Trabajo Práctico Especial
Programación Imperativa
Primer Cuatrimestre 2011

1. Objetivo.

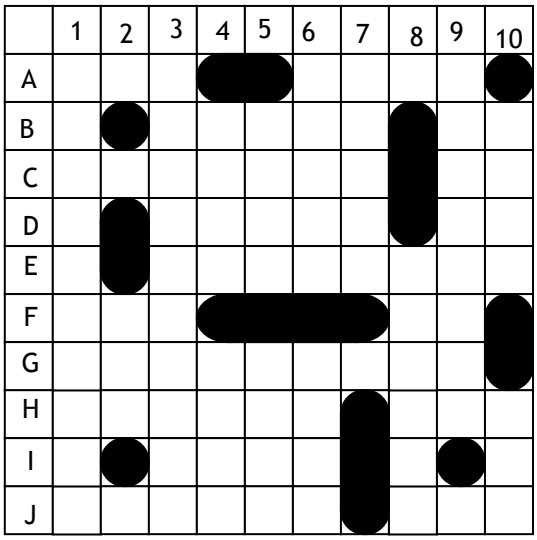
Implementar una variante del juego de "Batalla Naval" en lenguaje C, utilizando los conceptos y las técnicas de programación vistas durante el curso de Programación Imperativa, tales como asignación dinámica de memoria, modularización de programas y lectura y escritura de archivos.

2. Descripción del juego.

La "Batalla Naval" es un juego de dos participantes en el que cada uno tiene una flota de barcos que el otro intentará hundir.

En el juego clásico, cada jugador cuenta con un tablero de 10 filas y 10 columnas en el que ubica una cantidad preestablecida de barcos de tamaño también preestablecido.

Por ejemplo:



Submarinos: 4 (ocupan 1 casillero)

Destructores: 3 (ocupan 2 casilleros)

Cruceros: 2 (ocupan 3 casilleros)

Acorazados: 1 (ocupan 4 casilleros)

Antes de comenzar el juego, cada participante distribuye todos los elementos de su flota en **su tablero**, sin permitir que el oponente vea dicha distribución. Al ubicarlos, debe tener en cuenta que sólo puede hacerlo de manera horizontal o vertical, sin tocarse, es decir que siempre hay casilleros con "agua" entre dos barcos.

Una vez que cada participante ha ubicado su flota, comienza el juego propiamente dicho, el cual se desarrolla utilizando una modalidad de turnos. En el primer turno, uno de los jugadores le informa al otro en qué casillero va a realizar un ataque indicando el número de fila y columna. El adversario deberá informarle si ese casillero se encontraba ocupado por un integrante de su flota marítima o no. En caso de que ese disparo haya destruido por completo una nave, deberá informarlo exclamando la palabra "Hundido". Si, por el contrario, el disparo hubiera impactado en una nave pero no hubiera alcanzado a hundirla

(debido a que todavía quedan partes de la misma por impactar), el adversario deberá exclamar la palabra "Impacto", sin informar el tipo de la nave impactada. Finalmente, si el disparo no hubiera colisionado con ninguna nave, el oponente deberá exclamar la palabra "Agua". En cualquiera de los casos, el jugador que realizó el disparo deberá anotar el resultado del mismo en un **segundo tablero**, con el fin de contar con un registro visual de los ataques realizados y las respuestas de su oponente. El segundo turno repite la dinámica del primero, con la diferencia de que se invierten los roles de los jugadores.

El juego continua de la misma manera hasta que uno de los jugadores elimina por completo la flota de su adversario.

3. Temáticas de juego

Si bien el nombre del juego hace alusión a un combate marítimo, su conjunto de reglas puede ser utilizado sin modificaciones en el entorno de una temática distinta, tal como una de ciencia ficción similar a Star Trek. En este caso, los elementos de la flota de combate serán naves espaciales de distinto tipo, tales como destructores, navíos pequeños, etc. La temática de juego define las dimensiones del tablero y el número y tamaño de los elementos de la flota.

4. Descripción funcional del programa.

El programa deberá permitir jugar a una adaptación del juego de la Batalla Naval de acuerdo a las características anteriormente descriptas según se detalla a continuación.

4.1. Inicio del programa.

El programa deberá presentar un menú que permita elegir entre las siguientes opciones:

1. Iniciar juego nuevo (jugador humano vs computadora)
2. Recuperar un juego guardado.
3. Salir

4.2. Inicio de juego nuevo.

Una vez elegida la opción, si la misma corresponde a un **nuevo juego**, el programa mostrará una lista de las temáticas existentes y permitirá al jugador elegir una de ellas. Las temáticas disponibles se encuentran dentro del archivo ***tematicas.dat*** cuyo formato es el que se indica a continuación. En el archivo, los comentarios se indican a partir del carácter ';' y tiene vigencia sólo hasta el fin de la linea.

Formato de archivo ***tematicas.dat***.

```
Numero de tematicas: <nro>
Tematica <nro1>
Nombre: <nombretematica>
Tablero: <filas>x<columnas>
Numero de elementos de la flota: <nro>
Elemento <nro1>: <nombreElemento> – Longitud: <nro> - Cantidad: <nro>
Elemento <nro2>: <nombreElemento> – Longitud: <nro> - Cantidad: <nro>
…
Tematica <nro2>
Nombre: <nombretematica>
Tablero: <filas>x<columnas>
Numero de elementos de la flota: <nro>
```

```
Elemento <nro1>: <nombreElemento> – Longitud: <nro> - Cantidad: <nro>
Elemento <nro2>: <nombreElemento> – Longitud: <nro> - Cantidad: <nro>
…
```

Ejemplo de archivo ***tematicas.dat***.

```
; En este archivo se definen 2 temáticas de juego para ser
; usadas con las reglas del juego de la Batalla Naval.
Numero de tematicas: 2; Cantidad de temáticas del juego.
Tematica 1; Acá empieza la descripción de la primera temática
Nombre: Combate maritimo
Tablero: 10x15; El tablero es de 10 filas por 15 columnas.
Numero de elementos de la flota: 3
Elemento 1: Fragata – Longitud: 4 - Cantidad: 2; La fragata
    ; ocupa 4 casilleros
    ; y el jugador dispone de 2 de ellas.
Elemento 2: Submarino – Longitud: 1 – Cantidad: 4
Elemento 3: Destructor – Longitud: 3 – Cantidad: 3
Tematica 2; Acá empieza la descripción de
    ; la segunda temática.
Nombre: Batalla espacial
Tablero: 10x15; El tablero es de 10 filas por 15 columnas.
Numero de elementos de la flota: 3
Elemento 1: Nave Capital – Longitud: 4 - Cantidad: 1; La Nave
    ; Capital ocupa 4 casilleros y el jugador dispone de 1 de ellas.
Elemento 2: Destructor – Longitud 3 – Cantidad: 2; Existe un
    ; miembro de la flota que tiene el mismo nombre que uno
    ; usado en otra temática. Ocupa 3 casilleros.
Elemento 3: Caza – Longitud: 1 – Cantidad: 3
```

No se debe asumir que los campos del archivo se encuentren validados. Será responsabilidad del programa determinar si el archivo es correcto.

4.3. Ubicación de la flota.

Una vez obtenido a partir del archivo ***tematicas.dat*** el tamaño del tablero y los tipos y cantidad de embarcaciones a utilizar, el jugador deberá ubicar su flota en el mar.

Para ello, el programa le mostrará dos opciones:

1. ubicación interactiva.
2. ubicación aleatoria.

Si el usuario elige **ubicación interactiva**, el programa mostrará un listado de todas aquellas naves que aún no han sido posicionadas y el jugador deberá indicar dónde ubicar cada una de ellas.

Las celdas se identifican por sus coordenadas **`(fila, columna)`** donde la primer fila es la superior y la primer columna es la de la izquierda. La primer fila y la primer columna se numeran con 0.

Para elegir la ubicación, debe utilizar el comando:

set (tiponave,idnave)(filainicio, columnainicio)(filafin, columnafin)

Donde **tiponave** se refiere al tipo de elemento definido en el archivo de temáticas mientras que **idnave** se refiere al número al número de nave dentro de ese grupo. Tanto las temáticas como los identificadores se numeran a partir de 1. Las coordenadas (filainicio, columnainicio) son coordenadas de uno de los extremos de la nave mientras que las coordenadas (filafin, columnafin) son las del otro.

Por ejemplo, si se utilizara el archivo de temática mostrado en la sección anterior y se quisiera ubicar el segundo destructor con extremos en las celdas (3,4) y (3,6) escribirá

set (3,2)(3,4)(3,6)

En case de que la nave ya se encuentre posicionada en el tablero, se deberá moverla hacia la nueva ubicación.

El programa deberá validar:

- Que el tipo y el identificador de la nave sea válido.
- que la embarcación quede completamente dentro del tablero.
- que la embarcación quede siempre dispuesta en posición horizontal o vertical únicamente (nunca en diagonal)
- que no se toque con otra embarcación. (debe haber casilleros con "agua" entre medio)

En todos los casos, deberá indicarse claramente cuál es el error cometido para que el usuario pueda reingresar el comando apropiadamente.

Cada pieza que se ubica deberá mostrarse en el tablero propio con el carácter '@'. Las celdas vacías pueden mostrarse con el carácter de punto'.'.

Ejemplo de tablero propio al comienzo del juego:

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	@	@	·	·	·	·	@
1	·	@	·	·	·	·	·	·	·	·
2	·	@	·	·	@	@	·	·	·	@
3	·	@	·	·	·	·	·	·	·	@
4	·	·	·	·	·	·	·	·	·	@
5	·	·	@	@	@	@	·	@	·	·
6	·	·	·	·	·	·	·	@	·	·
7	·	@	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	·	·	@	·
9	·	·	·	@	·	·	·	·	·	·

Una vez que el jugador se encuentre conforme con la distrubución de sus naves, deberá escribir el comando **done**. El comando deberá validar que todas las naves se encuentren ubicadas en algún lugar del tablero. Si la configuración es correcta, el sistema preguntará al jugador si quiere salvar el estado actual de la partida en un archivo (permitiéndole especificar un nombre de archivo en caso de que así fuera) o si desea empezar con el juego.

Si el usuario hubiera elegido la opción de ubicación aleatoria, el programa deberá posicionar las naves de manera estocástica a lo largo del tablero, respetando las mismas reglas que aplican al posicionamiento manual. En caso de no poder generar un tablero válido (debido a que las dimensiones del mismo son demasiado pequeñas para la toda la cantidad de naves que hay que posicionar), el programa intentará a lo sumo 3 veces y luego terminará su ejecución indicando al jugador que no se pudo crear una nueva partida.

4.4. Desarrollo del juego

El programa determinará de manera aleatoria cuál de los dos participantes jugará durante el primer turno.

Cuando le toca el turno al jugador persona, se le muestran siempre dos tableros: el propio y el de registro de jugadas del oponente.

En el suyo verá, al principio del juego, las celdas ocupadas con sus naves señaladas con el símbolo '@' y las vacías con el símbolo '.' Con el correr del juego, cada vez que su oponente bombardee una parte de un nave, el simbolo '@' cambiará por 'X' si es impactada.

Ejemplo de tablero propio en mitad del juego:

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	X	X	·	·	·	·	X
1	·	@	·	·	·	·	·	·	·	·
2	·	@	·	·	@	@	·	·	·	@
3	·	@	·	·	·	·	·	·	·	@
4	·	·	·	·	·	·	·	·	·	@
5	·	·	X	@	X	@	·	@	·	·
6	·	·	·	·	·	·	·	@	·	·
7	·	@	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	·	·	@	·
9	·	·	·	@	·	·	·	·	·	·

Por otro lado, verá su registro de respuestas del oponente, que al principio del juego estará en blanco (señalado con el carácter de punto '.') y luego registrará las naves que haya impactado **('X')** o destruido ('D') y los lugares donde sabe que hay agua ('A').

Ejemplo de tablero de respuestas del oponente a mitad del juego:

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	A	X	·	·	·	·	·
1	A	A	A	·	·	·	·	·	·	·
2	D	D	A	·	A	A	·	·	·	·
3	A	A	A	·	·	·	·	·	·	·
4	·	·	·	·	·	·	·	·	·	·
5	·	·	A	A	A	X	·	·	·	·
6	·	·	·	·	·	·	·	·	·	·
7	·	A	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	·	·	·	·

En cada turno, el jugador podrá ingresar los siguientes comandos:

- **bomb (a, b)** permite bombardear la celda (a, b) del oponente.
- **undo**: volverá al estado anterior, deshaciendo la última jugada de la computadora y del jugador. En caso de repetir en forma inmediata esta acción, la segunda no tendrá efecto. Este comando no se podrá usar más de 3 veces en todo el partido.
- **save** *<filename>:* guardará el juego en un archivo de nombre *filename*.
- **quit**: saldrá del juego, ofreciendo la opción de guardar el estado actual.

Si al bombardear el jugador impacta en una nave del oponente, en su tablero de registro de respuestas del oponente deberá aparecer una **X** en la celda correspondiente.

Si al bombardear el jugador destruye una nave del oponente, en su tablero de registro de respuestas del oponente deberá aparecer una **D** en **todas las celdas** correspondientes a esa nave.

Después que juega la computadora habrá que mostrar cuál fue la jugada y si es "impacto" o "destrucción" se deberá apreciar esos cambios en el tablero del jugador persona.

Ejemplo de jugada de la computadora

>bomb (9,3)

	0	1	2	3	4	5	6	7	8	9
0	·	·	·	X	X	·	·	·	·	X
1	·	@	·	·	·	·	·	·	·	·
2	·	@	·	·	@	@	·	·	·	@
3	·	@	·	·	·	·	·	·	·	@
4	·	·	·	·	·	·	·	·	·	@
5	·	·	X	@	X	@	·	@	·	·
6	·	·	·	·	·	·	·	@	·	·
7	·	@	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	·	·	@	·
9	·	·	·	X	·	·	·	·	·	·

Por más que el jugador o la computadora hayan impactado en una nave enemiga al realizar un disparo, en el próximo turno se le pasará el control del juego al adversario.

4.5. Determinación del final de juego y del ganador

El juego termina cuando uno de los jugadores haya destruido todas las naves del oponente.

4.6. Recuperación de juego guardado.

Si el jugador decidiera **continuar una partida anterior**, el programa le pedirá que ingrese el nombre del archivo donde se encuentra almacenado el estado de una partida previa. Con los datos de dicho archivo, se carga el estado del juego para poder continuar.

El archivo debió ser guardado durante alguna partida con la opción **save *filename*.**

El archivo tiene el siguiente formato:

```
Tematica: <nro>
Proximo turno: <Jugador | Computadora>
Undos: <nro>
Datos del jugador
;Se comienza listando el estado del tablero del jugador.
Elemento 1
Miembro 1
<coordenada> -> <IMPACTADA|NO IMPACTADA>
…
Elemento 2
Miembro 1
<coordenada> -> <IMPACTADA|NO IMPACTADA>
…
Elemento n
Miembro 1
<coordenada> -> <IMPACTADA|NO IMPACTADA >
; A continuación, se lista el estado del tablero utilizado para
; marcar aquellos lugares donde el jugador ha disparado.

<coordenada> -> <DESTRUCCION | IMPACTO | AGUA>
…
; Se lista ahora el estado de los tableros de la computadora.
Datos de la computadora
Elemento 1
Miembro 1
<coordenada> -> <NO IMPACTADA|IMPACTADA>
…
Elemento 2
Miembro 1
<coordenada> -> <NO IMPACTADA|IMPACTADA>
```

```
…
Elemento n
Miembro 1
<coordenada> -> <NO IMPACTADA|IMPACTADA>
…
; A continuación, se lista el estado del tablero utilizado para
; marcar aquellos lugares donde el jugador ha disparado.
<coordenada> -> <DESTRUCCION | IMPACTO | AGUA>
..
```

Ejemplo de archivo de partida guardado ***partida***

```
; En este archivo se define el estado de una partida salvada.
Tematica: 2; Se usa la temática de "Batalla Espacial".
Proximo turno: Jugador; En este campo se indica a quién corresponde
   ; el próximo turno de juego. Los valores válidos para este campo
   ; son "Jugador" y "Computadora".
Undos: 2; Cantidad de comandos "undo" hechos por el jugador
; a lo largo de la partida.
; Se comienza listando el estado del tablero del jugador.
; Nótese que el archivo no informa cuántas naves han sido destruidas
; y cuántas naves quedan, pero es algo que puede determinarse
; fácilmente a partir de los datos provistos.
Datos del jugador
Elemento 1; A continuación, se lista la información de cada una de
    ; las partes la Nave Capital para cada una de las Naves
    ; Capitales existentes en esta temática (en este caso,
    ; una sola). Recordar que la Nave Capital ocupa 4 casilleros.
    ; Cada parte de la nave se indica si fue impactada o no.
Miembro 1
(2,3) -> IMPACTADA
(2,4) -> NO IMPACTADA
(2,5) -> NO IMPACTADA
(2,6) -> NO IMPACTADA
Elemento 2; A continuación, se lista la información de cada uno de
    ; los Destructores existentes (dos Destructores).
Miembro 1
(5,6) -> IMPACTADA
(6,6) -> IMPACTADA
(7,6) -> IMPACTADA
Miembro 2
(2,9) -> IMPACTADA
(3,9) -> IMPACTADA
```

```
(4,9) -> NO IMPACTADA
Elemento 3; A continuación, se lista la información de cada uno de
    ; los Cazas existentes (tres Cazas).
Miembro 1
(8,8) -> NO IMPACTADA
Miembro 2
(10,12) -> NO IMPACTADA
Miembro 3
(8,4) -> IMPACTADA
; A continuación, se lista el estado del tablero utilizado para
; marcar aquellos lugares donde el jugador ha disparado. El estado
; de un casillero puede ser una de las siguientes tres opciones:
; "AGUA" (indica que se disparó y que no se impactó contra nada),
; "IMPACTO" (indica que se disparó y que se impactó contra una parte
; de un navío que aún no se destruyó por completo) y "DESTRUCCION"
; (indica que en ese casillero estaba alojada una parte de un navío
; que ya fue destruido por completo).
(9,5) -> DESTRUCCION; Caza que fue destruido.
(3,6) -> IMPACTO; Parte de la Nave Capital que fue impactada.
(4,6) -> AGUA; El jugador pensaba que una parte de la Nave Capital
    ; estaba acá, pero sólo había vacío.
(4,10) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(4,11) -> AGUA; El jugador pensaba que una parte del Destructor
    ; estaba acá, pero sólo había vacío.
(3,10) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(5,10) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(2,9) -> AGUA; El jugador pensaba que una parte del Destructor
    ; estaba acá, pero sólo había vacío.
(1,9) -> AGUA; El jugador pensaba que una parte del Destructor
    ; estaba acá, pero sólo había vacío.
(9,9) -> DESTRUCCION; Caza que fue destruido.
; Se lista ahora el estado de los tableros de la computadora.
Datos de la computadora
Elemento 1; A continuación, se lista la información de cada una de
    ; las partes la Nave Capital para cada una de las Naves
    ; Capitales existentes en esta temática (en este caso,
    ; una sola). Recordar que la Nave Capital ocupa 4 casilleros.
    ; Cada parte de la nave se indica si fue impactada o no.
Miembro 1
(3,4) -> NO IMPACTADA
```

```
(3,5) -> NO IMPACTADA
(3,6) -> IMPACTADA
(3,7) -> NO IMPACTADA
Elemento 2; A continuación, se lista la información de cada uno de
    ; los Destructores existentes (dos Destructores).
Miembro 1
(6,7) -> NO IMPACTADA
(7,7) -> NO IMPACTADA
(8,7) -> NO IMPACTADA
Miembro 2
(3,10) -> IMPACTADA
(4,10) -> IMPACTADA
(5,10) -> IMPACTADA
Elemento 3; A continuación, se lista la información de cada uno de
    ; los Cazas existentes (tres Cazas).
Miembro 1
(9,9) -> IMPACTADA
Miembro 2
(11,13) -> NO IMPACTADA
Miembro 3
(9,5) -> IMPACTADA
; A continuación, se lista el estado del tablero utilizado para
; marcar aquellos lugares donde el jugador ha disparado. El estado
; de un casillero puede ser una de las siguientes tres opciones:
; “AGUA” (indica que se disparó y que no se impactó contra nada),
; “IMPACTO” (indica que se disparó y que se impactó contra una parte
; de un navío que aún no se destruyó por completo) y “DESTRUCCION”
; (indica que en ese casillero estaba alojada una parte de un navío
; que ya fue destruido por completo).
(8,4) -> DESTRUCCION; Caza que fue destruido.
(2,3) -> IMPACTO; Parte de la Nave Capital que fue impactada.
(2,2) -> AGUA; La computadora pensaba que una parte de la Nave
    ; Capital estaba acá, pero sólo había vacío.
(5,6) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(5,7) -> AGUA; La computadora pensaba que una parte del Destructor
    ; estaba acá, pero sólo había vacío.
(6,6) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(7,6) -> DESTRUCCION; Parte del Destructor que fue eliminado.
(2,9) -> IMPACTO; Parte del Destructor que fue impactado.
(1,9) -> AGUA; La computadora pensaba que una parte del Destructor
```

```
    ; estaba acá, pero sólo había vacío.
(3,9) -> IMPACTO; Parte del Destructor que fue impactado.
```

4.7. Estrategia de la computadora

Cuando la computadora elige una jugada, deberá hacerlo acorde a la siguiente estrategia:

a) Si impactó en una parte de la nave, deberá probar los siguientes disparos en los casilleros aledaños hasta destruirla por completo.
b) Si, por el contrario, ninguna parte impactada de una nave se encuentra al descubierto, deberá hacer disparos teniendo en cuenta las siguientes tres reglas:
    a. Si se destruyó una nave deberá considerar que no hay naves en las celdas adyacentes y, por ende, no deberá realizar disparos en dichas casillas.
    b. No se deberá bombardear en una zona con N casilleros libres contiguos si el barco más chico que queda por destruir es de tamaño N + 1.
    c. Si no se cumple ninguna de las dos anteriores, se elegirá una posición al azar.

Estas estrategias deberán ser respetadas en la implementación.

5. Diseño e Implementación del programa.

Se debe realizar un diseño donde se separe claramente la interfaz con el usuario (front-end) del procesamiento de los datos del juego (back-end). Esto se verá reflejado en la implementación de una biblioteca de funciones de back-end (`batallaNavalBack.c`) y otro conjunto de funciones de front-end que invocan a la biblioteca. Estas últimas incluyen a la función main y corresponden al archivo `batallaNavalFront.c`. Todos los archivos deben estar debidamente comentados y acordes a las reglas de estilo Indian Hill.

En ningún caso se debe repetir código para resolver situaciones similares, sino que debe implementarse una correcta modularización.

Tanto la biblioteca como el front-end deben estar correctamente comentados y en el caso de la biblioteca se debe escribir el archivo de encabezado correspondiente.

Importante: se debe respetar en forma estricta el formato de los archivos de temáticas y de partidas guardadas: un archivo creado por un grupo debe poder ser leído por el trabajo del resto de los grupos. Todos los trabajos serán testeados en pampero.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado.

El programa no debe tener "leaks" de memoria. Esto es, toda la memoria que se reserve en forma dinámica debe ser correctamente liberada antes de finalizar la aplicación.

Algunos consejos:

No escribir el programa entero y después probarlo todo junto, sino escribir cada función, probándola por separado. Programar defensivamente en todos los casos.

Escribir el esquema de cada función primero en papel, pudiendo utilizar pseudocódigo o lenguaje coloquial.

Una buena metodología es comenzar a escribir las funciones de más alto nivel, cableando las inferiores con cuerpo nulo o con un valor fijo, e ir reemplazándolas de a una por vez, en la medida en que al integrarlas todo siga funcionando correctamente. Esto es implementación Top-Down.

Otra metodología es comenzar a escribir las funciones desde el nivel inferior, una por vez, probando a cada una con un pequeño main que sólo la invoque a ella. Una vez escritas

y verificadas todas las funciones, unificarlas en un solo módulo. Esto es implementación Bottom-Up.

6. Material a entregar.

Cada grupo, conformado de un máximo de **2 ó 3 alumnos**, deberá entregar en sobre manila, con el nombre de los integrantes en el frente, el siguiente material:

- Impresión de todos los códigos fuente.
- Impresión del árbol de funciones (como los presentados en el TP Nro. 7)

A su vez, cada grupo deberá dejar en la carpeta trunk del repositorio de SVN asignado por la cátedra los siguientes archivos:

- batallaNavalFront.c
- batallaNavalBack.h
- batallaNavalBack.c
- makefile
- Otros archivos de código fuente (de ser necesario)
- Un archivo de temáticas de ejemplo con al menos 3 temáticas distintas.
- Archivos de ejemplo de partidas almacenadas que hagan uso del archivo de temáticas especificado en el inciso anterior.

Importante: se debe respetar en forma estricta la estructura de archivos pedida. Aquellos grupos que no la respeten recibirán una penalización en su calificación.

7. Fecha de entrega.

El trabajo debe entregarse en el repositorio de SVN provisto por la cátedra y se deberá informar el número de revisión vía e-mail a pi@it.itba.edu.ar. Para el caso de los grupos de 3 alumnos, la entrega será el viernes 27 de Mayo antes de las 19 hs. mientras que para los grupos de 2 alumnos la entrega será el viernes 3 de Junio antes de las 19 hs. Los grupos de 3 alumnos deben entregar los materiales impresos al comienzo de la clase del martes 31 de Mayo mientras que los grupos de 2 alumnos deben hacerlo al comienzo de la clase del martes 7 de Junio.

La fecha de entrega tardía (también vía e-mail indicando el número de revisión del repositorio de SVN) es el viernes 3 de Junio y el viernes 10 de Junio antes de las 19 hs. para los grupos de 3 y 2 alumnos, respectivamente. Recordar la política de entrega tardía detallada en el reglamento de la materia.

Un representante del grupo deberá enviar un e-mail a la cátedra a pi@it.itba.edu.ar antes del martes 17 de Mayo a las 19 hs. registrando la existencia del grupo y especificando los integrantes del mismo con el fin de asignarles las direcciones de los repositorios de SVN. Se considerará que aquellos alumnos que no pertenezcan a un grupo registrado han optado por no realizar el Trabajo Práctico Especial.