## ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

## FACULTAD DE INGENIERÍA EN ELECTRICIDAD Y COMPUTACIÓN FUNDAMENTOS DE PROGRAMACIÓN PRIMERA EVALUACIÓN - I TÉRMINO 2015

Nombre:	Matrícula:

## Tema 1

La *persistencia aditiva* de un número entero se calcula sumando sus dígitos y en caso que esta sumatoria tenga más de un dígito, se repetirá el proceso sobre esta, hasta alcanzar un único dígito. La cantidad de veces que se requiera realizar la sumatoria hasta obtener un único dígito se denomina persistencia aditiva. **Por ejemplo**:

- El número 1234 tiene una persistencia aditiva de 2 (la primera suma de dígitos es 10, luego la segunda suma es 1).
- El número 5978 tiene una persistencia aditiva de 3 (5978 $\rightarrow$ 29 $\rightarrow$ 11 $\rightarrow$ 2).
- El número 9 tiene una persistencia aditiva de 0.

A usted se le solicita implementar en Python:

1. La función *calcularPersistenciaAditiva*, la cual recibe como parámetro un número entero positivo denominado *número* y retorna su persistencia aditiva. [25%]

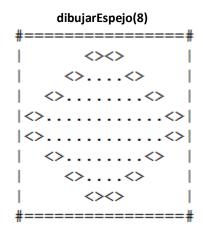
## Tema 2

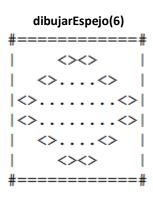
A usted se le solicita implementar en Python:

1. La función *dibujarEspejo* que recibe un número *n* y dibuja el siguiente patrón:

[20%]

Nota: Asuma que n es siempre un número par.





Tema 3 [45%]

El acumula-tesoros es un juego de tablero en el que dos jugadores deben atravesar una ciudad avanzando y retrocediendo por un corredor de 50 metros de largo. Mientras atraviesan la ciudad, a cada jugador se le asigna, aleatoriamente, al llegar a cada posición uno de los siguientes *estados*: fortaleza (-1), hambre (-2) y súper-héroe (-3). Algunos metros pueden tener asignados uno de los siguientes *elementos*: armas (1), víveres (2) o tesoros (3), elementos que podrán ser recogidos si el jugador está en el estado adecuado:

- Para recoger armas, el jugador debe estar en el estado de fortaleza.
- Para recoger víveres, el jugador debe estar en el estado de hambre.
- Para recoger tesoros, el jugador debe estar en el estado de súper-héroe.

Cada jugador avanza usando un dado de 6 caras y en cada movimiento se verifica:

- Si la ubicación ya ha sido visitada primero por el otro jugador, entonces se debe retroceder 10
  metros o volver al inicio si estuviera a menos de 10 metros del mismo. En caso que el jugador
  desee evitar el retroceso, puede evitarlo deshaciéndose de un tesoro de los que haya recogido.
- Si la ubicación no ha sido visitada aún, el jugador puede recoger algún elemento dependiendo del estado en el que se encuentre.

Al inicio del juego, usted debe distribuir aleatoriamente 8 armas, 8 víveres y 8 tesoros a lo largo del corredor, *excepto* en la posición 1, en donde se ubican ambos jugadores al iniciar la partida. El juego termina cuando uno de los jugadores llega *exactamente* al final del corredor. Se declara ganador al jugador que obtenga mayor riqueza, la misma que depende de los elementos recogidos a lo largo del juego y se calcula mediante la siguiente fórmula:

Riqueza = (armas/2+1)\*100 + viveres\*0.5 + tesoros

A usted se le solicita implementar en Python:

- 1. La función *asignarPosicionElementos* que retorna una colección que contiene 3 listas, cada una representando las *posiciones* en el corredor de las 8 armas, víveres y tesoros, respectivamente.
- 2. La función *lanzarDado* que retorna aleatoriamente el valor de una de las *caras* del dado.
- 3. La función *generarEstado* que retorna aleatoriamente uno de los posibles *estados*.
- 4. La función *mostrarElemento* que dada una *posición* y la colección de *posiciones de los elementos* imprime en pantalla el tipo de elemento que existe en esa posición.
- 5. La función *recogerElemento* que dada una posición, el estado de un jugador y la colección de *posiciones de los elementos* retorna el *tipo de elemento* que se ha recogido (1, 2 ó 3) o 0 si no ha sido posible.
- 6. La función *mostrarMensaje*, la cual recibe como parámetro un *jugador*, el *estado* del mismo y su *posición* para imprimirlos en pantalla.
- 7. La función *calcularRiqueza* que dada una cantidad de *armas, víveres* y *tesoros* retorna la *riqueza* obtenida.
- 8. Un programa que simule el juego y **use** las funciones implementadas anteriormente.

Al finalizar el juego, se debe declarar al ganador y la riqueza de cada uno.

En el turno de cada jugador, se deberá mostrar por pantalla:

- El jugador al que corresponde el turno actual
- El resultado del lanzamiento del dado y la nueva posición del jugador
- Si hay un retroceso

Tema 4 [10%]

Analice el código fuente de los programas que se muestran a continuación. Seleccione la respuesta correcta y justifique brevemente su respuesta.

a. Determine la salida por pantalla del siguiente código:

```
X = 2
y = 5
z = x + z
print("La suma es ,z")
```

- a) Error: La variable z no ha sido definida
- b) La suma es ,z
- c) Error: La variable z no se ha inicializado
- d) La suma es 7
- b. Dado el siguiente segmento de código y las listas A y B, seleccione correctamente la salida por pantalla:

```
A = [3, 2, 7, 5]

B = [31, 5, 4, 8, 12, 3, -9, 6]

C = 0

N = 3

for i in range(0, 4) :

B[A[i]] = B[A[i]] + N

C += B[A[i]]

print(C)
```

- a) 27
- b) 33
- c) 6
- d) Ninguna de las anteriores