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

IIC1103 – Introducción a la Programación

Examen

Instrucciones

- Es muy importante que recuerdes que está estrictamente prohibido: conversar con el compañero, **utilizar aparatos electrónicos** (celulares, tablets, ipads, etc.). La copia se sancionará con reprobación inmediata del curso con nota 1.1, sanción estipulada en la Política de Integridad Académica.
- Durante la evaluación NO está permitido utilizar apuntes.
- Los ejercicios deben ser respondidos en HOJAS SEPARADAS. No se aceptarán hojas de respuesta que contengan respuestas a dos ejercicios distintos.
- Puedes utilizar más de una hoja para responder una misma pregunta, pero tu nombre y el número de pregunta deben estar marcados correctamente en TODAS LAS HOJAS.
- El tiempo disponible para realizar la evaluación es estrictamente de 3 horas. No se dará tiempo adicional.
- No se podrán hacer preguntas a profesores o ayudantes.

Pregunta 1 (10 puntos)

Responde cada una de las siguientes preguntas (5 líneas de respuesta como máximo por pregunta):

a) Una receta de cocina, ¿es un algoritmo? Justifique.
`Respuesta:` Sí. La receta de cocina al igual que un algoritmo es una secuencia ordenada de instrucciones.

b) Se tiene la siguiente lista:

```
fib = [1,1,2,3,5]
```

- Escriba un trozo de código que muestre los elementos de la lista **fib** utilizando un for.
`Respuesta:`

```
for i in fib:
    print(i)
```

- Escriba un trozo de código que muestre los elementos de la lista **fib** utilizando un while.
`Respuesta:`

```
j=0;
while j < len(fib):
    print(fib[j])
    j+=1
```

c) ¿Qué imprime el siguiente código?

```
a = 10
b = 20
c = a > b
d = 0
e = 0

if c == (b%2 != 0):
   d = (a+b) // 10
   if d != 0 and not c:
      print(d)
   while d > 0:
      print("*")
else:
   e = (a+b) / 10
   if e > 3 and d > 0:
      print(e+d)
```

`Respuesta:` Imprime en pantalla `*` infinitas veces por que la condición del control de flujo `while` es siempre verdadera.

d) El próximo semestre, un/a profesor/a de IIC1103 dictará el curso IIC2113. Para ahorrarse trabajo, en el programa del curso está reemplazando la sigla de Introducción a la Programación por la sigla del curso nuevo, utilizando el código a continuación. ¿Qué imprimen ambos códigos?.

```
# Codigo 1
cadena = ['I','I','C','1','1','0','3']
print(cadena[0:3])
cadena[3] = '2'
cadena[5] = '1'
print(cadena)


# Codigo 2
```

```
cadena = "IIC1103"
print(cadena[0:3])
cadena[3] = '2'
cadena[5] = '1'
print(cadena)
```

Respuesta: El primer segmento de código imprime `IIC2113`, el segundo da error al momento de cambiar un elemento del string.

Pregunta 2 (15 puntos)

Para hacer rankings de películas la cadena de cines *PeliPeli* tiene registros de usuarios y sus votos de películas que han visto. Esta información esta almacenada en una lista de tuplas, como se muestra a continuación:

`preferencias = [(usuario, pelicula, voto), ... ]`

Cada `tupla` de la lista de `preferencias`, es de la forma `(usuario,pelicula,voto)`, donde `usuario` es un string que representa el nombre de un usuario, `pelicula` es el nombre de una película, y `voto` es la opinión del `usuario` sobre la película expresada en un rango de 1 a 5, siendo 1 un grado mínimo de satisfacción y 5 el máximo.
Una película puede tener muchos votos de diferentes usuarios, y un usuario puede tener muchas votaciones de distintas películas, con la sola restricción que, el voto de un usuario sobre una pelćula es único (un usuario no puede tener dos votos para una misma película).
En base a esta información se te pide que escribas las siguientes funciones:

1. *obtener_ranking* que retorne las 10 películas más vistas. Esta función recibe como parámetro la lista `preferencias`, genera un ranking y retorna una lista como se muestra a continuación:

   `[(pelicula1, t_votos1), (pelicula2, t_votos2), ... , (pelicula10, t_votos10)]`,

   donde `t_voto1` $\geq$ `t_voto2` $\geq$ `...` $\geq$ `t_voto10`, y `t_votoi` (con 1$\leq$i$\leq$10) es el total de votos que la `películai` ha recibido. **Respuesta:**

```
#obtener usuarios y peliculas una funcion auxiliar
def user_peli (registros):
  users = []
  films  = []
  for dato in registros:
    if not dato[0] in users:
      users.append(dato[0])
    if not dato[1] in films:
      films.append(dato[1])
  return users, films


# A mas vistas
def mas_vistas(registros):
  u,p = user_peli(registros)
```

```
  resultado = []
  for peli in p:
    cuenta = 0
    for item in registros:
      if item[1] == peli:
        cuenta += 1
    resultado.append((peli,cuenta))
  aux = []
  aux =sorted(resultado,reverse = True, key=key_pregunta_a)
  repuesta_final = []
  i=0
  while i <10:
    repuesta_final.append(aux[i])
    i += 1
  return repuesta_final
```

2. *obtener_ranking_pvaloradas* de las n películas mejor valoradas (en promedio). Para ello tu función recibe la lista `preferencias` y un valor entero n. La función debe retornar una lista como se muestra a continuación:

   `[(pelicula1, rank1), (pelicula2, rank2), ... , (peliculan, rankn)]`

   donde `rank1` $\geq$ `rank2` $\geq$ `...` $\geq$ `rankn`, y `ranki` (con 1≤i≤n) es el promedio de las votaciones que recibió la `películai`.

   El valor de n en este caso será parámetro, y podría corresponder a 10 (las 10 con mejor ranking), 5 (las 5 con mejor ranking), etc.

```
# B las con n mejor ranking
def mejor_ranking(registros,n):
  u,p = user_peli(registros)
  resultado = []
  for peli in p:
    cuenta = 0
    suma = 0
    for item in registros:
      if item[1] == peli:
        cuenta += 1
        suma += item[2]
    resultado.append((peli,float(suma)/float(cuenta)))
  aux = []
  aux =sorted(resultado,reverse = True, key=key_pregunta_a)
  i=0
  repuesta_final = []
  while i<n:
    repuesta_final.append(aux[i])
    i+=1
  return repuesta_final
```

3. Para el publico más crítico la cadena *PeliPeli* te pide incorporar un ranking de todas las películas que incluyan promedio y total de los votos. Es decir, teniendo como parámetro la lista de preferencias,

la función *obtener_ranking_critico* debe retornar la lista:

`[(pelicula1, rank1, t_votos1), (pelicula2, rank2, t_votos2), (pelicula3, rank3, t_votos3), ... ]`

donde `rank1*t_votos1` $\geq$ `rank2*t_votos2` $\geq$ `rank3*t_votos3` $\geq$ `...` $\geq$ `rankn*t_votosm` , y $\boldsymbol{m}$ el total de películas que existen.

```
# C ranking especial
def mejor_ranking2(registros):
  u,p = user_peli(registros)
  resultado = []
  for peli in p:
    cuenta = 0
    suma = 0
    for item in registros:
      if item[1] == peli:
        cuenta += 1
        suma += item[2]
    ranking = float(suma)/float(cuenta)
    resultado.append((peli,ranking,cuenta,ranking*cuenta))
  aux = []
  aux =sorted(resultado,reverse = True, key=key_pregunta_c)
  i=0
  resultado_final = []
  for item in aux:
    resultado_final.append((item[0],item[1],item[2]))
  return resultado_final
```

Puedes implementar funciones auxiliares si lo deseas.

Pregunta 3 (15 puntos)

Los semáforos dan señales que permiten controlar el tráfico de una ciudad. Estos dispositivos se sitúan en intersecciones viales y otros lugares para regular el tráfico, y por ende, el tránsito peatonal. El tipo mas frecuente controla el tránsito de autos y tiene tres luces de colores: rojo, verde y amarillo; que cambian de rojo a verde, de verde a amarillo, y de amarillo a rojo. En muchas intersecciones se usan además semáforos peatonales para indicar al peatón el momento seguro para que pueda cruzar la intersección. Este tipo de semáforo tiene dos luces de colores: rojo y verde; que cambian de rojo a verde, y de verde a rojo. Por seguridad ambos tipos de semáforos son puestos en marcha en el estado rojo.
El siguiente programa controla los semáforos de una intersección y muestra los siguientes mensajes. En esta parte se te pide que implementes las clases `SemaforoAuto` y `SemaforoPeatonal`.

```
import time
import random
semAuto = SemaforoAuto('semAuto')
semPeatonal  = SemaforoPeatonal('semPeatonal')
while(True):
    semAuto.cambiar()
```

```
    semPeatonal.cambiar(semAuto)
    time.sleep(10) #espera 10 segundos. No implementar este metodo
```

```
semAuto empezo a funcionar con luz  ROJO
semPeatonal empezo a funcionar con luz  ROJO
semAuto cambio de luces: ROJO->VERDE
semPeatonal cambio de luces: VERDE->ROJO
semAuto cambio de luces: VERDE->AMARILLO->ROJO
semPeatonal cambio de luces: ROJO->VERDE
semAuto cambio de luces: ROJO->VERDE
semPeatonal cambio de luces: VERDE->ROJO
semAuto cambio de luces: VERDE->AMARILLO->ROJO
semPeatonal cambio de luces: ROJO->VERDE
semAuto cambio de luces: ROJO->VERDE
semPeatonal cambio de luces: VERDE->ROJO
semAuto cambio de luces: VERDE->AMARILLO->ROJO
semPeatonal cambio de luces: ROJO->VERDE
semAuto cambio de luces: ROJO->VERDE
semPeatonal cambio de luces: VERDE->ROJO
semAuto cambio de luces: VERDE->AMARILLO->ROJO
semPeatonal cambio de luces: ROJO->VERDE
```

```
class SemaforoAuto:
    estado = None
    nombre = ''
    def __init__(self, nombre):
        self.estado = 'ROJO'
        self.nombre = nombre
        print(self.nombre, 'empezo a funcionar con luz ', self.estado)

    def cambiar(self):
        estadoInicial = self.estado
        if self.estado == 'ROJO':
            self.estado = 'VERDE'
            print(self.nombre, 'cambio de luces:', 'ROJO->VERDE')
        elif self.estado == 'VERDE':
            self.estado = 'ROJO'
            print(self.nombre, 'cambio de luces:', 'VERDE->AMARILLO->ROJO')

class SemaforoPeatonal:
    estado = None
    def __init__(self, nombre):
        self.nombre = nombre
        self.estado = 'ROJO'
        print(self.nombre, 'empezo a funcionar con luz ', self.estado)
    def cambiar(self, semaforo):
        estadoInicial = self.estado
        if semaforo.estado == 'ROJO':
            self.estado = 'VERDE'
```

```
            print(self.nombre, 'cambio de luces:', 'ROJO->VERDE')
        elif semaforo.estado == 'VERDE':
            self.estado = 'ROJO'
            print(self.nombre, 'cambio de luces:', 'VERDE->ROJO')
        estadoFinal = self.estado
```

Pregunta 4 (15 puntos)

Dados dos números enteros, **x** e **y**, donde **x** <= **y**, escribe un programa en Python para llegar del número **x** al número **y** en un número mínimo de operaciones, siendo las únicas operaciones posibles: +1 (sumar 1) y x2 (multiplicar por 2).

Ejemplo:

```
5  23
23 = ((5 * 2 + 1) * 2 + 1)

11   113
113 = ((((11 + 1) + 1) + 1) * 2 * 2 * 2 + 1)
```

Del ejemplo anterior, entre los número 5 y 23, hubo 4 operaciones para llegar de 5 a 23. Un ejemplo donde **NO** hay un mínimo de operaciones para llegar de 5 y 23 sería:

```
5  23
23 = ((5 + 1 + 1 + 1 + 1 + 1) * 2 + 1 + 1 + 1)
```

```
def convierte(a,b):
   if (a == b):
      return "" + str(a)
   elif (b % 2 == 1):
      return "(" + convierte(a, b-1) + " + 1)"
   elif (b < 2 * a):
      return "(" + convierte(a, b-1) + " + 1)"
   else:
      return convierte(a, b//2) + " * 2"
```

Pregunta 5 (15 puntos)

a) Escribe en python una función **codificar** que reciba como parámetro un string, **s**, y devuelva el string codificado. Para ello considera lo siguiente:

- Una secuencia de **n** caracteres iguales consecutivos en **s** es codificada como el carácter seguido del número **n**; excepto cuando **n** es igual a 1, en cuyo caso se omite **n**.

- Ejemplo, “aammmennnniiixooooooyyyyyyyuzz”, se codifica como “a2m3en4i3xo5y6uz2”. Puedes suponer que los caracteres del string s original no incluyen dígitos.

```
def codificar(texto):
    texto_codificado = ""
    i = 0
    while i < len(texto):
        cuenta = 0
        for j in range(i,len(texto)):
            if(texto[i] == texto[j]):
                cuenta+=1
                i = j
            else:
                break
        if cuenta > 1:
            texto_codificado+=texto[i]+str(cuenta)
        else:
            texto_codificado+=texto[i]
        i+=1;
    return texto_codificado
```

b) Escribe la función **decodificar** correspondiente a la pregunta a); es decir, **decodificar** recibe como parámetro un string **z**, que es un string codificado, y devuelve el string decodificado.

Para simplificar, asuma que nunca vienen 2 números consecutivos.

```
numeros = '0123456789'
def decodificar(texto):
    texto_decodificado = ""
    i = 0
    while i < len(texto):
        if texto[i] in numeros:
            texto_decodificado += texto[i-1]*int(texto[i])
        if texto[i] not in numeros and texto[i+1] not in numeros:
            texto_decodificado += texto[i]
        i +=1
    return texto_decodificado
```