

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
Segundo Semestre de 2014

IIC 1103 - Introducción a la Programación

Examen ⋆ Tiempo: 180 minutos ⋆ Sin apuntes ⋆ Sin consultas

1. Para cada una de las siguientes funciones, explica con tus propias palabras, en no más de tres líneas, qué propiedad(es) debe cumplir el parámetro `n` para que dicha función retorne `True`. Ilustra cada respuesta con dos ejemplos, uno que retorne `True` y uno que retorne `False` (esto no se considera parte de las tres líneas). Puedes suponer que `n` es un número natural.

   a)
   ```
   def misterio1(n):
       a = 2
       while a <= n:
           if a + 1 == n:
               return True
           a = a * 2

       return False
   ```

   b)
   ```
   def misterio2(n):
       a = 0
       for i in range(1, n // 2 + 1):
           if n % i == 0:
               a += i

       if n != a:
           return False

       return True
   ```

2. Dada una lista de números enteros $S_0, S_1, \ldots, S_N$, se te pide escribir una función que reciba como parámetro una lista de números y que retorne el valor máximo de

$$\sum_{k=i}^{k=j} S_k$$

   es decir, el valor de la suma de aquella sub-lista (consecutiva) que sume más. Por ejemplo, si la lista es [–2, 11, –4, 13, –5, –2], entonces la respuesta es 20 (la sub-lista [11, –4, 13]).

3. El Gimnasio "Full Sport" define sus precios en base a 4 categorías de servicios: *básico*, *premium*, *super premium* y *complementario* (gratis para todos los clientes). Cada cliente tiene una tarjeta donde carga un monto a gastar y desde donde se le descuentan los servicios que utiliza de acuerdo al precio del servicio y ciertos descuentos. Estos descuentos dependen de la categoría del cliente al que pertenece (descuento porcentual) y de la edad del cliente (descuento bruto, que se aplica luego del descuento porcentual). Si un cliente no tiene suficiente carga en su tarjeta, no se le realiza el cobro, ya que no se le permite usar el servicio.

   Se te pide escribir el código correspondiente a 6 funciones que facilitarán realizar los cobros.

Cuadro 1: Valores de servicios

Servicio	**Valor**
básico	$ 10.000
premium	$ 25.000
super premium	$ 45.000
complementario	$ 0

Cuadro 2: Descuentos porcentuales

Categoría	**Descuento**
400	50 %
300	25 %
200	10 %
otros	0 %

Cuadro 3: Descuentos por edad

Edad	**Descuento**
Menor a 18 o 60 o más	$ 5.000
Entre 18 y 30	$ 1.000
otros	$ 0

Código a completar:

```python
# Retorna el valor base de un servicio
def valor_servicio(tipo_servicio):

# Retorna el descuento proporcional por categoría.
# Por ejemplo si el descuento es del 25% debe retornar 0.25
# La categoria viene dada como string.
def descuento_proporcional_por_categoria(categoria):

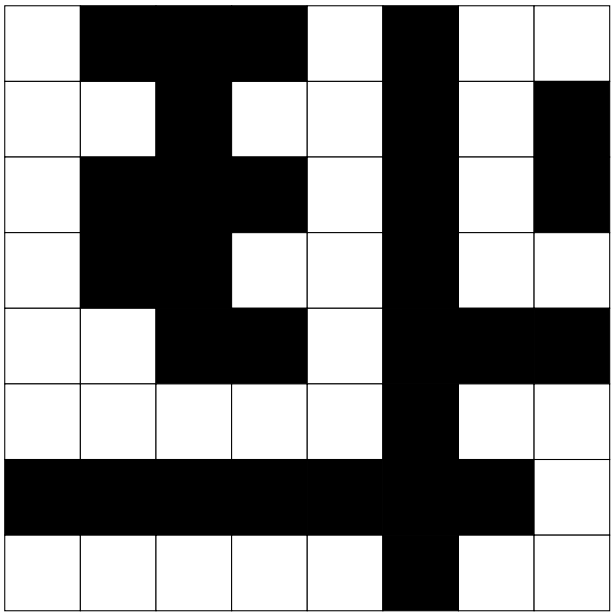
# Retorna el descuento bruto por edad
def descuento_bruto_por_edad(edad):

# Retorna el valor de cobro por servicio incluyendo descuentos
def valor_cobro(tipo_servicio, edad, categoria):

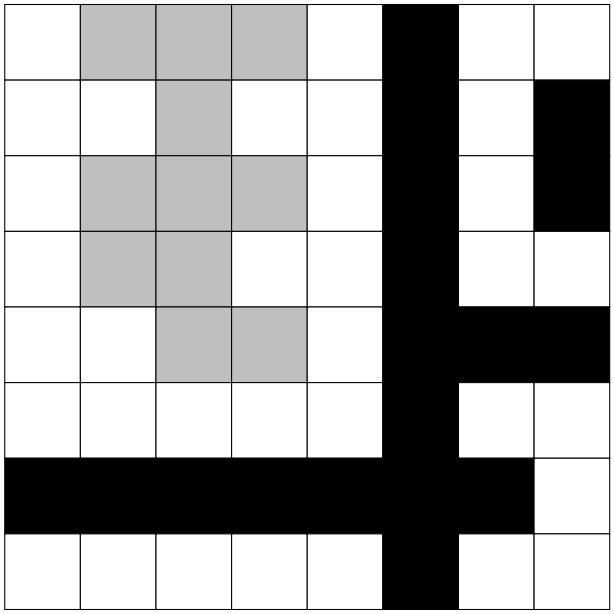
# Realiza cobro según tipo de servicio, y retorna el saldo restante en la tarjeta
# Aplica los descuentos correspondientes y despliega mensaje indicando
# cuánto cobró a la persona identificada por su nombre
def realizar_cobro(tipo_servicio, nombre_cliente, saldo, edad, categoria):

# Despliega mensaje indicando cuántas veces puede consumir una persona
# (identificada por su nombre) cada uno de los servicios no gratuitos
# Por ejemplo, una persona con saldo 50.000 de 20 años y de categoria "200"
# podría consumir 6 veces el servicio básico, 2 veces el premium
# y 1 vez el super premium
def consumos_posibles(nombre_cliente, saldo, edad, categoria):
```

4. Tienes un tablero de n x n celdas blancas y negras, distribuidas de cualquier manera. Una región blanca es un conjunto contiguo máximo de celdas blancas: dos celdas blancas son contiguas si tienen un borde común (si sólo tienen un vértice común, no son contiguas). Una región negra se define similarmente. Por ejemplo, en el tablero de 8 x 8 que se muestra en la figura hay 4 regiones blancas y 3 regiones negras.



El problema consiste en que dada una celda y un color, se pide pintar de ese color toda la región a la que pertenece la celda. Por ejemplo, si para el tablero de la figura anterior se especifica la celda (3, 2) y el color gris, el resultado debe ser el que se muestra a continuación.



Para ello, implementa la función `pintar_region(...)`, que reciba los parámetros apropiados.

Hint: Probablemente esta función incluye cuatro llamadas recursivas.

5. El ministro Gómez Lobo te ha pedido que implementes una simulación para determinar la necesidad de buses en los recorridos del Trasantiago. Cada recorrido tiene una cantidad determinada de paraderos; la separación entre paraderos es siempre la misma para un recorrido; y la velocidad de las micros fluctúa entre los 100 y 900 metros por minuto, y cambia minuto a minuto. Interesa saber la cantidad máxima y el promedio de micros en un recorrido dado, en un intervalo de un día.

   Para simularlo debes implementar dos clases: *Micro* y *Recorrido*. La clase *Micro* tiene un atributo *velocidad*, un atributo *último_paradero* que representa el último paradero visitado, y un atributo *distancia_del_paradero* que corresponde a los metros recorridos después de pasar por el *último_paradero*. Por ejemplo, si la micro está 100 metros más adelante del paradero 3, entonces *último_paradero* es igual a 3 y *distancia_del_paradero* es igual a 100. Si la micro está justo en el paradero 20, entonces *último_paradero* es igual a 20 y *distancia_del_paradero* es igual a 0.

   La clase *Recorrido* tiene como atributos *número_de_paraderos*, *distancia_entre_paraderos* y una lista *micros* que contiene las micros en el recorrido.

   a) Implementa los constructores de ambas clases. El de la clase *Micro* debe asignar a las micros una velocidad al azar entre 100 y 900 metros (considera solo números enteros) y comenzar justo en el paradero 1. El de *Recorrido* debe comenzar sin micros y recibir el valor de sus otros atributos como parámetro.

   b) Para la clase *Micro* debes implementar un método que recalcula la velocidad de la micro como un entero al azar entre 100 y 900.

   c) Para la clase *Recorrido* deberás implementar tres métodos.

   - Un método *agregar_micro* que agregue una micro en el primer paradero del recorrido.
   - Un método *avanzar_micro* que recibe una micro m y ajusta la posición de la micro m de acuerdo a lo que avanza en un minuto, considerando la posición actual, la velocidad y considerando además que cuando una micro visita un paradero no continúa avanzando ese minuto. Más específicamente, supón que la micro está ℓ metros más adelante del paradero p y que v es la velocidad actual. Si $\ell + v$ es mayor a la distancia entre paraderos, entonces la nueva posición de la micro es justo en el paradero $p + 1$. De lo contrario, se ubicará a $\ell + v$ metros más adelante del paradero p.
   - Un método *simular_minuto*, que se encargue de simular un minuto en la simulación: debe avanzar todas las micros, eliminar las que llegan al último paradero y, finalmente, recalcular la velocidad de cada micro para el próximo minuto. Recuerda que puedes eliminar un elemento `e` de una lista `lista` usando `lista.remove(e)`.

   d) Implementa una función *simulación* que reciba como parámetros un número n de paraderos, la distancia d entre paraderos y la cantidad *mins* de minutos que transcurren entre la salida de una micro y la próxima. Esta función debe imprimir el máximo y el promedio de la cantidad de micros que hubo en el recorrido. Cada iteración corresponde a un minuto, y deberás simular por 12 horas. Inicialmente el recorrido comienza con una micro en el primer paradero y se genera una nueva micro en ese paradero cada *mins* minutos.