



TRABAJO PRÁCTICO OBLIGATORIO 2021:

Cumplir con las siguientes condiciones es parte de la evaluación del Trabajo Final:

En las clases previas a la defensa del trabajo práctico deben mostrar avances de la resolución y realizar consultas a los docentes. El día de la defensa, el programa debe funcionar: no debe arrojar errores de sintaxis/ejecución.

El trabajo final es grupal: mínimo 2 alumnos, máximo 3 alumnos.

Luego de descargar todos los archivos del trabajo práctico final publicado en PEDCO, el grupo deberá reunirse y **seguir el instructivo "TPFinal2021_GITHUB.pdf"**. El instructivo explica que cada integrante del grupo deberá crear una cuenta en GITHUB (<https://github.com/>). Uno de los integrantes del grupo deberá crear un repositorio público llamado "tateti" y agregar como colaboradores al resto de los integrantes del grupo. Cada integrante deberá tener instalado GIT en su pc (<https://git-scm.com>).

Para cumplimentar el trabajo final, cada grupo deberá entregar tres (3) archivos:

- i) Un archivo "**DisenioEstructuras<Apellidos>.pdf**" que contenga apellidos, nombres, legajos, mails, carrera de los integrantes del grupo, nombre usuario github, la url del repositorio tateti de github. Este archivo deberá realizar la representación de las estructuras de datos utilizadas y los datos almacenados en las estructuras. (*<Apellidos> debe ser reemplazado con los apellidos de los integrantes*).
- ii) El archivo "**tateti.php**"
- iii) El archivo "**programa<Apellidos>.php**". (*<Apellidos> debe ser reemplazado con los apellidos de los integrantes*).

Uno de los integrantes del grupo será el encargado de subir la resolución a una tarea en el **curso de PEDCO antes** de la defensa. Habrá otra tarea post defensa, sólo si el docente solicita modificaciones y re-entregar.



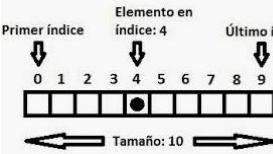





Además los archivos deben ser subidos al repositorio público "tateti" de **GITHUB**. Creado por el grupo utilizando el instructivo "TPFinal2021_GITHUB.pdf".

Consideraciones:

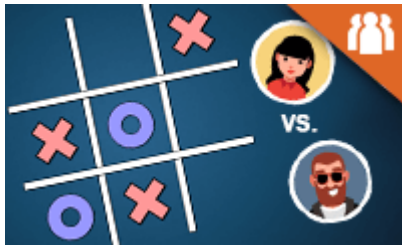
- Debe representar las estructuras de datos indicando índices/claves y valores. Esto le ayudará a identificar cómo crear, acceder y recorrer las estructuras, además de utilizar dicha representación para consultar a los docentes. Puede utilizar una planilla de cálculo o Diagrams.net en drive.
- Serán valoradas las soluciones sencillas, el código prolijo y legible.
- El trabajo será desarrollado íntegramente en PHP.
- Recuerde que es un Trabajo Final que será evaluado, utilice un vocabulario apropiado tanto en el código fuente como en los mensajes que mostrará en pantalla.
- Antes de comenzar a codificar, leer el enunciado completamente con el fin de hacer un buen análisis del problema a resolver: datos de entrada, objetivos, estructuras de datos que debe utilizar, posibles funciones a implementar, etc.
- Cada **función** implementada debe ser debidamente documentada (como fue visto en los apuntes de teoría utilizando `/** */`, especificando descripción de la función, tipo y nombre de los parámetros, tipo del retorno). Los nombres de las funciones y de los parámetros formales deben ser acorde a la funcionalidad que resuelven. Utilizar notación lowerCamelCase.

- El día de la defensa debe explicar la ejecución del programa y el código fuente, además de responder las preguntas que realizará el docente.

Resumen:

 <p>Leer atentamente todo el enunciado del trabajo práctico final.</p> <p>Para identificar todo lo que hay que hacer y establecer los pasos para resolverlo.</p>	 <p>Descargar los archivos de pedco: Enunciados y archivos php.</p> <p>Renombrar los archivos como indica el enunciado reemplazando <apellidos>.</p> <p>Juntarse en grupo para seguir la guía GIT/GITHUB:</p> <p>TPFinal2021_GITHUB.pdf</p>	 <p>Identifique las estructuras de datos diseñadas para resolver el problema, qué tipos de datos almacenan.</p>  <p>Represéntelas en papel según lo visto en la teoría, ¡Le ayudará a entender para qué se usan y cómo recorrerlas y accederlas!</p> <p>Genere el archivo “diseñoEstructuras”</p>	 <p>Relacionar el enunciado con lo programado en los archivos tateti.php y programaApellidos.php, para determinar qué está programado y qué falta. Esto le ayudará a reusar funciones y tener que codificar menos.</p> 	 <p>¡Manos a la obra! Comenzar a completar el código del archivo programaApellidos.php</p> 
---	---	---	---	--

ENUNCIADO DEL PROBLEMA A RESOLVER:



¡ JUGUEMOS AL TATETI !

El TATETI se juega entre dos (2) jugadores que eligen **X** o **O** para jugar. El Jugador que elige **X** comienza el juego.

El tablero del TATETI es una matriz de 3 Filas y 3 Columnas, es decir, un total de 9 casilleros libres, que los jugadores irán completando de a un turno por vez. El tablero vacío se puede ver en la siguiente imagen:

		COLUMNA		
		1	2	3
FILA				
1				
2				
3				

Tablero vacío

El juego consiste en colocar tres (3) símbolos en línea, y concluye cuando uno de los jugadores es ganador o bien, se completan todos los casilleros en un empate.

Las siguientes imágenes muestran ejemplos de juegos ganados por alguno de los dos jugadores:

		COLUMNA		
		1	2	3
FILA				
1		O		O
2		X	X	X
3				

Jugador X completó la fila 2

		COLUMNA		
		1	2	3
FILA				
1		O	X	X
2		O	X	
3		O		

Jugador O completó la columna 1

		COLUMNA		
		1	2	3
FILA				
1		X		
2			X	
3		O	O	X

Jugador X completó la 1er diagonal

		COLUMNA		
		1	2	3
FILA				
1		X		O
2		X	O	X
3		O		

Jugador O completó la 2da diagonal

La siguiente imagen muestra un ejemplo de empate, donde ningún jugador pudo colocar 3 símbolos en línea:

		COLUMNA		
		1	2	3
FILA				
1		X	X	O
2		O	O	X
3		X	O	X

Tablero completo con empate

En particular en esta versión del TATETI, al jugador que gane se le otorgarán 2 puntos + la cantidad de casilleros libres que logre. El jugador que pierde obtiene 0 puntos, y en caso de empate a cada jugador se le asigna 1 punto.

EXPLICACIÓN 1 (se enfoca en el punto de vista del Usuario)

En la materia de **Introducción a la Programación** especificaremos en lenguaje PHP un Programa con un menú de opciones, que permita jugar al TATETI y mostrar información de los distintos juegos que jugaron los jugadores.



Aprovechando los beneficios de la modularización, un equipo de programadores ya se adelantó y especificó una librería llamada "tateti.php" que posee las funciones necesarias para jugar un (1) juego de tateti. Además de varias funciones que podremos utilizar (reusar).

Nuestra tarea como equipo será especificar el archivo “Programa<Apellidos>.php”. Donde <Apellidos> debe ser reemplazado por los apellidos de los integrantes del equipo.

El objetivo del programa es permitir a un usuario interactuar con el siguiente menú de opciones:

Menú de opciones:

- 1) Jugar al tateti
- 2) Mostrar un juego
- 3) Mostrar el primer juego ganador
- 4) Mostrar porcentaje de Juegos ganados
- 5) Mostrar resumen de Jugador
- 6) Mostrar listado de juegos Ordenado por jugador O
- 7) salir

Cada opción del menú requiere que se invoquen una o varias funciones que serán descritas en la sección EXPLICACIÓN 3 del presente documento (también puede revisar si alguna función del archivo tateti.php puede ser utilizada).

A nivel general, de cada opción de menú el usuario que ejecuta el programa espera lo siguiente:

- 1) Jugar al tateti: se inicia un juego de tateti solicitando los nombres de los jugadores. Luego de finalizar, los datos del juego deben ser guardados en una estructura de datos de juegos (ver la sección EXPLICACION 2, de Estructura de datos del presente enunciado)
- 2) Mostrar un Juego: Se le solicita al usuario un número de juego y se muestra en pantalla con el siguiente formato:
Juego TATETI: <numero> (<empate | gano X | gano O>)
Jugador X: <nombre> obtuvo <puntaje> puntos
Jugador O: <nombre> obtuvo <puntaje> puntos

Ejemplo al visualizar el juego número 13:

```
*****
Juego TATETI: 13 (empate)
Jugador X: MAJO obtuvo 1 puntos
Jugador O: PEPE obtuvo 1 puntos
*****
```

Si el número de juego no existe, el programa deberá indicar el error y volver a pedir un número de juego válido.

- 3) Mostrar el primer juego ganador: Se le solicita al usuario un nombre de jugador y se muestra en pantalla el primer juego ganado por dicho jugador. Por ejemplo si el usuario ingresa el nombre “Majo”

```
*****
Juego TATETI: 2 (gano O)
Jugador X: JUAN obtuvo 0 puntos
Jugador O: MAJO obtuvo 5 puntos
*****
```

En caso que el jugador no ganó ningún juego, se debe indicar: “El jugador Majo no ganó ningún juego”.

- 4) Mostrar porcentaje de Juegos ganados: Se le solicita al usuario que elija uno de los símbolos (X o O), y se muestra qué porcentaje de todos los juegos ganados, el ganador es el símbolo elegido por el usuario.

Ejemplo: En total se jugaron 56 juegos de tatetí, de los cuales 16 son empates y 40 son juegos ganados (29 son ganados por X y 11 son ganados por O). Si el usuario elige el símbolo O, entonces el programa debe mostrar que O ganó el 27.50% de los juegos ganados

- 5) Mostrar resumen de Jugador: Se le solicita al usuario un nombre de jugador y se muestra en pantalla un resumen de los juegos ganados, los juegos perdidos, empates y acumulado de puntos:

```
*****
Jugador: MAJO
Ganó: 5 juegos
Perdió: 2 juegos
Empató: 0 juegos
Total de puntos acumulados: 26 puntos
*****
```

- 6) Mostrar listado de juegos Ordenado por jugador 0: Se mostrará en pantalla la estructura ordenada alfabéticamente por jugador 0, utilizando la función predefinida **uasort** de php, y la función predefinida **print_r**. (En el código fuente documentar qué hace cada una de estas funciones predefinidas de php, utilizar el manual php.net). (Este es el único menú de opciones que debe utilizar la función **print_r** para mostrar la estructura de datos)
- 7) Salir: Sale del programa.

EXPLICACIÓN 2 (Estructuras de datos)

La presente sección deberá ser utilizada para representar las estructuras de datos en el archivo "**DiseñoEstructuras<Apellidos>.pdf**" indicando los tipos de estructuras (indexado, asociativo o multidimensional), de qué tipo de datos son los índices y mostrar un ejemplo de la estructura (para representar las estructuras puede utilizar una planilla de cálculo o el software diagrams.net)

Archivo tateti.php:

Deben revisar el código del archivo **tateti.php** e identificar qué estructuras de datos se utilizan y representarlas (dibujarlas).

Archivo programaApellidos.php:

a) Utilizará una estructura indexada de arreglos asociativos que almacene información de los juegos que se jugaron. Cada arreglo asociativo tendrá el siguiente formato: ("**jugadorCruz**"=> "**nombreX**", "**jugadorCirculo**" => "**nombre0**", "**puntosCruz**"=> **ptosX**, "**puntosCirculo**" => **ptos0**)

Por ejemplo:

```
coleccionJuegos[0] = ["jugadorCruz"=> "majo", "jugadorCirculo" => "pepe", "puntosCruz"=> 5, "puntosCirculo" => 0]
coleccionJuegos[1] = ["jugadorCruz"=> "juan", "jugadorCirculo" => "majo", "puntosCruz"=> 1, "puntosCirculo" => 1]
coleccionJuegos[2] = ["jugadorCruz"=> "ana", "jugadorCirculo" => "lisa", "puntosCruz"=> 1, "puntosCirculo" => 1]
```

b) Utilizará una estructura asociativa para almacenar el resumen de un jugador que tendrá los siguientes datos: nombre, juegosGanados, juegosPerdidos, juegosEmpatados, puntosAcumulados.

EXPLICACIÓN 3 (desde el punto de vista del Programador)

Como mínimo deberán:

1. Implementar una función llamada **cargarJuegos**, que inicialice una estructura de datos con ejemplos de juegos y que retorne la colección de juegos descrita en la sección EXPLICACION 2. Mínimo debe cargar 10 Juegos donde vayan variando los jugadores y los puntajes, en algunos casos los jugadores se deben repetir.
2. Para visualizar el menú de opciones (que siempre es el mismo), implementar una función **seleccionarOpcion** que muestre las opciones del menú en la pantalla (ver sección EXPLICACION 1), le solicite al usuario una opción válida (si la opción no es válida vuelva a solicitarla en la misma función hasta que la opción sea válida), y retorne **el número** de la opción. La última opción del menú debe ser "Salir".
3. Implementar una función que solicite al usuario un número entre un rango de valores. Si el número ingresado por el usuario no es válido, la función se encarga de volver a pedirlo. La función retorna un número válido.
4. Implementar una función que dado un juego, muestre en pantalla los datos del juego como lo indica la sección EXPLICACIÓN 1.
5. Implementar una función **agregarJuego** cuya entrada en la colección de juegos y un juego, y la función retorna la colección modificada al agregarse el nuevo juego.
6. Implementar una función que dada una colección de juegos y el nombre de un jugador, retorne el índice del primer juego ganado por dicho jugador. Si el jugador no ganó ningún juego, la función debe retornar el valor -1

7. Implementar una función que dada la colección de juegos y el nombre de un jugador, retorne el resumen del jugador utilizando la estructura b) de la sección EXPLICACIÓN 2.
8. Implementar una función sin parámetros formales que solicite al usuario un símbolo X o O, y retorne el símbolo elegido. La función debe validar el dato ingresado por el usuario (Utilice funciones predefinidas de string).
9. Implementar una función que dada una colección de juegos retorne la cantidad de juegos ganados (sin importar si es X o O, es decir, algún jugador debe haber ganado, no debe haber empate.)
10. Implementar una función que dada una colección de juegos y un símbolo (X o O) retorne la cantidad de juegos ganados por el símbolo ingresado por parámetro.
11. Implementar una función sin retorno que, dada una colección de juegos, muestre la colección de juegos ordenado por el nombre del jugador cuyo símbolo es O.
12. Implementación del Programa Principal. Deberá seguir los siguientes pasos:
 - a. Precargar las estructuras de juegos.
 - b. Repetir el menú de opciones mientras la opción seleccionada no sea la opción Salir.
 - c. Cuando el usuario selecciona la opción del menú, debe invocar a la/s función/es necesarias. Salvo algunas excepciones, debe contar con funciones con parámetros formales y retorno. Asesorarse con la Cátedra para implementar las funciones correctamente de modo que los resultados de las funciones puedan ser reusados.
 - d. Investigar la instrucción **switch** en el manual de PHP. ¿a qué tipo de estructura de control vista en teoría corresponde? Escriba un comentario sobre la instrucción en el código fuente.
13. En el desarrollo utilizar funciones predefinidas por PHP (strtolower, strtoupper, strlen, etc.). En la defensa se le preguntará que funciones predefinidas utilizaron, qué hacen dichas funciones y para qué las utilizaron.

Ayuda para organizar al grupo:

- a) El grupo deberá reunirse para leer todo el enunciado y resolver el instructivo de github
- b) Luego de identificar en qué consiste todo el enunciado la cátedra les recomienda llevar un documento compartido con el siguiente listado:
<https://docs.google.com/spreadsheets/d/1urpZKJyRKZUdsEy8KMGmyYjF4B6J76LTiGzfHLNmOwU/edit?usp=sharing> para dimensionar todo lo que hay que hacer y quién se puede ir encargando de cada tema.