienvenida con el nombre del usuario y, a continuación, el "Menú de juego". Por el contrario, si los datos ingresados no corresponden a una cuenta de usuario válida se informa el error. Una vez cometido el primer error, el usuario tendrá 3 intentos consecutivos para ingresar los datos correctos. Si en alguno de estos intentos los datos son correctos, se continúa con la ejecución normal. Vencidos los tres intentos, el programa ejecuta un *proceso de verificación captcha*. Este proceso genera un código alfanumérico que es mostrado al usuario a fin de garantizar su existencia. Para esto se le solicita que ingrese el código, lo que dará lugar a una de las siguientes situaciones:

- El usuario ingresa correctamente el código, entonces el programa le otorga 3 nuevos intentos para el inicio de sesión. Si vencidos estos tres intentos no se ha logrado iniciar sesión, el programa mostrará un mensaje de error por posible violación de seguridad y luego finalizará su ejecución.
- El usuario ingresa un código incorrecto, entonces el programa mostrará un mensaje de error por posible violación de seguridad y luego finalizará su ejecución.

Proceso de verificación captcha

Este proceso de verificación consiste en la generación de una cadena de caracteres según el siguiente algoritmo:

"Sea R un número aleatorio de 5 cifras (comprendido entre 10000 y 50000 inclusive), P la cadena de caracteres 'ALGORITMOS' y N el nombre de usuario con el cual se desea iniciar sesión. Tomar el primer y último caracter de N, transformarlos al valor numérico del código ASCII al cual corresponden y sumarlos a R. Ordenar las cifras de este nuevo número de forma descendente, lo que dará lugar a un nuevo número que llamaremos A. Formar una nueva cadena que alterne los caracteres de P con las cifras de A de forma tal que cada dos caracteres de P (uno mayúscula y el siguiente minúscula) se encuentre una cifra de A. Luego, invertir la cadena resultante a fin de obtener el código de verificación."

Para validar este proceso se utilizará SPOJ. Tomando como punto de partida el siguiente código, cada grupo de trabajo deberá realizar una implementación de la función *captcha()* que cumpla con la descripción precedente. Luego, deberá subir a http://www.spoj.com/problems/CAPTCHA/ el código provisto por la cátedra agregando su implementación. Una vez que se obtenga el aprobado, el grupo podrá utilizar la función implementada como parte de su juego.

```
#include <string>
#include <iostream>
using namespace std;

string captcha(int R,string P,string N);

int main(int argc, char *argv[]) {
	int numeroAleatorio;
	string nombreUsuario;
	cin>>nombreUsuario;
	cin>>numeroAleatorio;
	cout<<captcha(numeroAleatorio,"ALGORITMOS",nombreUsuario);
	return 0;
}

string captcha(int R,string P,string N){
	//Implementar esta función y luego probarla en SPOJ.
}
```

```
Ejemplo:
   Sean: R: 21300
         P: ALGORITMOS
         N: gomezHI83
   El proceso de generación de captcha es:
         - Primer caracter de N: 'g' → ASCII(g) = 103
         - Último caracter de N: '3' → ASCII(3) =  51
         - Suma de los dos ASCII con R: 103 + 51 + 21300 = 21454
         - Ordenando las cifras A=54421
         - Cadena intermedia: "Al5Go4Ri4Tm2Os1"
         - Código captcha: 1sO2mT4iR4oG5lA
```

Nota: Junto con este enunciado se entrega un archivo ejecutable que contiene una solución a este problema. Este archivo puede usarse como soporte a fin de realizar pruebas y comparar resultados.

3. *Estadísticas del Juego*
   Al ingresar a esta opción el usuario visualizará un conjunto de estadísticas del juego. Debido a que en esta etapa no se solicita esta funcionalidad, al ingresar a esta opción se mostrará el mensaje "FUNCIONALIDAD EN DESARROLLO". Luego, el programa volverá al menú principal.

4. *Cerrar aplicación*
   Al ingresar a esta opción el programa pedirá al usuario que confirme la acción. Si la acción es confirmada, el programa finalizará su ejecución. En caso contrario, el programa debe volver al "Menú principal".

Ejemplo de "Menú principal":

2. Menú de juego

Una vez iniciada la sesión, el usuario deberá visualizar un menú denominado "Menú de juego". Este menú deberá mostrar las siguientes opciones:

1. *Mi cuenta*
   Al ingresar a esta opción el usuario visualizará su información (nombre de usuario y contraseña). Luego, el programa volverá al "Menú de juego".

2. *Iniciar partida*
   Al ingresar a esta opción el usuario creará una nueva partida contra un usuario elegido al azar. Para esto el programa le solicitará un nombre de partida. Luego, el juego elegirá aleatoriamente el oponente y se lo informará al usuario actual. Posteriormente, el usuario tirará la ruleta en busca de la categoría de la pregunta a responder y el programa determinará cuál es la pregunta que debe responderse. Este proceso se repetirá para cada una de las 10 preguntas de la partida. Ambos usuarios responderán las mismas 10 preguntas de forma consecutiva. Una vez que el usuario actual ha respondido las 10 preguntas, la partida quedará pendiente (es decir, en espera) de que el otro usuario la juegue. El usuario actual volverá al "Menú de juego".
   - Nombre de partida: Quedará definido por una cantidad mínima de 6 caracteres y máxima de 15, los cuales podrán ser letras, números y/o símbolos del conjunto {+,-,/,*,?,¿,!,¡}. No existe un formato para su conformación, pero debe ser único.
   - Elección aleatoria del oponente: El programa generará un número aleatorio entre 1 y la cantidad máxima de usuarios (100). Este número indicará el orden del usuario elegido. Es decir, si el número generado es el 5 el usuario actual jugará la partida contra el 5º usuario

registrado en el juego. En caso de que no exista ese número de usuario o de que el número generado corresponda al usuario actual, se repetirá este proceso. Una vez sorteado el oponente, el usuario actual verá un mensaje similar al siguiente:

"La partida se jugará contra: *oponente*"

donde *oponente* será el nombre de usuario del oponente sorteado.
Nota: En esta etapa, debido a la imposibilidad de mantener almacenado más de un usuario, la elección del oponente se limitará a la generación del número aleatorio comprendido entre 1 y 100, sin validar que este valor no corresponda al usuario actual. En lugar del nombre de usuario, el mensaje deberá mostrar el número generado.

- Ruleta: Ver la sección "**3. Ruleta**".

El almacenamiento de las partidas se implementará de forma gradual. En esta etapa se solicita implementar las partidas de forma tal que exista un único conjunto de variables asociado a su creación. La información almacenada en estas variables se sobreescribirá cada vez que se ejecute la opción "Iniciar partida".

3. *Partidas iniciadas*
   Al ingresar a esta opción el usuario visualizará el listado de partidas que él ha iniciado pero que aún no han sido jugadas por su oponente. Este listado se ordenará cronológicamente según su fecha de creación. Debido a que en esta etapa no se solicita implementar esta funcionalidad, al ingresar a esta opción se mostrará el mensaje "FUNCIONALIDAD EN DESARROLLO". Luego, el programa volverá al "Menú de juego".

4. *Partidas pendientes de juego*
   Al ingresar a esta opción el usuario visualizará las partidas en las cuales otros usuarios lo han incluido como oponente. Una vez que el usuario actual juegue la partida, se le mostrará el resultado final (puntajes y ganador) y la misma quedará cerrada. Debido a que en esta etapa no se solicita implementar esta funcionalidad, al ingresar a esta opción se mostrará el mensaje "FUNCIONALIDAD EN DESARROLLO". Luego, el programa volverá al "Menú de juego".

5. *Partidas terminadas por el oponente*
   Al ingresar a esta opción el usuario visualizará las partidas que él ha iniciado y que los oponentes han finalizado. En este caso, se le mostrará al usuario el resultado final (puntajes y ganador) y la misma quedará cerrada. Debido a que en esta etapa no se solicita implementar esta funcionalidad, al ingresar a esta opción se mostrará el mensaje "FUNCIONALIDAD EN DESARROLLO". Luego, el programa volverá al "Menú de juego".

6. *Mis estadísticas*
   Al ingresar a esta opción el usuario visualizará un conjunto de estadísticas asociadas a sus partidas. Debido a que en esta etapa no se solicita implementar esta funcionalidad, al ingresar a esta opción se mostrará el mensaje "FUNCIONALIDAD EN DESARROLLO". Luego, el programa volverá al "Menú de juego".

7. *Cerrar sesión*
   Al ingresar a esta opción el programa pedirá al usuario que confirme la acción. Si la acción es confirmada, el programa cierra la sesión y vuelve al "Menú principal". En caso contrario, el programa debe volver al "Menú de juego".

Ejemplo de "Menú de juego":

3. Ruleta

La selección aleatoria de categorías y preguntas se implementará utilizando una ruleta. Esta ruleta simulará la elección de la categoría y determinará la pregunta que deberá responder el usuario. Cada partida guardará esta información en dos arreglos paralelos de 10 elementos enteros (un arreglo almacenará el identificador de la categoría y el otro el identificador de la pregunta). En cada uno de los arreglos el subíndice de los elementos determinará la correspondencia del par {categoría, pregunta}.
Se desarrollarán dos módulos:

1. *Elección de la categoría*
   Este módulo determinará la categoría de la pregunta. Junto con este enunciado se entregan tres archivos ejecutables que contienen posibles soluciones para este módulo. Cada grupo deberá elegir uno de estos esquemas y realizar una implementación propia. Sin embargo, también podrán optar por la realización de nuevos desarrollos (similares a los propuestos), en los cuales la determinación de la categoría se logre de forma aleatoria. Una vez elegida la categoría, se la almacena en el subíndice correspondiente al número de tiro de la ruleta en el arreglo de categorías. Luego, se procede a la elección de la pregunta.

2. *Elección de la pregunta*
   Este módulo determinará la pregunta que deberán responder los usuarios involucrados en la partida. Se sabe que las categorías están conformadas de la siguiente manera:

IDENTIFICADOR	CATEGORÍA	CANTIDAD DE PREGUNTAS
0	Física I	86
1	Matemática Discreta	88
2	Algoritmos y Estructuras de Datos	87
3	Arquitectura de Computadoras	86
4	Análisis Matemático I	84

Una vez que el usuario ha ejecutado el módulo "elección de categoría", se generará un número aleatorio (comprendido entre 0 y la cantidad de preguntas de la categoría sorteada menos uno) que indicará el orden de la pregunta a realizar. Este valor será el identificador de la pregunta, el cual será almacenado en el arreglo de preguntas de la partida (específicamente en la posición que corresponde al número de tiro de la ruleta). Debe tenerse en cuenta que en una misma partida, no puede realizarse dos veces la misma pregunta de la misma categoría. En el caso de que el identificador sorteado para una categoría coincida con un par {categoría, pregunta} elegido previamente, se deberá volver a generar el identificador de pregunta.

Una vez elegida la pregunta, se la buscará y se la visualizará en pantalla a fin de que el usuario ingrese la respuesta que cree correcta. Luego, se continuará con este proceso hasta que el usuario actual haya respondido 10 preguntas. Al finalizar, el usuario visualizará una tabla que mostrará el nombre de la categoría (obtenido a partir del identificador almacenado en el arreglo de categorías) y el identificador de cada una de las preguntas realizadas (es decir, se visualizará el contenido de ambos arreglos).
Nota: En esta etapa, en lugar de visualizarse la pregunta con sus opciones (luego de cada tiro de la ruleta) se visualizará el identificador sorteado para la pregunta (valor almacenado en el arreglo).

Ejemplo:

- Arreglo de categorías:

4	0	3	1	4	2	1	0	4	0

- Arreglo de preguntas:

75	15	26	41	30	9	65	3	16	27

- Esto implica que el usuario responderá:
  - La 75° pregunta de la categoría 4.
  - La 15° pregunta de la categoría 0.
  - La 26° pregunta de la categoría 3.
  - La 41° pregunta de la categoría 1.
  - La 30° pregunta de la categoría 4.
  - La 9° pregunta de la categoría 2.
  - La 65° pregunta de la categoría 1.
  - La 3° pregunta de la categoría 0.
  - La 16° pregunta de la categoría 4.
  - La 27° pregunta de la categoría 0.

4. Validaciones

El juego debe solicitar en todo momento el ingreso de información de forma clara y consistente. La interacción con el usuario debe ser lo más amigable posible (uso de menús de opción, mensajes de información y/o de error específicos, limpieza de la pantalla, etc.). Se deben implementar todas las validaciones que se consideren necesarias.

Nota: En el campus de la cátedra ha quedado disponible un documento llamado "*Errores comunes en la implementación de menús*" que explica los errores que suelen cometerse al realizar este tipo de implementaciones. Se recomienda realizar una lectura de este documento previo a comenzar con el desarrollo del trabajo a fin de garantizar una correcta solución.

5. Documentos a entregar

En esta etapa, se solicita ***entregar***:

1. Una nueva versión del juego implementado en la Etapa 1 (***código C++*** y ***archivo ejecutable***) que incluya una solución a los ***puntos 1, 2, 3 y 4***.
2. Como parte del archivo ***integrantes.txt***, el nombre de usuario de SPOJ con el cual se obtuvo el aprobado para el ***proceso de validación captcha***.

~~***Fecha máxima de entrega: 26/09/2015***~~

Etapa SUPERADA

PREGUTNADOS: ETAPA DE DESARROLLO Nº 3

El objetivo de esta *tercera etapa* es mejorar las funcionalidades definidas en la etapa previa e incorporar nuevas secciones al juego.

1. Estructuras (structs)

Cada grupo deberá definir las siguientes estructuras, las cuales serán utilizadas a lo largo del desarrollo para la gestión de la información.

1.1 Estructura usuario

Esta estructura almacenará la información asociada a una cuenta de usuario.

```
struct usuario {
      int id;
      string nombre;
      string clave;
}
```

1.2 Estructura partida

Esta estructura almacenará la información asociada a una partida.

```
struct fecha {
    int anio, mes, dia, hora, mins, segs;
};

struct partida {
      long idPartida;
      int idUI, idUO; //Identificador del usuario inicial y oponente.
      int categorias[10];
      int preguntas[10];
}
```

Nota: El idPartida se conformará en base al procedimiento descrito en la sección *"**4. Generación del identificador de la partida**"*.

1.3 Estructura jugada

Esta estructura almacenará los datos que forman parte de una jugada específica dentro de una partida. Toda partida tendrá dos jugadas asociadas (lo que se representa en el campo `idPartida` de una jugada). Debe considerarse que una jugada es conjunto de resultados (true/false) que el usuario con identificador `idUsr` obtuvo en cada una de las preguntas.

```
struct jugada {
      long idPartida;
      int idUsr;
      bool rtas[10];
      int puntaje; //Se computará cuando el usuario finaliza su jugada.
}
```

1.4 Estructura pregunta

Esta estructura almacenará los datos que forman parte de una pregunta.

```
struct pregunta {
      int idPregunta;
      int idCategoria;
      int numeroGrupo;
      string pregunta;
      string opciones[4];
      int opcionCorrecta;   //Subíndice de la opción correcta.
};
```

Como se indicó en la etapa 2:

ID CATEGORÍA
0 -> FÍSICA I
1 -> MATEMÁTICA DISCRETA
2 -> ALGORITMOS Y ESTRUCTURAS DE DATOS
3 -> ARQUITECTURA
4 -> ANÁLISIS MATEMÁTICO I

Nota: La lista presentada es exhaustiva. No podrán incorporarse otras estructuras ni modificarse las solicitadas.

2. Archivos

Cada grupo deberá utilizar los siguientes archivos para mantener la persistencia de la información.

2.1 Archivo usuarios.txt

Este archivo almacena la información de las cuentas de usuario existentes en el programa. **No debe ser creado, sino que deberá utilizarse el archivo provisto junto con este enunciado**.

Además, en el archivo *manejodeusuarios.cpp* se provee un conjunto de funciones que deberán utilizarse para manipular la información del archivo. El objetivo de cada una de estas funciones se encuentra definido a modo de comentario en el mismo archivo.

bool buscarXNombreUsuario(string nombre, usuario & usr); busca en el archivo usuarios.txt un usuario a partir del nombre. Si lo encuentra retorna true y devuelve en usr los datos del usuario, sino retorna false.

bool buscarXIdUsuario(int id, usuario & usr); busca en el archivo usuarios.txt un usuario a partir de un id de usuario. Si lo encuentra retorna true y devuelve en usr los datos del usuario, sino retorna false.

bool registrarUsuario(usuario usr); Intenta registrar un nuevo usuario al sistema. Si lo puede registrar retorna true, pero si existe un usuario con el mismo nombre, o si hubo algún problema con la gestión del archivo, no pudiendo registrarlo, retorna false.

2.2 Archivo partidas.dat

Este archivo debe ser almacenado en formato binario y de acceso directo. Debe mantener la información de todas las partidas creadas utilizando la aplicación. **Debe ser creado y actualizado por el grupo.** Las partidas se almacenarán una vez creadas por medio de la incorporación de un nuevo registro del tipo `partida` al final de este archivo. Las funciones para leer/escribir esta información deberán ser desarrolladas por el grupo.

2.3 Archivo jugadas.dat

Este archivo debe ser almacenado en formato binario y de acceso directo. Debe mantener la información de las jugadas de cada uno de los participantes de las distintas partidas. **Debe ser creado y actualizado por el grupo.** Las jugadas se almacenarán una vez creadas por medio de la incorporación de un nuevo registro del tipo `jugada` al final de este archivo. Las funciones para leer/escribir esta información deberán ser desarrolladas por el grupo.

2.4 Archivo basepreguntas.txt

Este archivo almacena la información de las preguntas formuladas como parte de la *Etapa 1*. **No debe ser creado ni actualizado, sino que deberá utilizarse el archivo provisto junto con este enunciado**.
Además, en el archivo *manejodepreguntas.cpp* se entrega una función llamada *cargaDePreguntas()* que deberá utilizarse para levantar a memoria la información. Esta función lee el archivo de preguntas y lo carga en una matriz de 5 x 88 elementos del tipo `pregunta`, en la cual las filas corresponden a la categoría y las columnas a las preguntas. Así entonces el elemento (0,0) corresponde la pregunta con identificador 0 de la categoría Fisica I. Al momento de recorrer esta matriz debe recordarse que no todas las categorías constan de 88 preguntas, por lo que existirán filas de la matriz cuya cantidad de elementos sea menor a 88.

Nota: La lista presentada es exhaustiva. No podrán incorporarse otros archivos ni modificarse la definición esquematizada.

3. Observaciones

Deberá tenerse en cuenta que:

- En un momento dado, sólo un usuario puede estar activo en el programa. Este usuario será denominado *usuarioActual*.
- Todos los usuarios registrados en el programa poseen un identificador de usuario (valor entero).

- Un usuario cualquiera podrá tener, como máximo, 25 partidas iniciadas.
- Un usuario cualquiera podrá tener, como máximo, 25 partidas pendientes.
- Un usuario cualquiera podrá tener, como máximo, 50 partidas terminadas.

4. Menú principal

Para cada una de las opciones detalladas a continuación, complementar el desarrollo realizado en las etapas previas en base a las siguientes consignas:

1. *Registrarse*
   Esta funcionalidad no se ve afectada en esta nueva etapa. Los usuarios existentes en el juego son los entregados por la cátedra como parte del archivo *usuarios.txt*. Si un usuario nuevo quiere registrarse, se le permitirá hacer todo el proceso (conforme lo pedido en la *Etapa 2*).

2. *Iniciar Sesión*
   Esta opción se mantiene conforme a lo descrito en la etapa previa. Sin embargo, debe tenerse en cuenta que la validación de la cuenta de usuario deberá modificarse. Esto es, una vez que el usuario ingresa su nombre de usuario y contraseña, se deberá buscar en la lista de usuarios registrados si alguna de las cuentas coincide con el nombre de usuario ingresado y luego deberá validarse si la contraseña coincide. Para esto se deberá hacer uso de las funciones entregadas como parte del archivo *manejodeusuarios.cpp*. Una vez finalizado con éxito el inicio de sesión, este usuario pasa a ser el *usuarioActual*.

3. *Estadísticas del Juego*
   En esta opción se visualizará las siguientes estadísticas del juego:

   - Categoría que tuvo la mayor cantidad de preguntas sorteadas como parte de todas partidas jugadas hasta el momento (tanto pendientes como terminadas).
   - Porcentaje de preguntas que fueron acertadas por los dos usuarios en una partida (considerando el histórico de todas las partidas terminadas).

   Luego de esta visualización, el programa volverá al "menú principal".

4. *Cerrar aplicación*
   Esta opción se mantiene conforme a lo descrito en la etapa previa. Sin embargo debe tenerse en cuenta que, previo a la finalización de la ejecución, puede llegar a ser necesario realizar operaciones sobre algunos de los archivos.

5. Menú de juego

Para cada una de las opciones detalladas a continuación, complementar el desarrollo realizado en las etapas previas en base a las siguientes consignas:

1. *Mi cuenta*
   Esta opción se mantiene conforme a lo descrito en la etapa previa. Es decir, al ingresar a esta opción el *usuarioActual* visualizará la información asociada a su cuenta. Luego, el programa volverá al "Menú de juego".

2. *Iniciar partida*

Al ingresar a esta opción el usuario creará una nueva partida contra un usuario elegido al azar, siempre y cuando no se haya alcanzado el tope de partidas que puede iniciar un usuario.
Consideraciones:

- Toda la información que se genere durante este proceso deberá ser almacenada en una variable del tipo *partida* y una de tipo *jugada*.
- Se genera un identificador de partida válido en base al procedimiento descrito en la sección *"**4. Generación del identificador de la partida**"*. Nota: se suprime la funcionalidad donde el programa solicita un nombre de partida según pautas definidas en Etapa 2.
- A continuación, el juego elegirá aleatoriamente el *oponente*. Este usuario deberá ser elegido aleatoriamente de la lista de usuarios registrados, no pudiendo coincidir con el *usuarioActual*. En caso de que el *usuarioActual* sea el único registrado, no podrá iniciar una partida hasta que se haya registrado otro usuario en el sistema. En el caso de que el *usuarioActual* sea elegido como oponente, se sorteará nuevamente el identificador (tantas veces como sea necesario).
  Mecanismo de elección de oponente: El programa generará un número aleatorio entre 1 y la cantidad máxima de usuarios registrados en el archivo *usuarios.txt*. Este número indicará el identificador del usuario elegido. Una vez sorteado el oponente, el *usuarioActual* verá un mensaje similar al siguiente:

  `"La partida se jugará contra: oponente"`

  donde `oponente` será el nombre de usuario del oponente sorteado.
- Definido el oponente, el usuario tirará la ruleta en busca de la categoría y luego se sorteará aleatoriamente la pregunta a responder. El programa procederá a buscar la pregunta sorteada dentro de la tabla de preguntas que ha sido cargada con la función *cargaDePreguntas()*. Una vez obtenida la pregunta, visualizará aleatoriamente el orden de sus opciones por pantalla. Luego, el programa esperará a que el usuario responda. Este proceso se repetirá para cada una de las 10 preguntas de la partida.
  Nota: Tener en cuenta que la información que previamente era almacenada como parte de la ruleta haciendo uso de arreglos paralelos, ahora deberá ser almacenada como parte de la "partida".
- Todas las respuestas del *usuarioActual* deberán almacenarse en una única variable *jugada* asociada a la *partida*. Ambos usuarios responderán las mismas 10 preguntas y en el mismo orden. Una vez que el *usuarioActual* ha respondido las 10 preguntas, se computará y visualizará su resultado final (ver **4. Puntajes y ganador**) y la partida quedará pendiente (es decir, en espera) de que el oponente la juegue. En este caso, el *usuarioActual* sólo visualizará el puntaje obtenido (ya que el resultado de la partida depende de la jugada que realice el oponente).
- El *usuarioActual* deberá responder las 10 preguntas durante su jugada, no pudiendo interrumpir la constitución de la partida. Si por algún motivo el *usuarioActual* abandona la creación de la partida, se le debe advertir que toda la información de la partida y la jugada, se perderá.
- Luego de finalizada la primera jugada de la partida iniciada, se almacenará la información en los archivos correspondientes y el *usuarioActual* volverá al "Menú de juego".

Nota 1: Tener en cuenta que la información que previamente era almacenada como parte de la ruleta haciendo uso de arreglos paralelos, ahora deberá ser almacenada como parte de la "partida".

Nota 2: Tener en cuenta que mucho de lo solicitado en este punto ya se había desarrollado con anterioridad en la Etapa 2. Se recomienda volver a leer las consignas previas para tener en claro cuales son las modificaciones a implementar.

3. *Partidas iniciadas*
   Al ingresar a esta opción el usuario visualizará el listado de partidas que él ha iniciado pero que aún no han sido jugadas por su oponente. Este listado se ordenará cronológicamente según su fecha de creación (las más recientes al final) y mostrará en pantalla elementos del tipo:

   ```
   (número de partida, oponente, fecha de creación)
   ```

   Este listado deberá elaborarse haciendo uso de la información existente acerca de todas las partidas del juego iniciadas por el *usuarioActual*. Luego de la visualización del listado, el programa volverá al "Menú de juego".

4. *Partidas pendientes de juego*
   Al ingresar a esta opción el usuario visualizará las partidas en las que ha sido seleccionado como oponente. Este listado se ordenará cronológicamente según su fecha de creación (las más recientes al final) y mostrará en pantalla elementos numerados del tipo:

   ```
   (número de partida, oponente, fecha de creación).
   ```

   Del listado visualizado, el *usuarioActual* seleccionará la partida a jugar haciendo uso del número de partida de la lista. Una vez elegida la partida, el *usuarioActual* comienza el juego. La información que se genere durante esta jugada deberá ser almacenada en una variable *jugada* asociada a la *partida*. El programa mostrará las preguntas 10 sorteadas como parte de la *partida* creada por el usuario inicial, ordenando aleatoriamente sus opciones en pantalla. El *usuarioActual* no podrá abandonar la partida una vez que ha decidido jugarla (es decir, deberá forzosamente responder las 10 preguntas involucradas en la partida). Una vez que el *usuarioActual* ha respondido las 10 preguntas, se computará y visualizará el resultado final (ver *"**4. Puntajes y ganador**"*) y la *partida* quedará cerrada. La información de la *jugada* realizada se almacenará en el archivo *jugadas.dat*. Luego, el programa volverá al "Menú de juego".

5. *Partidas terminadas por el oponente*
   Al ingresar a esta opción el usuario visualizará las partidas que él ha iniciado y que los oponentes han finalizado. Este listado se ordenará cronológicamente según su fecha de finalización (las más recientes al principio) y mostrará en pantalla elementos numerados del tipo:

   ```
   (número de partida, oponente, fecha de finalización)
   ```

   Del listado visualizado, el *usuarioActual* seleccionará la partida que desea ver haciendo uso del número de lista. Una vez elegida la partida, el *usuarioActual* visualizará el resultado final (ver *"**4. Puntajes y ganador**"*). Luego, el programa volverá al "Menú de juego".

6. *Mis estadísticas*
   Al ingresar a esta opción el *usuarioActual* visualizará el siguiente conjunto de estadísticas asociadas a sus partidas:

   - Cantidad de partidas ganadas.

- Cantidad de partidas perdidas.
- Cantidad de partidas empatadas.
- Porcentaje de preguntas que fueron contestadas correctamente.

Luego, el programa volverá al "Menú de juego".

7. *Cerrar sesión*
   Esta opción se mantiene conforme a lo descrito en la etapa previa. Sin embargo debe tenerse en cuenta que, previo al cierre de sesión, puede llegar a ser necesario realizar operaciones sobre algunos de los archivos.

6. Generación del identificador de la partida

El idPartida se conformará obteniendo la fecha y hora del momento en que se genera la partida invocando la función provista cuyo prototipo es:

```
void obtenerFechaActual(fecha & f);
```

A partir de los valores de mes, dia, hora, minutos y segundos (sin considerar el año) almacenados en la fecha f se deberá llegar al id correspondiente según lo indican los siguientes ejemplos:

```
1.-  Fecha Hora obtenida: 06/11  14:07:05
     Se debe generar el siguiente id: 1106140705
2.-  Fecha Hora obtenida: 04/02  03:12:01
     Se debe generar el siguiente id:  204031201
```

En caso de que se necesite obtener la fecha y hora de generación de la partida, será suficiente decodificar el idPartida.

7. Puntajes y ganador

El puntaje de una partida se computará de la siguiente manera:

- Cada pregunta respondida correctamente vale 3 puntos.
- Es considerado ganador aquel usuario que ha sumado mayor puntaje.
- Si ambos usuarios han sacado el mismo puntaje, la partida se da como "empatada".

El cálculo de los puntajes de ambos jugadores debe realizarse una única vez (luego de que cada usuario ha jugado). Al mostrar el resultado, el *usuarioActual* visualizará:

- Su estado como jugador de la partida (ganador, perdedor, empate).
- Su puntaje.
- El puntaje del oponente.

8. Validaciones

El juego debe solicitar en todo momento el ingreso de información de forma clara y consistente. La interacción con el usuario debe ser lo más amigable posible (uso de menús de opción, mensajes de información y/o de error específicos, limpieza de la pantalla, etc.). Se deben implementar todas las validaciones que se consideren necesarias.

9. Documentos a entregar

En esta etapa, se solicita ***entregar***:

1. Una nueva versión del juego implementado en la Etapa 2 (***código C++*** y ***archivo ejecutable***) que incluya una solución a todos los ***puntos*** detallados.
2. Un documento PDF con las consideraciones particulares del equipo de trabajo sobre el enunciado del TP y su forma de resolución. Este documento será el *Análisis del Enunciado*.
3. Un documento PDF con capturas de pantallas y explicaciones sobre éstas que demuestren el comportamiento del programa con diferentes casos de prueba. Este documento será el *Documento de Casos de Prueba*.

Nota: Se ha subido al campus de la cátedra un ejemplo de *Análisis del Enunciado* y *Documento de Casos de Prueba* de trabajos de años anteriores para que sean tomados como referencia en la elaboración de los entregables solicitados.

Fecha máxima de entrega (diciembre): 19/12/2015

Fecha máxima de entrega (febrero): 10/02/2016