

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

IIC1103 – Introducción a la Programación
2019 - 1

Interrogación 2

Instrucciones generales:

- **Para resolver la interrogación solo necesitas hasta la sección 10 del Recordatorio 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

¡S_ n_s p_rd__r_n l_s v_c_l_s! (Es decir, ¡Se nos perdieron las vocales!). Afortunadamente, encontramos algunas, pero están desordenadas: vienen en un *string* en el siguiente formato: `"o4e1e26o14a18e12"`; (es decir, cada letra seguida de su posición en el *string* original). En esta ocasión, debes escribir la función `reconstruir(texto, vocales)` que reciba el *string* `texto` sin vocales, y el *string* `vocales` con las vocales que encontraste, y retorne el *string* reconstruido. Si hay alguna vocal que no está en el *string* `vocales`, estas se deben dejar como `"_"` (es decir, con un guión bajo, como venía originalmente).

Puedes asumir que las posiciones que acompañanan a las vocales corresponderán siempre a caracteres `"_"` en el texto original. Es decir, si en `vocales` aparece `"o4e1"`, entonces en las posiciones de índice 1 y 4 de la variable `texto` original serán `"_"`. Además de esto, pueden asumir que todos los caracteres de `vocales` serán en minúscula y sin tilde. Respecto a la cantidad de dígitos que tendrán los índices de las vocales, estos pueden ser tan grandes como se quiera (ej: puede haber una vocal que haya que ponerla en la posición 345992346034 del texto), por lo que no puedes asumir la cantidad de dígitos que tiene cada índice.

```
reconstruir("H_l_ M_nd_", "o1a3u6o9") retorna "Hola Mundo"

reconstruir("H_l_ M_nd_", "") retorna "H_l_ M_nd_"

reconstruir("H_l_ M_nd_", "a3o9") retorna "H_la M_ndo"

reconstruir("S_ n_s p_rd__r_n l_s v_c_l_s", "o4e1e26o14a18e12") retorna "Se nos p_rd_eron las v_c_les"
```

Pregunta 2

Te han pedido que implementes un programa para permitir a una cadena hotelera administrar sus distintos hoteles. Para esto, te dan las siguientes especificaciones.

La cadena posee varios hoteles, y cada hotel posee una cierta cantidad de pisos (la cantidad de pisos varía en cada hotel), y cada piso contiene piezas que se pueden arrendar a los clientes. Además, cada pieza tiene un precio por noche. Desde el punto de vista de la administración, las personas solo poseen nombre y edad. Lo que se te ha pedido a ti, es que implementes un programa que entregue las siguientes funcionalidades:

- Imprimir los nombres y edades de los ocupantes de una pieza en específico en este momento.
- La ganancia de un piso en específico en este momento.
- Desocupar una pieza.
- Ocupar una pieza. Para esto, debes registrar todos los nombres y edades de los nuevos huéspedes.
- Imprimir todas las piezas que están habitadas actualmente

Para facilitar el desarrollo, se te entregó un método y algunas de las clases necesarias para el programa, por lo que **sólo debes implementar el código principal y la(s) clase(s) que no esté(n) en el código que te entregan** (**Hint:** No está la clase `Pieza`)

```
class Huesped:
    def __init__(self, n, e):
        self.nombre = n
        self.edad = e

class Piso:
    def __init__(self, numero):
        self.numero = numero
        self.piezas = []

class Hotel:
    def __init__(self, nombre):
        self.nombre = nombre
        self.pisos = []

def get_menu():
    menu = "¿Qué deseas hacer?\n[1] Mostrar ocupantes de una pieza\n[2] Mostrar ganancias de un piso\n"
    menu += "[3] Desocupar pieza\n[4] Ingresar nuevos huéspedes\n"
    menu += "[5] Mostrar todas las piezas ocupadas\n[6] Salir"
    return menu
```

Si lo estimas conveniente, puedes implementar nuevas clases y agregar métodos a las clases ya implementadas. Para desarrollar el programa, puedes asumir que ya existe un hotel creado con todos sus pisos y piezas correspondientes. La instancia de este hotel se encuentra en una variable que se llama `h`. Además de esto, al querer ocupar una pieza para atender nuevos clientes, la asignación debe ser automática dando prioridad a las habitaciones que se encuentran en los primeros pisos (ej: Si llega un cliente a una pieza, y está desocupada la 103, la 205 y la 301, se le asigna la 103 porque pertenece al piso de más abajo). A continuación, se muestra un diálogo de ejemplo. Los caracteres en negrita son los datos que son ingresados por el usuario

```
¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
1

¿Qué pieza quiere mostrar?
203
Pedro, 43 años
Francisca, 42 años
Juan, 10 años

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
2

¿Las ganancias de qué piso quiere mostrar?
2
El piso 2 está ganando $350000 pesos por noche

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
3

¿Qué pieza quiere desocupar?
103
Pieza 103 desocupada

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
4
¿Cuántos huéspedes quiere agregar?
2
Nombre huésped 1: Juan
Edad Huésped 1: 26
Nombre huésped 2: Rosario
Edad Huésped 2: 23
Usuarios ingresados en pieza 103.

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
5

Las piezas ocupadas son 103, 104, 105, 201, 302, 303

¿Qué deseas hacer?
[1] Mostrar ocupantes de una pieza
[2] Mostrar ganancias de un piso
[3] Desocupar pieza
[4] Ingresar nuevos huéspedes
[5] Mostrar todas las piezas ocupadas
[6] Salir
6
```

Considera que las ganancias se generan solamente por las piezas que están siendo ocupadas por algún huésped. Por simplicidad, puedes asumir que siempre habrá al menos una habitación ocupada, y que siempre habrá una habitación vacía para los nuevos huéspedes ingresados.

Pregunta 3

Con todos los compromisos sociales, carretes, y encuentros para estudiar, organizar tu fin de semana es un infierno. Así es que decides crear tu propio sistema para gestionarlo. Para ello, decides tener un sistema de bloques para agendar actividades. El fin de semana constará de 24 bloques numerados del 0 al 23, y cada bloque tiene una duración de dos horas. Considerando esto, se concluye que el sábado tiene los bloques del 0 al 11, y el domingo tiene los bloques del 12 al 23. Para llevar un bloque a su respectiva hora y día, se tiene que el bloque 0 es el sábado entre 00:00 y 1:59 hrs, bloque 1 es el sábado entre 02:00 y 3:59 hrs, y así sucesivamente, luego el bloque 12 es el domingo entre 00:00 y 1:59, el bloque 13 es el domingo entre 02:00 y 3:59, y así sucesivamente.

Se te pide implementar un programa utilizando listas que permita hacer lo siguiente:

a) Cree un nuevo fin de semana vacío.

b) Pregunte por la cantidad de compromisos que se quieren incluir.

c) Agregue cada uno de los compromisos a los bloques correspondientes si es que es posible.

d) Cada vez que se intente agregar una actividad, se imprima el calendario para cada uno de los dos días por separado.

e) Al final debe mostrar el itinerario final del día sábado y domingo por separado, y una lista de todas las actividades que no pudieron ser agregadas.

Para esto, al momento de pedir los compromisos, se le debe pedir al usuario el nombre del compromiso, en qué bloque comienza y la duración de dicho compromiso en bloques. Un compromiso no cabe en el itinerario si es que cualquiera de los bloques necesarios para realizar dicho compromiso ya está ocupado por otra actividad ingresada anteriormente. Los bloques de un compromiso deben ser continuos y no tener saltos en el tiempo, por lo que si se tiene un compromiso que comienza en el bloque 12 y dura 4 bloques, para poder agendarlo deben estar disponible los bloques 12, 13, 14 y 15. A continuación, se muestra un ejemplo del flujo del programa. Los caracteres en negrita son los datos que son ingresados por el usuario:

```
¿Cuántos compromisos desea intentar agregar? 7
Nombre compromiso 1: Ir al cine
Duración compromiso 1: 2
Bloque de inicio compromiso 1: 8
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "", ""]
dgo = ["", "", "", "", "", "", "", "", "", "", "", ""]

Nombre compromiso 2: Dormir
Duración compromiso 2: 7
Bloque de inicio compromiso 2: 10
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "", "", "", "", "", ""]

Nombre compromiso 3: Almuerzo familiar
Duración compromiso 3: 1
Bloque de inicio compromiso 3: 18
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar", "", "", "", "", ""]

Nombre compromiso 4: Salir con amigos
Duración compromiso 4: 2
Bloque de inicio compromiso 4: 9
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar", "", "", "", "", ""]

Nombre compromiso 5: Ver documental de Game of Thrones
Duración compromiso 5: 1
Bloque de inicio compromiso 1: 22
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
```

```
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"", "", "", "Ver documental de Game of Thrones", ""]

Nombre compromiso 6: Estudiar para IIC1103
Duración compromiso 6: 2
Bloque de inicio compromiso 6: 19
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"Estudiar para IIC1103", "Estudiar para IIC1103", "",  "Ver documental de Game of Thrones", ""]

Nombre compromiso 7: Descansar
Duración compromiso 7: 6
Bloque de inicio compromiso 7: 23
sab = ["", "", "", "", "", "", "", "", "Ir al cine", "Ir al cine", "Dormir", "Dormir"]
dgo = ["Dormir", "Dormir", "Dormir", "Dormir", "Dormir", "", "Almuerzo familiar",
"Estudiar para IIC1103", "Estudiar para IIC1103", "",  "Ver documental de Game of Thrones", ""]

Los siguientes compromisos no podrán realizarse:
- Salir con amigos
- Descansar

Tu programación del fin de semana es:
Sabado:
00 hrs: ""
02 hrs: ""
04 hrs: ""
06 hrs: ""
08 hrs: ""
10 hrs: ""
12 hrs: ""
14 hrs: ""
16 hrs: "Ir al cine"
18 hrs: "Ir al cine"
20 hrs: "Dormir"
22 hrs: "Dormir"

Domingo:
00 hrs: "Dormir"
02 hrs: "Dormir"
04 hrs: "Dormir"
06 hrs: "Dormir"
08 hrs: "Dormir"
10 hrs: ""
12 hrs: "Almuerzo familiar"
14 hrs: "Estudiar para IIC1103"
16 hrs: "Estudiar para IIC1103"
18 hrs: ""
20 hrs: "Ver documental de Game of Thrones"
22 hrs: ""
```

Nota que las actividades se pueden realizar ocupando bloques del sábado y domingo (en el ejemplo, dormir), pero no se pueden agendar desde domingo hasta el lunes (en el ejemplo, descansar). Por otra parte, nota que luego de cada intento de ingreso, se imprime una lista con el itinerario del sábado y otra con el itinerario del domingo, por separado.