Comenzado el	jueves, 3 de diciembre de 2020, 08:47
Estado	Finalizado
Finalizado en	jueves, 3 de diciembre de 2020, 09:22
Tiempo empleado	34 minutos 57 segundos
Puntos	18/25
Calificación	**7** de 10 (**70**%)

Pregunta **1**

Finalizado

Puntúa 1 sobre 1

Suponga que para un problema *p* dado, cuyo volumen de entrada es *n*, se conocen tres algoritmos diferentes *a1*, *a2* y *a3* para resolverlo. Suponga que los tiempos de ejecución de estos tres algoritmos son t(a1) = $O(2^n)$, t(a2) = $O(n^4)$ y t(a3) = $O(n^2*\log(n))$ *¿Cuáles de las siguientes afirmaciones son ciertas respecto del problema p desde el punto de vista de la Teoría de la Complejidad Computacional*? (Más de una respuesta puede ser cierta... marque todas las que considere correctas...)

Seleccione una o más de una:

- [ ] a.
  No hay forma de decidir si p es intratable o no, ya que algunos de sus algoritmos ejecutan en tiempo exponencial y otros lo hacen en tiempo polinomial o subpolinomial.

- [x] b.
  Los algoritmos a2 y a3 tienen tiempos de ejecución aceptables para la teoría, por lo tanto p no es un problema intratable.

- [ ] c.
  El problema p es intratable, ya que el algoritmo a1 ejecuta en tiempo exponencial (considerado impráctico por la teoría).

- [x] d.
  El problema p no es intratable, ya que al menos uno de los algoritmos que se conocen para resolverlo ejecuta en tiempo polinomial o subpolinomial.

Pregunta **2**

Finalizado

Puntúa 1 sobre 1

Suponga que la clase *Insumo* está convenientemente declarada para representar los insumos que se usan en la fabricación de una pieza. Asuma que también se ha definido un vector *pieza* de referencias a registros de tipo *Insumo* y que ese vector fue creado con ***n*** casilleros valiendo *None*. El programa mostrado en la Ficha 18 (problema 48) prevé que el arreglo será cargado luego con registros de ese tipo *Insumo*, entrando en la opción 1 del menú. Más abajo mostramos una variante de la función *opcion3()* para calcular el monto total insumido para fabricar la pieza completa, la cual a su vez invoca a la función *total_value()* (que se ha dejado sin cambios) ¿Qué puede decirse respecto de la forma en que afecta al programa este replanteo de la función *opcion3()*?

```
def total_value(pieza):
    tv = 0
    for insumo in pieza:
        monto = insumo.valor * insumo.cantidad
        tv += monto
    return tv


def opcion3(pieza):
    print('Monto total en insumos para la pieza:',
total_value(pieza))
```

Seleccione una:

a.
Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador provocará un error y se interrumpirá ya que no existe en ese caso ningún campo llamado valor o cantidad. El programa se interrumpirá con un mensaje de error.

b.
Si el vector *pieza* no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador automáticamente cambiará el None de cada casillero por un registro de tipo Insumo inicializado con valores cero. Por lo tanto en ese caso la función retornará el valor final 0.

c.
El programa funcionará correctamente de todos modos. Si cada casillero del vector valiese None, el ciclo iterador de la función total_value() terminará sin hacer nada y la función retornará 0.

d.
Si el vector pieza no hubiese sido efectivamente cargado antes, la función opcion3() invocará a total_value() enviándole un vector con todos sus casilleros valiendo None, y el ciclo iterador simplemente dejará el valor None en el acumulador tv. La función terminará en ese caso retornarndo None.

Pregunta **3**

Finalizado

Puntúa 0 sobre 1

Asuma que se quiere definir una función *registrar()* como la que se muestra más abajo, en la cual se necesita crear una matriz *reg* de forma que en esa matriz cada fila represente un año calendario desde el 2000 en adelante (la fila 0 representaría al año 2000, la 1 al 2001 y así sucesivamente) y cada columna represente una de las carreras disponibles en una universidad (numeradas desde 0 en adelante). La idea es que una vez creada esa matriz, debe usarse para almacenar en cada casillero una cierta cantidad fija de inscriptos ¿Hay algún problema con la siguiente función en Python, asumiendo que ***a*** es la cantidad de filas y ***c*** es la cantidad de columnas?

```
def registrar(a, c):
    reg = []
    for i in range(a):
        for j in range(c):
            reg[i][j] = 50

    return reg
```

Seleccione una:

a.
No, no hay problema alguno. La función crea la matriz pedida, e inicializa cada casillero con el valor 50.

b.
Sí, hay un problema: los índices *i* y *j* están al revés cuando se usaron como índices para entrar a una casilla de la matriz.

c.
Sí, hay un problema: al ejecutar la función de producirá un error de intérprete y el programa se interrumpirá, debido a que la matriz en realidad nunca fue creada.

d.
Sí, hay un problema pero es menor: la función crea efectivamente la matriz pedida, pero luego inicializa cada casillero con el mismo valor, lo cual no tiene sentido práctico.

Pregunta **4**

Finalizado

Puntúa 1 sobre 1

¿Cuántas comparaciones en el ***peor caso*** obliga a hacer una búsqueda secuencial en una *lista ordenada* (o en un *arreglo ordenado*) que contenga *n* valores?

Seleccione una:

a.
Peor caso: O(n) comparaciones.

b.
Peor caso: O(1) comparaciones.

c.
Peor caso: $O(n^2)$ comparaciones.

d.
Peor caso: O(log(n)) comparaciones.

Pregunta **5**

Finalizado

Puntúa 1 sobre 1

¿Qué se entiende (en el contexto de la Teoría de la Complejidad Computacional) por *Problemas Intratables*?

Seleccione una:

a.
Son problemas para los que sólo se conocen soluciones cuyo tiempo de ejecución es $O(n^k)$ (o sea, tiempo de "ejecución polinomial") siendo n el tamaño del problema.

b.
Son problemas para los que *sólo* se conocen soluciones cuyo tiempo de ejecución es $O(2^n)$ (o alguna otra función o relación super polinomial) siendo n el tamaño del problema.

c.
Son problemas para los que no se conoce solución alguna.

d.
Son problemas para los que se conocen soluciones cuyo tiempo de ejecución es $O(2^n)$ (o alguna otra función o relación super polinomial), pero para los que *también* se conocen soluciones de tiempo polinomial $O(n^k)$, siendo n el tamaño del problema.

Pregunta **6**

Finalizado

Puntúa 1 sobre 1

El siguiente programa usa una función para verificar si un número que se carga por teclado es mayor que el triple de otro número que también se carga. ¿Cuál es el efecto de ejecutar esta función?

```
__author__ = 'Catedra de AED'


def ejemplo():
    i = int(input('Ingrese un valor: '))
    t = int(input('Ingrese otro: '))
    if i > 3*t:
        ok = True

    if ok:
        print('El primer valor es mayor al triple del
segundo...')


# script principal
ejemplo()
```

Seleccione una:

a.
Mostrará el mensaje *El primer valor es mayor al triple del segundo* si *i > 3*t*, pero lanzará un error y se interrumpirá si *i <= 3*t*, ya que en este caso la variable *ok* quedaría sin definir.

b.
Mostrará el mensaje *El primer valor es mayor al triple del segundo* sean cuales sean los valores que se carguen en *i* y en *t*.

c.
Mostrará el mensaje *El primer valor es mayor al triple del segundo* sólo si el valor cargado en *i* es menor al valor de *t* multiplicado por 3. No hará nada en caso contrarío.

d.
Mostrará el mensaje *El primer valor es mayor al triple del segundo* sólo si el valor cargado en *i* es menor o igual al valor cargado en *t* multiplicado por 3. No hará nada en caso contrario.

Pregunta **7**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes propuestas generales son ***ciertas*** en relación al segmento de memoria conocido como *Stack Segment*?

Seleccione una o más de una:

- [x] a.
  El Stack Segment funciona como una pila (o apilamiento) de datos (modalidad *LIFO*: Last In - First Out): el último dato en llegar, se almacena en la cima del stack, y será por eso el primero en ser retirado.

- [ ] b.
  El Stack Segment se utiliza como soporte interno *solamente* en el proceso de invocación a funciones recursivas: se va llenando a medida que la cascada recursiva se va desarrollando, y se vacía a medida que se produce el proceso de regreso o vuelta atrás.

- [x] c.
  El Stack Segment se utiliza como soporte interno en el proceso de invocación a funciones *(con o sin recursividad)*: se va llenando a medida que se desarrolla la cascada de invocaciones, y se vacía a medida que se produce el proceso de regreso o vuelta atrás.

- [ ] d.
  El Stack Segment funciona como una cola (o fila) de datos (modalidad *FIFO*: First In - First Out): el primer dato en llegar, se almacena al frente del stack, y será por eso el primero en ser retirado.

Pregunta **8**

Finalizado

Puntúa 0 sobre 1

Suponga que luego de haber terminado el cursdo de la materia, usted es realmente un genio de la programación y que, gracias a los grandes profesores que ha tenido, logra resolver el problema de la Satisfactibilidad Booleana a partir de aplicar una reducción polinómica hacia el algoritmo de la Búsqueda Binaria. ¿Qué afirmaciones son correctas a partir de ese hecho? (Más de una podría ser cierta, por lo que marque todas las que considere correctas)

Seleccione una o más de una:

- [ ] a.
  Me haré mundialmente famoso pues habré demostrado que P es distinto de NP.

- [ ] b.
  Me haré mundialmente famoso pues habré demostrado que P es igual a NP.

- [x] c.
  No ocurrirá nada, pues ambos los algoritmos para resolver ambos problemas son del mismo orden en tiempo de ejecución y por lo tanto el hallazgo no tendría ninguna relevancia. Nada habrá cambiado en la Teoría de la Complejidad.

- [ ] d.
  Me haré famoso, pero no tanto. Resolver la Satisfactibilidad Booleana reduciéndilo al algoritmo de Búsqueda Binaria solo implica haber encontrado una solución de orden menor para ese problema en particular y eso es todo. Nada habrá cambiado en la Teoría de la Complejidad.

Pregunta **9**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método *read()*, contenido en cualquier variable *file object* usada para manipular un archivo de texto en Python? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- [x] a.
  Si *m* es el *file object* que representa un archivo de texto, la invocación *cad = m.read()* lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, se preservan y se retornan con la cadena devuelta.

- [ ] b.
  Si *m* es el *file object* que representa un archivo de texto, la invocación *cad = m.read()* lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Los caracteres de salto de línea que pudiese contener el archivo, no se preservan: la cadena devuelta no contendrá saltos de línea de ningún tipo, en ninguna posición.

- [x] c.
  Si *m* es el *file object* que representa un archivo de texto y *n* es un número entero mayor a 0, la invocación *cad = m.read(n)* lee una cantidad *n* de bytes del archivo los retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*. Si el archivo no llega a tener *n* bytes, se retorna la cadena vacía ("").

- [ ] d.
  Si *m* es el *file object* que representa un archivo de texto, la invocación *cad = m.read()* lee el contenido *completo* del archivo y lo retorna como una cadena de caracteres, asignando esa cadena en este caso en la variable *cad*.

Pregunta **10**

Finalizado

Puntúa 1 sobre 1

Si los algoritmos de *ordenamiento simples* tienen todos un tiempo de ejecución $O(n^2)$ en el peor caso, entonces: ¿cómo explica que las mediciones efectivas de los tiempos de ejecución de cada uno sean diferentes frente al mismo arreglo?

Seleccione una:

a.
La notación *Big O* rescata el término más significativo en la expresión que calcula el rendimiento, descartando constantes y otros términos que podrían no coincidir en los tres algoritmos.

b.
Los tiempos deben coincidir. Si hay diferencias, se debe a errores en los intrumentos de medición o a un planteo incorrecto del proceso de medición.

c.
La notación *Big O* no se debe usar para estimar el comportamiento en el peor caso, sino sólo para el caso medio.

d.
La notación *Big O* no se usa para medir tiempos sino para contar comparaciones u otro elemento de interés. Es un error, entonces, decir que los tiempos tienen "*orden n cuadrado*".

Pregunta **11**

Finalizado

Puntúa 0 sobre 1

¿Cuáles de las siguientes son características **correctas** del algoritmo *Shellsort*? (Más de una puede ser cierta... marque TODAS las que considere válidas)

Seleccione una o más de una:

a.
El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Selección Directa*, consistente en buscar iterativamente el menor (o el mayor) entre los elementos que quedan en el vector, para llevarlo a su posición correcta, pero de forma que la búsqueda del menor en cada vuelta se haga en tiempo logarítmico.

b.
El algoritmo *Shellsort* es complejo de analizar para determinar su rendimientos en forma matemática. Se sabe que para la serie de incrementos decrecientes usada en la implementación vista en las clases de la asignatura, tiene un tiempo de ejecución para el peor caso de $O(n^{1.5})$.

c.
El algoritmo *Shellsort* consiste en una mejora del algoritmo de *Inserción Directa* (o *Inserción Simple*), consistente en armar suconjuntos ordenados con elementos a distancia $h > 1$ en las primeras fases, y terminar con $h = 1$ en la última.

d.
En el caso promedio, el algoritmo *Shellsort* es tan eficiente como el *Heapsort* o el *Quicksort*, con tiempo de ejecución $O(n*log(n))$.

Pregunta **12**

Finalizado

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. Se ha indicado que se puede usar un arreglo rc de componentes, en el cual el casillero *rc[col] = fil* indica que la reina de la columna *col* está ubicada en la fila *fil*. ¿Cuáles de las siguientes configuraciones para el arreglo *rc* representan ***soluciones incorrectas*** para el problema de las *Ocho Reinas*? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- [ ] a.
  rc = [5, 3, 6, 0, 7, 1, 4, 2]

- [x] b.
  rc = [3, 5, 7, 0, 5, 1, 2, 4]

- [x] c.
  rc = [2, 0, 7, 4, 5, 1, 6, 3]

- [ ] d.
  rc = [4, 7, 3, 0, 2, 5, 1, 6]

Pregunta **13**

Finalizado

Puntúa 1 sobre 1

Dado un arreglo de *n* componentes... ¿qué significa decir que en el peor caso la cantidad de comparaciones que realiza el algoritmo de *búsqueda secuencial* es O(n) (o sea: del orden de n)?

Seleccione una:

- (•) a.
  Significa que en el peor caso el algoritmo hará n comparaciones.

- ( ) b.
  Significa que en el peor caso el algoritmo siempre hará menos de n comparaciones.

- ( ) c.
  Significa que en el peor caso el algoritmo no hará ninguna comparación.

- ( ) d.
  Significa que en el peor caso el algoritmo hará siempre más de n comparaciones.

Pregunta **14**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **FALSA** respecto de la serialización en Python?

Seleccione una:

a.
La *serialización* es un proceso por el cual el contenido de una variable se convierte automáticamente en una secuencia de bytes listos para ser almacenados en un archivo, y luego recuperarse desde el archivo y volver a crear la variable original.

b.
El módulo *pickle* es uno de los que brindan funciones para serializar en Python, pero no es el único (existen otros, tales como *json*).

c.
El método del módulo *pickle* que permite grabar una variable directamente es *pickle.dump()* y el que permite recuperar una variable serializada es *pickle.load()*.

d.
En Python no se pueden serializar variables de tipo simple (variables de tipo int, float, bool, etc.)

Pregunta **15**

Finalizado

Puntúa 0 sobre 1

El proceso de búsqueda del mayor en el siguiente programa está ligeramente cambiado con relación a su versión original (la tercera variante del problema de la búsqueda del mayor (Ficha 7): En la versión original, se comenzaba definiendo la variable *may* con el valor *None*, y ahora en esta versión que proponemos, hemos eliminado esa inicialización marcándola como un comentario (y sin borrarla, pero sólo por razones de claridad). ¿Hay algún problema en el planteo de esta pequeña variante? (Recuerde: si la instrucción está marcada como comentario, es lo mismo que si no estuviese).

```
__author__ = 'Catedra de AED'


print('Determinacion del mayor de una sucesion (variante 3)...')

# may = None
b = False
num = int(input('Ingrese un numero (con 0 finaliza): '))
while num != 0:
    if b == False:
        may = num
        b = True
    elif num > may:
        may = num
    num = int(input('Ingrese otro (con 0 finaliza): '))

print('El mayor es:', may)
```

Seleccione una:

a.
Sí, hay un problema: si en la primera carga antes del ciclo se ingresa un 0, el ciclo no ejecutará su bloque de acciones y la variable *may* quedará sin definir, provocando en ese caso que el programa se interrumpa y lance un error al intentar mostrar el valor de *may* en el programa.

b.
Sí, hay un problema: si todos los números que cargan fuesen negativos, en ese caso la variable *may* por defecto quedaría valiendo 0, y el programa mostraría un cero al final.

c.
No. No hay ningún problema. Funcionará correctamente en todos los casos, y la instrucción *may = None* era completamente innecesaria en la versión original.

d.
Sí, hay un problema: la variable *may* estará sin definir incluso dentro del ciclo. Cuando se intente asignar en ella el valor de *num* que corresponda a cada vuelta, el programa se interrumpirá y lanzará un error de variable no definida.

Pregunta **16**

Finalizado

Puntúa 1 sobre 1

Analice el siguiente script Python, en el cual se están grabando registros de tipo *Libro* en un archivo por serialización:

```
__author__ = 'Catedra de AED'

import pickle


class Libro:
    def __init__(self, cod, tit, aut):
        self.isbn = cod
        self.titulo = tit
        self.autor = aut


def display(libro):
    print('ISBN:', libro.isbn, end='')
    print(' - Título:', libro.titulo, end='')
    print(' - Autor:', libro.autor)


def test():
    print('Prueba de grabación de varios registros...')
    lib1 = Libro(2134, 'Fundación', 'Isaac Asimov')
    lib2 = Libro(5587, 'Fundación e Imperio', 'Isaac
Asimov')
    lib3 = Libro(3471, 'Segunda Fundación', 'Isaac
Asimov')

    fd = 'libros.dat'
    m = open(fd, 'wb')
    pickle.dump(lib1, m)
    pickle.dump(lib2, m)
    m.close()
    print('Se grabaron varios registros en el archivo',
fd)

    m = open(fd, 'rb')
    lib1 = pickle.load(m)
    lib2 = pickle.load(m)
    lib3 = pickle.load(m)
    m.close()

    print('Se recuperaron estos registros desde el
archivo', fd, ':')
    display(lib1)
    display(lib2)
    display(lib3)


if __name__ == '__main__':
    test()
```

¿Hay algún problema con este script?

Seleccione una:

a.
Sí. El problema es que no se pueden grabar registros en un archivo si el mismo fue abierto con *open()*. Debió usar *open_register_file()*.

b.
Sí. El problema es que el archivo fue abierto en modo wb cuando se pretendió grabar, pero en ese modo no se puede grabar.

c.
No hay nada de malo en este segmento.

d.
Sí. Se grabaron sólo dos registros en el archivo, pero luego se intentaron tres lecturas. La tercera fallará por encontrar el final del archivo en forma inesperada.

Pregunta **17**

Finalizado

Puntúa 1 sobre 1

¿Cuál de los siguientes tipos estructurados es un tipo de *secuencia mutable* en Python?

Seleccione una:

a.
Lista (list)

b.
Rango (range)

c.
Tuplas (tuples)

d.
Cadenas de caracteres (str)

Pregunta **18**

Finalizado

Puntúa 1 sobre 1

En el problema 51 de la Ficha 18 se introdujo la explicación de cómo generar en forma aleatoria un vector de registros. Uno de los campos de ese registro debía contener el número de patente de un vehículo (en formato argentino de 1994). En el programa original se mostró una forma simple de hacerlo. Se indican ahora algunas otras posibles maneras de lograrlo. ¿Cuáles de las siguientes ideas efectivamente logran generar una patente argentina? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

a.

```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) +
random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos)) +
str(random.choice(digitos)) +
str(random.choice(digitos))

patente = p1 + p2
print(patente)
```

b.

```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) +
random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = str(random.choice(digitos) +
random.choice(digitos) + random.choice(digitos))

patente = p1 + p2
print(patente)
```

c.

```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) +
random.choice(letras)

digitos = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
p2 = random.choice(digitos) + random.choice(digitos) +
random.choice(digitos)

patente = p1 + p2
print(patente)
```

d.

```
import random

letras = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
p1 = random.choice(letras) + random.choice(letras) +
random.choice(letras)

digitos = '0123456789'
p2 = random.choice(digitos) + random.choice(digitos) +
random.choice(digitos)

patente = p1 + p2
print(patente)
```

Pregunta **19**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es ***falsa*** en relación a las *Estrategias de Resolución de Problemas* (o *Estrategias de Planteo de Algoritmos*)?

Seleccione una:

a.
Un mismo problema podría ser resulto en base a dos o más estrategias de resolución diferentes, dando lugar a distintos algoritmos para ese mismo problema.

b.
Son técnicas y recomendaciones para el planteo de problemas que garantizan que se encontrará una solución, empleando la estrategia correcta para cada problema que se enfrente.

c.
Se trata de un conjunto de técnicas diversas que podrían ayudar a encontrar la solución a un problema, pero sin garantía de éxito, y aún si se llega a una solución, tampoco se garantiza que esa solución sea eficiente.

d.
Se trata de un conjunto de técnicas diversas que podrían ayudar a encontrar la solución a un problema, pero sin garantía de éxito.

Pregunta **20**

Finalizado

Puntúa 0 sobre 1

Analice el siguiente script en el cual se recorre una secuencia llamada sec con un ciclo *for*:

```
# suponga que sec es una secuencia (cadena, tupla, rango, etc.)
# ya inicializada
for x in sec:
    print('x:', x)
```

¿Qué hace este script si la secuencia *sec* estuviese inicialmente vacía?

Seleccione una:

a.
El ciclo *for* no hace ninguna repetición pero el script se interrumpe lanzando un mensaje de error.

b.
El ciclo *for* hace una y sólo una repetición y muestra el mensaje *x: None*.

c.
El ciclo *for* no hace ninguna repetición y el script termina normalmente sin mostrar nada en pantalla.

d.
El ciclo *for* entra en repetición infinita.

Pregunta **21**

Finalizado

Puntúa 0 sobre 1

Considere el problema del cálculo de los montos a pagar por el viaje de estudios de tres cursos de secundaria, tal como se presentó en la Ficha 10. El programa que se muestra más abajo, está replanteado de forma de usar variables globales (**ver Ficha 09**) en lugar de parámetros y retornos, y la función *mayor()* (que determinaba cuál era el curso con más cantidad de alumnos, y además, cuál era el monto a pagar por ese curso), está definida de la siguiente forma:

```
__author__ = 'Cátedra de AED'


def mayor():
    global may, may_cur
    if c1 > c2 and c1 > c3:
        may = m1
        may_cur = 'Primero'
    else:
        if c2 > c3:
            may = m2;
            may_cur = 'Segundo'
        else:
            may = m3
            may_cur = 'Tercero'


def montos():
    global m1, m2, m3
    m1 = c1 * 1360
    m2 = c2 * 1360
    m3 = c3 * 1360

    if c1 > 40:
        m1 = m1 - m1/100*5

    if c2 > 40:
        m2 = m2 - m2/100*5

    if c3 > 40:
        m3 = m3 - m3/100*5


def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = may / mtot * 100
    else:
        porc = 0


def test():
    global c1, c2, c3

    # título general y carga de datos...
    print('Cálculo de los montos de un viaje de
estudios...')
    c1 = int(input('Ingrese la cantidad de alumnos del
primer curso:  '))
    c2 = int(input('Ingrese la cantidad de alumnos del
segundo curso:  '))
    c3 = int(input('Ingrese la cantidad de alumnos del
tercer curso:  '))

    # procesos... invocar a las funciones en el orden
correcto...
    montos()
    mayor()
    porcentaje()

    # visualización de resultados
    print('El curso mas numeroso es el', may_cur)
    print('El monto del viaje del primer curso es:',
m1)
    print('El monto del viaje del segundo curso es:',
m2)
    print('El monto del viaje del segundo curso es:',
m3)
    print('El porcentaje del monto del mas numeroso en
el total es:', porc)


# script principal: sólo invocar a test()...
test()
```

¿Qué efecto causaría en el programa que la función *mayor()* fuese reemplazada por esta otra versión que se muestra a continuación, y qué cambios debería hacer el programador para que su programa siga

funcionando y cumpliendo con los mismos requerimientos? (Más de una respuesta puede ser válida. Marque todas las que considere correctas):

```
def mayor():
    global salida
    if c1 > c2 and c1 > c3:
        salida = 'Primero', m1
    else:
        if c2 > c3:
            salida = 'Segundo', m2
        else:
            salida = 'Tercero', m3
```

Seleccione una o más de una:

a.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[0] / mtot * 100
    else:
        porc = 0
```

b.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[1])
print('El monto del viaje del primer curso es:',
m1)
print('El monto del viaje del segundo curso es:',
m2)
print('El monto del viaje del segundo curso es:',
m3)
print('El porcentaje del monto del mas numeroso en
el total es:', porc)
```

c.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la visualización de resultados dentro de la función *test()*, que debería verse así:

```
# función test()... visualización de resultados...
print('El curso mas numeroso es el', salida[0])
print('El monto del viaje del primer curso es:',
m1)
print('El monto del viaje del segundo curso es:',
m2)
print('El monto del viaje del segundo curso es:',
m3)
print('El porcentaje del monto del mas numeroso en
el total es:', porc)
```

d.

En lugar de usar dos variables simples para almacenar los dos resultados, usa la tupla *salida* para guardarlos a ambos. Un cambio que debe hacer el programador es en la función *porcentaje()*, que debería verse así:

```
def porcentaje():
    global mtot, porc
    mtot = m1 + m2 + m3
    if mtot != 0:
        porc = salida[1] / mtot * 100
    else:
        porc = 0
```

Pregunta **22**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de los siguientes son conocidos algoritmos basados en la estrategia *Divide y Vencerás*? (Más de una respuesta puede ser correcta, por lo que marque todas las que considere válidas)

Seleccione una o más de una:

- [ ] a.
  Algoritmo *Heapsort* para ordenamiento de arreglos.

- [x] b.
  Algoritmo *Quicksort* para ordenamiento de arreglos.

- [ ] c.
  Algoritmo *Shellsort* para ordenamiento de arreglos.

- [x] d.
  Algoritmo *Mergesort* para ordenamiento de arreglos.

Pregunta **23**

Finalizado

Puntúa 0 sobre 1

¿Cuál es el motivo por el cual el *acoplamiento entre procesos e interfaz de ususario* es un problema en cuanto a las posibilidades de reuso de una función? [Más de una opción puede ser válida... marque ***todas*** las que considere correctas]

Seleccione una o más de una:

- a. Si en la función se supone que la interfaz de usuario estará soportada en la consola estándar, y luego se pretendiese reusarla pero en un contexto de ventanas y componentes visuales de alto nivel, entonces la función deberá ser rediseñada completamete.
- b. El concepto de acoplamiento entre procesos e interfaz de usuario no es aplicable a funciones, sino a diagramas de flujo y pseudocódigos. Por lo tanto, la pregunta en sí misma carece de sentido.
- c. En la medida de lo posible, el acoplamiento entre procesos e interfaz de ususario debería ser evitado, pero en algún momento el programador necesitará funciones que hagan el trabajo de gestionar la interfaz con el usuario. Tendrá entonces ciertas funciones con objetivos de control de interfaz, y cirtas otras funciones genéricas y reutilizables.
- d. Si la función procede a ingresar desde la interfaz de usuario el valor de alguna variable, pero el programador no necesitaba esa carga debido a que sus variables venían con valores definidos previamente, entonces deberá modificar la función para adaptarla al funcionamiento de su programa. Tendrá un problema similar si la función despliega en pantalla algún resultado.

Pregunta **24**

Finalizado

Puntúa 1 sobre 1

*¿Cuáles de las siguientes afirmaciones **serían ciertas** si el problema **P** vs NP se resolviese por la afirmativa (es decir, si se demostrase que P y NP son efectivamente iguales*? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

- a. Las máquinas deterministas serían al menos tan potentes como las no deterministas: Ambas resolverían eficientemente los mismos tipos de problemas
- b. Todo problema actualmente en NP, tendría solución de tiempo polinomial en una máquina determinista.
- c. Todo problema actualmente considerado intratable tendría solución de tiempo de ejecución polinomial en una máquina determinista.
- d. Todo problema actualmente en P admitiría también soluciones de tiempo exponencial en las mismas máquinas *deterministas* para las que esos problemas ya tenían soluciones de tiempo polinomial.

Pregunta **25**

Finalizado

Puntúa 1 sobre 1

Suponga que se quiere representar datos de distintos libros en un programa, y se propone utilizar un esquema basado en el uso de registros, como se muestra en el script de más abajo. ¿Hay algún problema con el programa siguiente?

```
class Libro:
    pass


def init(libro, cod, nom, aut):
    libro = Libro()
    libro.codigo = cod
    libro.titulo = nom
    libro.autor = aut


def write(libro):
    print('Datos del libro:')
    print('Código:', libro.codigo, ' - Título:',
libro.titulo, ' - Autor:', libro.autor)


def test():
    lib1 = Libro()
    init(lib1, 2345, 'El Aleph', 'Jorge Luis Borges')

    lib2 = Libro()
    init(lib2, 1267, 'Rayuela', 'Julio Cortázar')

    lib3 = Libro()
    init(lib3, 1928, 'El Túnel', 'Ernesto Sábato')

    write(lib1)
    write(lib2)
    write(lib3)


if __name__ == '__main__':
    test()
```

Seleccione una:

a.
No es correcto. Si la función *init()* está creando el registro (al hacer *libro = Libro()*) entonces no debe crearse el registro también en *test()*. En *test()* es suficiente con invocar a *init()* para crear cada registro.

b.
No es correcto. La función *init()* está creando nuevamente un registro vacío en su bloque de acciones (al hacer *libro = Libro()*). Cuando la función termina de ejecutarse, ese registro se pierde y el parámetro actual enviado a ella seguirá apuntando al registro vacío creado en *test()*. Cuando se invoque a la función *write()* se producirá entonces un error de intérprete porque intentará mostrar campos que no existen.

c.
No es correcto. La función *init()* no puede invocar a la función constructora (sólo puede invocarse en la función *test()* o en cualquier función que sea usada como función de arranque del programa).

d.
Sí. Es correcto y el esquema mostrado funciona sin problemas.

◄ Avisos

Ir a...

Examen Final - Registro de Calificación Final (Libres, Regulares y Promocionados - Cualquier año) ►