

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC1103 – Introducción a la Programación
2019 - 1

Interrogación 1

Instrucciones generales:

- **Para resolver la interrogación solo necesitas hasta la sección 7 del Recordatorio de contenidos adjunto.**
- La interrogación consta de 3 preguntas con igual ponderación.
- Solo se reciben consultas sobre el enunciado en los primeros 30 minutos.
- No está permitido utilizar material de apoyo adicional al recordatorio adjunto.
- Todas las preguntas están pensadas para ser resueltas en forma sencilla con lo visto en clases. No es necesario utilizar otros mecanismos más avanzados.
- Si el enunciado de la pregunta no lo indica explícitamente, no es necesario validar los ingresos del usuario. Puedes suponer que no hay errores por ingreso de datos.
- Responde cada pregunta en una hoja separada, anotando tu número de alumno y el número de pregunta en todas las hojas. Si usas más de una hoja en una pregunta, enumera las hojas de esa pregunta.
- Puedes utilizar cualquier cosa que te hayan pedido programar en un ítem anterior dentro de las preguntas (incluso si no respondiste ese ítem).
- Tienes dos horas y media para resolver la prueba desde que se entregan los enunciados.

Pregunta 1

La `morra` es un juego parecido al cachipún pero que permite jugar a 2 o más jugadores: en cada ronda, todos los jugadores ponen sus manos por delante, y al dar una señal todos muestran los dedos que quieran de ambas manos, al mismo tiempo que dicen cuántos dedos creen que serán mostrados en total por todos. El jugador que adivine el número de dedos totales mostrados en 2 rondas consecutivas es el ganador de la partida.

Para practicar como funciona el juego, decides implementarlo. El juego permite indicar la cantidad de jugadores, y luego, para cada ronda, los dedos totales que crees que saldrán y los dedos que tú muestras. La cantidad de dedos que son mostrados por cada uno de los otros jugadores son generados de forma aleatoria. El programa calcula el número total de dedos, y si acertaste o no. El juego solo puede acabar cuando el usuario aciertas dos veces consecutivas (independiente de si algún jugador virtual lo haga antes). A continuación, se muestran tres ejemplos del funcionamiento del programa.

```
========= Ejemplo nº1 =========
Cuántos jugadores? 2

Cuántos dedos saldrán? 4
TÚ Cuántos dedos? 3
JUGADOR 2: 2
Dedos totales: 5 FALLASTE
Aciertos: 0, Consecutivos: 0

Cuántos dedos saldrán? 5
TÚ Cuántos dedos? 2
JUGADOR 2: 3
Dedos totales: 5 ACERTASTE
Aciertos: 1, Consecutivos: 1

Cuántos dedos saldrán? 4
TÚ Cuántos dedos? 2
JUGADOR 2: 2
Dedos totales: 4 ACERTASTE
Aciertos: 2, Consecutivos: 2

FELICIDADES! GANASTE
Juego finalizado
```

```
========= Ejemplo nº2 =========
Cuántos jugadores? 5

Cuántos dedos saldrán? 22
TÚ Cuántos dedos? 3
JUGADOR 2: 2
JUGADOR 3: 7
JUGADOR 4: 0
JUGADOR 5: 10
Dedos totales: 22 ACERTASTE
Aciertos: 1, Consecutivos: 1

Cuántos dedos saldrán? 27
TÚ Cuántos dedos? 8
JUGADOR 2: 3
JUGADOR 3: 6
JUGADOR 4: 1
JUGADOR 5: 8
Dedos totales: 26 FALLASTE
Aciertos: 1, Consecutivos: 0

Cuántos dedos saldrán? 15
TÚ Cuántos dedos? 0
JUGADOR 2: 3
JUGADOR 3: 5
JUGADOR 4: 5
JUGADOR 5: 2
Dedos totales: 15 ACERTASTE
Aciertos: 2, Consecutivos: 1

Cuántos dedos saldrán? 18
TÚ Cuántos dedos? 2
JUGADOR 2: 7
JUGADOR 3: 5
JUGADOR 4: 1
JUGADOR 5: 3
Dedos totales: 18 ACERTASTE
Aciertos: 3, Consecutivos: 2

FELICIDADES! GANASTE
Juego finalizado
```

```
========= Ejemplo nº3 =========
Cuántos jugadores? 1
Para poder jugar deben ser al
menos 2 jugadores.
Juego finalizado
```

Estos ejemplos son sólo una guía para facilitarte el desarrollo. No debe ser exactamente igual, pero sí debe cumplir con las mismas funcionalidades.

Pregunta 2

ShopDCCorner es un emprendimiento tecnológico de alumnos de pregrado del DCC, cuyo fin es permitir que sus profesores no pierdan tiempo yendo a comprar su mercadería.

Cuando un profesor desea realizar sus compras, ingresa a la aplicación, navega por los productos, va agregando a su carro de compras aquellos que desea comprar y, una vez terminado, procede a efectuar la solicitud de compra. En ese instante, a algún DCComprador se le asigna la compra a su dispositivo móvil y va al supermercado en nombre del profesor.

ShopDCCorner exige que sus DCCompradores, antes de pasar por caja, sigan un protocolo para personalizar el servicio entregado. Este protocolo consiste en llamar al cliente siempre que haya un producto del carro que no haya sido encontrado, para reemplazarlo por otro o bien retirarlo del carro. A continuación, a la izquierda se muestran ejemplos de algunos de los productos del sistema; y a la derecha, el carro del pedido actual del profesor Edran Yaterovic.

`nombre producto`	`precio`	`nombre categoria`	`cod`
Leche chocolatada Lunco	1189	Leches	101
Leche chocolatada LecheLonco	1149	Leches	102
Leche descremada Lunco	749	Leches	103
Pack de 12 leches descremadas Lunco	8500	Cajas de leches	104
Yogurt Griego Frutilla	449	Yogurt	105
Yogurt Griego Frutilla con Trozos	479	Yogurt	106
Spaghetti 87 400g	750	Pastas	202
Tallarines 5 400g	700	Pastas	205
6 Huevos GallinaFeliz	1500	Huevos	301

`pos`	`cod`
0	101
1	101
2	103
3	202
4	202
5	202
6	301

Para ayudar a los DCCompradores en este protocolo, les han pedido ayuda para desarrollar un programa que les permita guiarlos desde su dispositivo móvil. Para facilitarles a ustedes el trabajo, les facilitan el módulo `sdcc` que deben **importar** que posee algunas funciones implementadas que pueden utilizar para interactuar con los productos y el carro:

- `precio(cod)`: recibe un *int* que corresponde al código de barra de un producto, y retorna el precio como *int*. Ej: hacer el llamado `precio(101)` retorna `1189`.
- `categoría(cod)`: recibe un *int* que corresponde al código de barra de un producto, y retorna un *str* con el nombre de la categoría del producto. Ej: hacer el llamado `categoria(101)` retorna `"Leches"`
- `tamano()`: retorna la cantidad de productos que el usuario que está siendo atendido tiene en su carro. Ej: En el ejemplo de arriba, esta función retorna `7`.
- `codigo(pos)`: recibe un *int*, y retorna el código de barra del producto que se encuentra en la posición `pos` del carro. Considera que el primer producto del carro se encontrará en la posición 0, y el último en una posición que corresponde a la cantidad de productos en el carro -1. Ej: En el ejemplo de arriba, `codigo(2)` retorna `103`
- `alternativas(cod)`: recibe un código de barra, y retorna un *str* listo para imprimir en consola con todos los productos y sus códigos de barra por los cuales el producto `cod` puede ser reemplazado. Ej: Hacer el llamado `alternativas(101)` retorna:
  `"(102) Leche chocolatada LecheLonco`
  `(103) Leche descremada Lunco"`.
- `eliminar(pos)`: actualiza el carro de compras eliminando el producto que está en la posición `pos` del carro.
- `reemplazar(cod_viejo, cod_nuevo)`: actualiza el carro de compras, sustituyendo cada aparición del producto de código de barras `cod_viejo` por el producto de código de barras `cod_nuevo` en el carro.

Para completar el software, se te ha pedido que implementes las siguientes funciones:

a) `(5 pts) costo_envio()`: Debe retornar el costo de envío de la compra. Este corresponde a 1400 pesos si hay menos de 10 productos en el carro, a 700 pesos si hay menos de 20 productos, y es completamente gratuito si hay 20 o más productos. En el ejemplo del carro de Edran Yaterovic, el costo de envío sería 1400 pesos, ya que hay 7 elementos (<10).

b) (5 pts) `costo_total()`: Debe retornar el costo total a cargar al profesor. Este considera el precio total de todos los productos en el carro de compra en este momento, sumado al costo de envío. En el ejemplo, el costo total sería 1189 + 1189 + 749 + 750 + 750 + 750 + 1500 + 1400 = 8277.

c) (10 pts) `costo_por_categoria(cat)` : recibe un *str* con el nombre de una categoría, y la función debe retornar el costo sumado de todos los productos que pertenezcan a la categoría recibida. En el ejemplo, se tiene que `costo_por_categoria("Pastas")` debería retornar `2250`, al ser el total de sumar 750 + 750 + 750.

d) (15 pts) `cantidad_de_bolsas()`: Debe retornarle al DCComprador la cantidad de bolsas que debe usar (todas ecológicas, por supuesto). Cada bolsa permite guardar hasta 8 productos (no considere factores de volumen o peso). Sin embargo, considera que si en algún producto de los del carro está la categoría `"Huevos"`, se requiere una bolsa adicional única para estos productos (independiente de la cantidad); y que los productos cuya categoría es `"Cajas de leches"` no utilizan bolsa. En el ejemplo, esta función retornaría `2`, pues se requiere esa cantidad de bolsas: una para los primeros 6 productos y una extra para los huevos.

e) (25 pts) Finalmente, se te pide implementar la parte del código principal de la aplicación que guía la conversación del DCComprador con el cliente en caso de que hayan productos que no se encuentren en el supermercado. Este debe recibir como input un código de barra, y primero debe imprimir en consola las alternativas del producto con dicho código de barra. Luego, debe indicarle al DCComprador que pregunte al cliente si desea eliminarlo o cambiarlo. Si desea cambiarlo, el DCComprador debe ingresar el código de barra del producto a ser reemplazado por el inexistente, y el programa debe realizar el reemplazo. Por otro lado, si no desea cambiarlo por ninguna de las alternativas, entonces el producto `cod` debe ser eliminado del carro. Finalmente, debe imprimir el costo total de la compra a cargar al profesor, y la cantidad de bolsas que requerirá para trasladar la compra. Puedes asumir que todos los inputs recibidos por el usuario serán correctos.

A continuación, se muestra un diálogo de ejemplo de la parte e). Suponga que al comienzo del diálogo, el carro de la compra es el del profesor Edran Yaterovic mostrado anteriormente. En este diálogo, el producto de código 103 no estaba, y el profesor quiso cambiarlo por el producto 102. Por lo tanto, el costo final será 1189 + 1189 + **1149** + 750 + 750 + 750 + 1500 + 1400 = 8677.

```
============== Ejemplo parte e) ==============
¿Hay algún producto que no esté en el
supermercado?
1) Si
2) No
1

Ingrese código de producto que no está en
el supermercado:
103

Las alternativas para dicho producto son:
(101) Leche chocolatada Lunco
(102) Leche chocolatada LecheLonco


Desea cambiar el producto?
1) Si
2) No
1

¿Cuál producto desea elegir como reemplazo?
102
Producto reemplazado exitosamente.

¿Hay algún producto que no esté en el
supermercado?
1) Si
2) No
2

Compra finalizada
El costo total de la venta es de $8677
y debe llevar 2 bolsas
```

Pregunta 3

El director del Departamento de Ciencias de la Computación, Edran Yaterovic, está contento porque en unos años más cumplirá 64 años, que es un edad `BITY`. Las edades BITY, muy importantes para los computines, son aquellas que son potencia de 2. Por ejemplo, las primeras edades BITY son 1 (2^0), 2 (2^1), 4 (2^2), 8 (2^3), 16 (2^4), 32 (2^5), 64 (2^6), etc. Crea dos funciones, una **iterativa** y otra **recursiva** que hagan lo mismo: dada la edad de una persona(que puedes asumir mayor o igual a 0), retorne la siguiente edad BITY que va a cumplir. Si ya tiene una edad BITY, retornaría esa edad. En concreto:

a) `(15 pts) bit_it(n)`: recibe (como mínimo) la edad de una persona como un *int*, y retorna un *int* con la próxima edad BITY. Esta función debe ser **iterativa**. La función puede recibir otros parámetros si lo estimas necesario.

b) `(20 pts) bit_rec(n)`: recibe (como mínimo) la edad de una persona como un *int*, y retorna un *int* con la próxima edad BITY. Esta función debe ser **recursiva**. La función puede recibir otros parámetros si lo estimas necesario.

A continuación, se muestran algunos ejemplos de los parámetros y salida retornadas por las funciones. Si bien es (bastante) difícil que una persona llegue a los 1468 años, la función debe funcionar para cualquier entero a partir de 0.

`parámetro`	`retorna`	`explicación`
5	8	corresponde a 8 (2^3)
7	8	corresponde a 8 (2^3)
9	16	corresponde a 16 (2^4)
32	32	corresponde ya a 32 (2^4)
1468	2048	corresponde a 2048 (2^{10})

c) `(10 pts) bit_comp(n1, n2)`: recibe dos *int* con las edades de dos personas, y devuelve un *bool* indicando si las siguientes edades BITY de las dos personas son la misma (`True`) o no (`False`). Esta función debe llamar **obligatoriamente** a una de las dos funciones anteriores. Puedes asumir que las funcionas ya fueron implementadas correctamente.

A continuación, se muestran ejemplos de los valores de entrada y valores retornados por la función

`n1`	`n2`	`retorna`	`explicación`
5	7	*True*	Ambos son 8 (2^3)
7	9	*False*	Uno es 8 (2^3) y el otro 16 (2^4)
32	18	*True*	Ambos son 32 (2^5)
1468	2040	*True*	Ambos son 2048 (2^{11})

d) `(15 pts) aniversario(n, a)`: recibe dos *int*: el primero, el número de años de una organización y el segundo, el número de nuevos aniversarios BITY que están interesados en celebrar; y devuelve un *int* con el último año BITY de las celebraciones. Por ejemplo, la organización DCCelebraciones lleva 258 años de funcionamiento (su próximo aniversario BITY será cuando cumplan 512 años de funcionamiento), y quieren saber cuántos años tendrán en 10 aniversarios BITY más (increíble pero cierto: será cuando cumplan 262.144 años). Por lo tanto, `aniversario(258, 10)` retornará `262144` (y `aniversario(512, 10)` retornaría `524288`). Esta función debe llamar **obligatoriamente** a al menos una de las funciones anteriores. Puedes asumir que éstas ya están implementadas correctamente.