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

Clase 22: Ejercicios Examen (parte 1)

Rodrigo Toro Icarte (rntoro@uc.cl)

IIC1103 Introducción a la Programación - Sección 5

15 de Abril, 2015

Examen

Temario examen:

- Control de Flujo.
- Funciones.
- Strings.
- Lectura y Escritura de Archivos.
- Listas.
- Búsqueda y Ordenamiento.
- Programación Orientada a Objetos.
- Recursión.
- Backtracking.

Examen

Hoy:

- Control de Flujo.
- Funciones.
- Strings.
- Lectura y Escritura de Archivos.
- Listas.
- Búsqueda y Ordenamiento.
- Programación Orientada a Objetos.
- Recursión.
- Backtracking.

Midterm P1

Para transformar un número binario b (de n bits) a decimal se utiliza la siguiente fórmula:

$$d = \sum_{i=0}^{n-1} b_i * 2^i$$

Donde b_i es el i-ésimo bit de b, contados de **derecha a izquierda** y **comenzando desde cero**.

Realiza un programa que pida un número binario y muestre el valor de cada uno de sus bytes (8 dígitos). Por ejemplo si el usuario ingresa 10100101011011011101, se debiera mostrar 221, 86 y 10.

1010	01010110	11011101
10	86	221

Midterm P1

Primero debemos ser capaces de descomponer el binario en sus dígitos.

Midterm P1

Primero debemos ser capaces de descomponer el binario en sus dígitos.

```
1 def pedir_binario():
2   return int(input("Ingrese un binario: "))
3
4 n = pedir_binario()
5
6 while(n > 0):
7   print(n%10)
8   n = n //10
```

Midterm P1

Ahora que tenemos los dígitos, utilicemos la fórmula para obtener el entero:

$$d = \sum_{i=0}^{n-1} b_i * 2^i$$

Midterm P1

Ahora que tenemos los dígitos, utilicemos la fórmula para obtener el entero:

$$d = \sum_{i=0}^{n-1} b_i * 2^i$$

```
4  n = pedir_binario()
5  entero = 0
6  i = 0
7  while(n > 0):
8    entero += (n%10)*2**i
9    n = n //10
10   i += 1
11 print(entero)
```

Midterm P1

Finalmente mostremos cada 8 dígitos:

Midterm P1

Finalmente mostremos cada 8 dígitos:

```
4  n = pedir_binario()
5  entero = 0
6  i = 0
7  while(n > 0):
8    entero += (n%10)*2**i
9    n = n //10
10   i += 1
11   if(i == 8):
12     print(entero)
13     i = 0
14     entero = 0
15
16 if(i != 0):
17   print(entero)
```

Midterm P2

a) Escribe una función `divisor(x,i)` que entregue el i-ésimo divisor de x (sin incluir a x). Si i está fuera de rango, debe devolver -1. Por ejemplo, `divisor(12,1)` = 1, `divisor(12,5)` = 6 y `divisor(12,6)` = -1.

Midterm P2

a) Escribe una función `divisor(x,i)` que entregue el i-ésimo divisor de x (sin incluir a x). Si i está fuera de rango, debe devolver -1. Por ejemplo, `divisor(12,1)` = 1, `divisor(12,5)` = 6 y `divisor(12,6)` = -1.

```
1 def divisor(x,i):
2   num_divisor = 0
3   for n in range(1,x):
4     if(x%n == 0):
5       num_divisor+=1
6       if(num_divisor == i):
7         return n
8   return -1
```

Midterm P2

b) Un número abundante es un número para el cual la suma de todos sus divisores (excluido él mismo) suma más que el número. Usa la función anterior (divisor) para escribir una función `es_abundante(x)` que entregue `True` si `x` es abundante, `False` si no. Por ejemplo los divisores de 12 suman 16, por lo que 12 es un número abundante.

Midterm P2

```
10 def es_abundante(x):
11   suma = 0
12   i = 1
13   while(True):
14     d = divisor(x,i)
15     if(d == -1):
16       break
17     suma += d
18     i += 1
19   return suma > x
```

Midterm P2

c) Usa la/s función/es anteriores (`divisor` y/o `es_abundante`) para escribir un programa que pregunte un número al usuario y en caso de ser posible, lo muestre como la suma de dos números abundantes. Si hay dos posibles respuestas puedes mostrar cualquiera de las dos.

Ejemplos:

- $20222 = 12 + 20150$.
- $60 = 30 + 30$.
- 34 no es posible.

Midterm P2

```
21 n = int(input("Ingrese número: "))
22 es_posible = False
23 for i in range(1,n):
24   for j in range(1,n):
25     if(not es_posible and
26         es_abundante(i) and
27         es_abundante(j) and
28         (i + j) == n):
29       print(n,"=",i,"+",j)
30       es_posible = True
31
32 if(not es_posible):
33   print("No es posible")
```

Midterm P3

Recibes de un amigo código escrito en Python, pero resulta que lo escribió para Python 2.7 y tú ya usas Python 3.4. La mayor diferencia entre ambas versiones es la función `print`: en Python 2.7 va sin paréntesis.

Escribe un programa que modifique el archivo `juego_python27.py` y lo guarde en un nuevo archivo `juego_python34.py`, en el que cada instrucción `print` que encuentre se modifique por la versión más nueva de la instrucción. Por supuesto, no debes cambiar las otras instrucciones. Puedes considerar que los strings siempre tienen comillas dobles.

Midterm P3

La siguiente tabla muestra ejemplos de líneas y cómo deben quedar en el archivo final.

Python 2.7	Python 3.4
`print "hola!"`	`print("hola!")`
`print x`	`print(x)`
`#esta linea no tiene print, solo comentario`	`#esta linea no tiene print, solo comentario`
`print x #imprimi x con print`	`print(x) #imprimi x con print`
`print " hola" #imprimi " hola" con print`	`print(" hola")#imprimi " hola" con print`
`print " comentario es con #"`	`print(" comentario es con #")`

Midterm P3

Leer el archivo `juego_python27.py`:

Midterm P3

Leer el archivo `juego_python27.py`:

```
1 f = open('juego_python27.py')
2 for l in f:
3   print(l.strip())
4 f.close()
```

Midterm P3

Leer el archivo `juego_python27.py`:

```
1 f = open('juego_python27.py')
2 for l in f:
3   print(l.strip())
4 f.close()
```

Escribir el archivo `juego_python34.py`:

Midterm P3

Leer el archivo juego_python27.py:

```
1 f = open('juego_python27.py')
2 for l in f:
3   print(l.strip())
4 f.close()
```

Escribir el archivo juego_python34.py:

```
1 f = open('juego_python27.py')
2 w = open('juego_python34.py','w')
3 for l in f:
4   w.write(l)
5 f.close()
```

Midterm P3

¿Cómo agregamos los paréntesis?

Python 2.7	Python 3.4
`print "hola!"`	`print("hola!")`
`print x`	`print(x)`
`#esta linea no tiene print, solo comentario`	`#esta linea no tiene print, solo comentario`
`print x #imprimi x con print`	`print(x) #imprimi x con print`
`print " hola" #imprimi " hola" con print`	`print(" hola")#imprimi " hola" con print`
`print " comentario es con #"`	`print(" comentario es con #")`

Midterm P3

```
18 f = open('juego_python27.py')
19 w = open('juego_python34.py','w')
20 contador = 0
21 for l in f:
22   pos_print = l.find("print")
23   pos_comentario = l.find("#")
24   if(pos_print == -1 or -1 < pos_comentario <
       pos_print):
25     w.write(l)
26   else:
27     s = tiene_print(l,pos_print)
28     w.write(s)
29
30 f.close()
```

Midterm P3

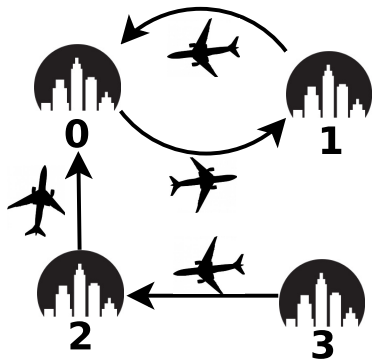
```
1  def tiene_print(l,pos_print):
2    s = l[:pos_print+5] + "("
3    resto = l[pos_print+5:].strip(" ")
4    # Recorro el resto
5    buscar = 0
6    for c in resto:
7      if(buscar == 0 and c == '"'):
8        buscar = 1; s+=c
9      elif(buscar == 0 and c != ' '):
10       buscar = 2; s+=c
11     elif(buscar == 1 and c == '"'):
12       buscar = 3; s+= c + ")"
13     elif(buscar == 2 and c in [' ','#','\n']):
14       buscar = 3; s+= ")" + c
15     else: s+=c
16   return s
```

Midterm P4

Una aerolínea representa sus vuelos directos mediante una tabla con unos y ceros. La tabla es un cuadrado tal que cada fila y columna está asociada a una ciudad particular. Si la casilla `(i,j)` tiene un 1, significa que existe un vuelo directo que lleva desde la ciudad `i` hasta la ciudad `j`, mientras que un cero significa que no existe. El orden de las ciudades en filas y columnas es el mismo, es decir, la ciudad asociada a la *fila* `i` es la misma que la asociada a la *columna* `i`.

Midterm P4

Por ejemplo, considera que la aerolínea llega a 4 ciudades (cuyos índices son: 0, 1, 2 y 3) y tiene los siguientes vuelos: 3 $\rightarrow$ 2, 2 $\rightarrow$ 0, 0 $\rightarrow$ 1 y 1 $\rightarrow$ 0 (figura 1a). Su tabla de vuelos en Python sería la figura 1b, que denominaremos `t`.




```
33 t = [[0,1,0,0],
34      [1,0,0,0],
35      [1,0,0,0],
36      [0,0,1,0]]
```

(a) **(b)**

Figure : Ejemplo con 4 ciudades.

Midterm P4

a) `vuelosSinVuelta(tabla)`: Esta función recibe la tabla de vuelos y retorna una lista con los vuelos sin vuelta directa. Los elementos de la lista retornada deben contener los índices de la ciudad de origen y destino del vuelo. Por ejemplo, el retorno de la función para `t` debería ser `[[2,0],[3,2]]`.

Midterm P4

Recorrer la tabla de vuelos:

Midterm P4

Recorrer la tabla de vuelos:

```
1 def vuelosSinVuelta(tabla):
2   for i in range(len(tabla)):
3     for j in range(len(tabla[i])):
```

¿Cuándo existe un vuelo de `i` a `j`?

Midterm P4

Recorrer la tabla de vuelos:

```
1 def vuelosSinVuelta(tabla):
2   for i in range(len(tabla)):
3     for j in range(len(tabla[i])):
```

¿Cuándo existe un vuelo de `i` a `j`?

```
1 def vuelosSinVuelta(tabla):
2   for i in range(len(tabla)):
3     for j in range(len(tabla[i])):
4       if(tabla[i][j] == 1)
```

Midterm P4

¿Cuándo el vuelo de `i` a `j` no tiene vuelta?

Midterm P4

¿Cuándo el vuelo de `i` a `j` no tiene vuelta?

```
1 def vuelosSinVuelta(tabla):
2   for i in range(len(tabla)):
3     for j in range(len(tabla[i])):
4       if(tabla[i][j] == 1 and tabla[j][i] == 0):
```

Midterm P4

¿Cómo agrego los vuelos sin vuelta a una lista?

Midterm P4

¿Cómo agrego los vuelos sin vuelta a una lista?

```
1 def vuelosSinVuelta(tabla):
2   l = []
3   for i in range(len(tabla)):
4     for j in range(len(tabla[i])):
5       if(tabla[i][j] == 1 and tabla[j][i] == 0):
6         l.append([i,j])
7   return l
```

Midterm P4

b) `vuelosConUnaEscala(tabla)`: Esta función recibe la tabla de vuelos y retorna una nueva tabla (de igual dimensión) que tiene un `1` en la casilla `(i,j)` si y solo si puedo ir desde la ciudad `i` a la ciudad `j` en forma directa, o realizando una escala (i.e. puedo ir de `i` a `k`, y luego de `k` a `j`). Por ejemplo, en `t` es posible ir de `3` a `0` haciendo escala en `2`, y de `2` a `1` haciendo escala en `0`. No consideres casos en que `i` sea igual a `j`.

Midterm P4

¿Cómo creo una copia del tablero original?

Midterm P4

¿Cómo creo una copia del tablero original?

```
1 def vuelosConUnaEscala(tabla):
2   ret = []
3   for i in range(len(tabla)):
4     fila = []
5     for j in range(len(tabla[i])):
6       fila.append(tabla[i][j])
7     ret.append(fila)
8   return ret
```

Midterm P4

¿Qué falta?

Midterm P4

¿Qué falta?

```
1  def vuelosConUnaEscala(tabla):
2    ret = []
3    for i in range(len(tabla)):
4      fila = []
5      for j in range(len(tabla[i])):
6        v = 0
7        # Si existe vuelo directo cambio v -> 1
8        # Si existe vuelo con una escala cambio v -> 1
9        fila.append(v)
10     ret.append(fila)
11   return ret
```

Midterm P4

¿Cuándo existe un vuelo directo?

Midterm P4

¿Cuándo existe un vuelo directo?

```
1  def vuelosConUnaEscala(tabla):
2    ret = []
3    for i in range(len(tabla)):
4      fila = []
5      for j in range(len(tabla[i])):
6        v = 0
7        # Si existe vuelo directo cambio v -> 1
8        if(tabla[i][j] == 1):
9          v = 1
10       # Si existe vuelo con una escala cambio v -> 1
11       fila.append(v)
12     ret.append(fila)
13   return ret
```

Midterm P4

¿Cuándo existe un vuelo con una escala?

Midterm P4

¿Cuándo existe un vuelo con una escala?

```
9  def vuelosConUnaEscala(tabla):
10   ret = []
11   for i in range(len(tabla)):
12     fila = []
13     for j in range(len(tabla[i])):
14       v = 0
15       if(tabla[i][j] == 1):
16         v = 1
17       for k in range(len(tabla)):
18         if(tabla[i][k]==1 and tabla[k][j]==1 and
19             i!=j):
20           v = 1
21       fila.append(v)
22     ret.append(fila)
23   return ret
```

Midterm P4

c) `vuelosConEscala(tabla)`: Esta función recibe la tabla de vuelos y retorna una nueva tabla que tiene un `1` en la casilla `(i,j)` si y solo si puedo ir desde la ciudad `i` a la ciudad `j` en forma directa, o realizando algún número de escalas. Por ejemplo, en `t` es posible ir de `3` a `1` realizando escalas en `2` y `0`.
Hint: Utilice la función `vuelosConUnaEscala(t)`.

Midterm P4

c) `vuelosConEscala(tabla)`: Esta función recibe la tabla de vuelos y retorna una nueva tabla que tiene un `1` en la casilla `(i,j)` si y solo si puedo ir desde la ciudad `i` a la ciudad `j` en forma directa, o realizando algún número de escalas. Por ejemplo, en `t` es posible ir de `3` a `1` realizando escalas en `2` y `0`. **Hint**: Utilice la función `vuelosConUnaEscala(t)`.

```
25 def vuelosConEscala(tabla):
26   while(True):
27     r = vuelosConUnaEscala(tabla)
28     # comparo tablas
29     if(r == tabla):
30       return r
```

Pregunta inédita

¿Cómo ordenamos una lista de palabras alfabéticamente?

Pregunta inédita

¿Cómo ordenamos una lista de palabras alfabéticamente?

```
1 palabras = ['ajedrez','Jaguar','foto','especular','
      carpintero','colgante','trepar']
2 palabras.sort()
3 print(palabras)
```

Pregunta inédita

¿Cómo ordenamos una lista de palabras alfabéticamente?

```
palabras = ['ajedrez','Jaguar','foto','especular','
    carpintero','colgante','trepar']
palabras.sort()
print(palabras)
```

¿Cómo ordenarían una lista de palabras siguiendo otro orden alfabético (por ejemplo, donde la `"r"` esté antes que la `"g"`)?

Pregunta inédita

Ejemplo:

- Alfabeto = **p**,a,b,c,d,e,f,**r**,g,h,i,j,k,l,m,n,o,q,s,t,u,v,w,x,y,z.
- Palabras = `gato`, `agua`, `arena`, `pasto`, `roedor`.

Pregunta inédita

Ejemplo:

- Alfabeto = **p**,a,b,c,d,e,f,**r**,g,h,i,j,k,l,m,n,o,q,s,t,u,v,w,x,y,z.
- Palabras = `gato`, `agua`, `arena`, `pasto`, `roedor`.

Las palabras ordenadas normalmente serían: `agua`, `arena`, `gato`, `pasto` y `roedor`.

Pregunta inédita

Ejemplo:

- Alfabeto = **p**,a,b,c,d,e,f,**r**,g,h,i,j,k,l,m,n,o,q,s,t,u,v,w,x,y,z.
- Palabras = `gato`, `agua`, `arena`, `pasto`, `roedor`.

Las palabras ordenadas normalmente serían: `agua`, `arena`, `gato`, `pasto` y `roedor`.

Sin embargo, bajo el nuevo orden alfabético sería: `pasto`, `arena`, `agua`, `roedor` y `gato`.

Pregunta inédita

a) Implementa la función `indice(c,l)`, donde `c` es un caracter y `l` una lista con los caracteres alfabéticos en algún orden cualquiera. La función debe retornar el índice en que aparece `c` dentro de `l`.

Pregunta inédita

a) Implementa la función `indice(c,l)`, donde `c` es un caracter y `l` una lista con los caracteres alfabéticos en algún orden cualquiera. La función debe retornar el índice en que aparece `c` dentro de `l`.

```
1 def indice(c,l):
2   for i in range(len(l)):
3     if(l[i] == c):
4       return i
5   return -1
```

Pregunta inédita

b) Implementa la función `mayor_que(s1,s2,l)`, que recibe como parámetros dos palabras (`s1` y `s2`), una lista con los caracteres alfabéticos `l`, y retorna `True` ssi `s1` $>$ `s2` siguiendo el alfabeto `l`. Rompe empates según el largo de la palabra.

Pregunta inédita

b) Implementa la función `mayor_que(s1,s2,l)`, que recibe como parámetros dos palabras (`s1` y `s2`), una lista con los caracteres alfabéticos `l`, y retorna `True` ssi `s1` $>$ `s2` siguiendo el alfabeto `l`. Rompe empates según el largo de la palabra.

```
7  def mayor_que(s1,s2,l):
8    for i in range(min([len(s1),len(s2)])):
9      i1 = indice(s1[i],l)
10     i2 = indice(s2[i],l)
11     if(i1 > i2):
12       return True
13     if(i1 < i2):
14       return False
15   return len(s1) > len(s2)
```

Pregunta inédita

c) Implementa la función `ordenar(p,l)`, que recibe una lista de palabras `p` con un alfabeto `l`; y retorna la lista de palabras ordenadas según `l`.

Pregunta inédita

c) Implementa la función `ordenar(p,l)`, que recibe una lista de palabras `p` con un alfabeto `l`; y retorna la lista de palabras ordenadas según `l`.

```
17 def ordenar(p,l):
18   for i in range(len(p)):
19     for j in range(i+1,len(p)):
20       if(mayor_que(p[i],p[j],l)):
21         aux = p[j]
22         p[j] = p[i]
23         p[i] = aux
24   return p
```

Ejercicios propuestos

1) Programa desde cero los ejercicios vistos en la clase.

2) Resuelve Midterms y Exámenes de semestres anteriores.