Comenzado el	jueves, 16 de diciembre de 2021, 08:58
Estado	Finalizado
Finalizado en	jueves, 16 de diciembre de 2021, 09:38
Tiempo empleado	40 minutos 1 segundos
Puntos	12/25
Calificación	**5** de 10 (**47**%)

Pregunta **1**

Finalizado

Puntúa 0 sobre 1

¿Cuál es la diferencia entre el *peor caso* y el *caso promedio* en el análisis de algoritmos?

Seleccione una:

a. El *peor caso* es la configuración de datos de entrada más favorable para el algoritmo, mientras que el *caso promedio* describe una configuración aleatoria de datos (no pensada ni para favorecer ni para desfavorecer al algoritmo)

b. El *peor caso* es la configuración de datos de entrada más favorable para el algoritmo, mientras que el *caso promedio* describe una configuración de datos pensada para desfavorecer al algoritmo.

c. El *peor caso* es la configuración de datos de entrada más desfavorable para el algoritmo, mientras que el *caso promedio* describe una configuración de datos pensada para favorecer al algoritmo.

d. El *peor caso* es la configuración de datos de entrada más desfavorable para el algoritmo, mientras que el *caso promedio* describe una configuración aleatoria de datos (no pensada ni para favorecer ni para desfavorecer al algoritmo).

Pregunta **2**

Finalizado

Puntúa 0 sobre 1

¿Cuáles de los siguientes elementos favorecen la *reutilización de una función*? [Aclaración: varias respuestas pueden ser válidas. Marque todas las que considere correctas]

Seleccione una o más de una:

a. La independencia de la función en relación a elementos definidos fuera de ella.

b. La correcta escritura de la función apegándose a principios, reglas y consejos de buenas prácticas (usar nombres descriptivos, emplerar comentarios, dejar espacios en blanco donde contribuya a una mejor lectura, etc).

c. La independencia de la función en referencia a procesos de carga de datos y visualizacicón de resultados.

d. La declaración de la función de forma que acepte parámetros para gestionar sus datos, y emplee el mecanismo de retorno para devolver sus resultados.

Pregunta **3**

Finalizado

Puntúa 0 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. ¿Cuáles de las siguientes afirmaciones son ***ciertas*** en relación a las ***diagonales inversas del tablero*** en el cual deben colocarse la reinas, suponiendo que el tablero es el normal del ajedrez, de 8 * 8? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- a. Las diagonales inversas pueden representarse con un arreglo *qid* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor *(col + fil)*, se haga coincidir el casillero *qid[(col + fil)]*.
- b. Las diagonales inversas pueden representarse con un arreglo *qid* de 15 componentes, en el que cada diagonal cuyos elementos tengan el mismo valor *(col - fil)*, se haga coincidir el casillero *qid[(col - fil) + 7]*.
- c. El valor de la resta entre el número de columna y el número de fila de cada componente de una diagonal inversa, es un número constante para cada diagonal, y los posibles valores están en el intervalo [-7..7]
- d. El valor de la suma entre el número de columna y el número de fila de cada componente de una diagonal inversa, es un número constante para cada diagonal, y los posibles valores están en el intervalo [0..14]

Pregunta **4**

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. ¿Cuáles podrían ser razones válidas por las que sería útil generar datos en forma automática? (Más de una puede ser cierta... marque TODAS las que considere correctas)

Seleccione una o más de una:

- a. Para facilitar la prueba de un programa antes de dejarlo en su versión definitiva.
- b. Para aplicar técnicas de generación de valores aleatorios y selección de valores aleatoriamente de una secuencia: aun cuanddo en un programa real los datos se carguen desde el teclado o desde un archivo, la generación automática es un recurso válido de aprendizaje y prueba.
- c. Para ganar tiempo (sobre todo en fase de prueba o cuando el programa es una simulación): la generacion automática ahorra mucho tiempo si los datos reales a cargar son muchos y/o si constan de combinaciones complicadas de valores.
- d. No hay una razón válida para hacerlo. Los datos siempre deben ser cargados por teclado o recuperados desde un archivo externo si el mismo existe.

Pregunta **5**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de los siguientes son *factores de eficiencia comunes* a considerar en el análisis de algoritmos? (Más de una respuesta puede ser válida... marque *todas* las que considere correctas).

Seleccione una o más de una:

- a. El tiempo de ejecución.
- b. La complejidad aparente del código fuente.
- c. La calidad aparente de la interfaz de usuario.
- d. El consumo de memoria.

Pregunta **6**

Finalizado

Puntúa 0 sobre 1

¿Qué ocurre si en un programa Python se usa el método *seek()* de un *file object* y se salta con ese método a un byte que está más allá del final del archivo?

Seleccione una:

- a. Se interrumpe el programa lanzando un error de la forma *EOFError*.
- b. El método *seek()* no puede saltar a un byte que esté fuera del archivo, por lo tanto, no hará nada.
- c. El *file pointer* del archivo quedará posicionado en ese byte, aunque esté fuera del archivo.
- d. El *file pointer* quedará posicionado en ese byte, y el tamaño del archivo se ajustará a ese byte.

Pregunta **7**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes situaciones haría que el algoritmo *Quicksort* degenere en su peor caso en cuanto al tiempo de ejecución, de orden n^2?

Seleccione una:

- a. Que el arreglo de entrada tenga sus elementos dispuestos de tal forma que cada vez que se seleccione el pivot en cada partición, resulte que ese pivot sea siempre el menor o el mayor de la partición que se está procesando.
- b. Que el arreglo esté ya ordenado, pero al revés.
- c. Que el arreglo esté ya ordenado, en la misma secuencia en que se lo quiere ordenar.
- d. El algoritmo Quicksort no tiene un peor caso $O(n^2)$. Su tiempo de ejecución siempre es $O(n*log(n))$.

Pregunta **8**

Finalizado

Puntúa 0 sobre 1

Oportunamente se presentó en clases el algoritmo de búsqueda binaria, el cual toma un arreglo ordenado, busca un valor *x* en el mismo, y retorna el índice de la casilla que lo contiene (si *x* está en el arreglo) o retorna -1 si *x* no está en el arreglo. Suponga que propone la siguiente variante para el algoritmo de búsqueda binaria:

```
def binary_search(v, x):
    # busqueda binaria... asume arreglo ordenado...
    izq, der = 0, len(v) - 1
    while izq <= der:
        c = (izq + der) // 2
        if x == v[c]:
            return c
        else:
            return -1

        if x < v[c]:
            der = c - 1
        else:
            izq = c + 1

    return -1
```

¿Funciona correctamente esta variante? Si no funciona, ¿cuál es el problema?

Seleccione una:

a. La variante propuesta funciona correctamente.

b. La variante propuesta no funciona correctamente: sólo llega a analizar el contenido de la casilla del centro del arreglo.

c. La variante propuesta funciona solamente si el valor x está en el arreglo (falla si x no está).

d. La variante propuesta provoca un error de intérprete y no llega a arrancar: el segundo if incluido dentro del ciclo nunca puede ejecutarse, ya que las dos ramas del if anterior terminan con una instrucción return.

Pregunta **9**

Finalizado

Puntúa 1 sobre 1

¿Qué se entiende por ***momento epoch*** cuando se trabaja con ciertas funciones para medir tiempos en Python, y cuál es ese momento?

Seleccione una:

a. Al momento a partir del cual se mide el tiempo transcurrido en Python. Ese momento coincide con la fecha y hora en la cual Python fue instalado en el computador del programador que mide el tiempo.

b. Al momento (arbitrariamente elegido) a partir del cual se mide el tiempo transcurrido en Python. Todos los sistemas operativos usan exactamente el mismo momento (misma fecha y hora) de inicio del tiempo.

c. Al momento (arbitrariamente elegido) a partir del cual se mide el tiempo transcurrido en Python. De acuerdo a cada sistema operativo, ese momento inicial del tiempo puede ser establecido en fechas y horas diferentes.

d. Al momento en el cual la licencia de uso libre de Python vencerá. Ese momento es exactamente el día 31/12/2030, a las 23:59.

Pregunta **10**

Finalizado

Puntúa 0 sobre 1

¿Qué hace el siguiente segmento de instrucciones en Python?

```
p = []
for i in range(3):
    p.append([])
    for j in range(2):
        p[i].append([])
        for k in range(4):
            p[i][j].append([])

p[2][1][3] = 'Prueba'
print(p)
```

Seleccione una:

- a. Crea un arreglo *p* de cuatro dimensiones, y almacena una cadena en el último casillero de ese arreglo (el casillero de índices más altos de cada dimensión posible).
- b. Crea un arreglo *p* de tres dimensiones, y almacena una cadena en el primer casillero de ese arreglo (el casillero de índices más bajos de cada dimensión posible).
- c. Crea un arreglo *p* de tres dimensiones, y almacena una cadena en el último casillero de ese arreglo (el casillero de índices más altos de cada dimensión posible).
- d. Lanza un error de intérprete: en Python no se pueden crear arreglos (variables de tipo *list*) de dimensión mayor a 2.

Pregunta **11**

Finalizado

Puntúa 0 sobre 1

¿Por qué es considerada una *mala idea* tomar como pivot al *primer elemento* (o al *último*) de cada partición al implementar el algoritmo *Quicksort*?

Seleccione una:

- a. Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese completamente desordenado.
- b. Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el arreglo estuviese ya ordenado o casi ordenado.
- c. Porque de esa forma aumenta el riesgo de caer en el peor caso, o aproximarse al peor caso, si el tamaño *n* del arreglo fuese muy grande.
- d. No es cierto que sea una mala idea. Ambas alternativas son tan buenas como cualquier otra.

Pregunta **12**

Finalizado

Puntúa 0 sobre 1

Considere el problema del Cambio de Monedas analizado en clases, y la solución mediante un Algoritmo Ávido también presentada en clases ¿Cuáles de las siguientes afirmaciones ***son ciertas*** en relación al problema y al algoritmo citado? (Más de una respuesta puede ser cierta, por lo que marque todas las que considere correctas...)

Seleccione una o más de una:

- a. El Algoritmo Ávido sugerido para el Problema del Cambio de Monedas falla si el valor x a cambiar tiene una moneda igual a x en el conjunto de valores nominales: en ese caso, el algoritmo provoca un error de runtime y se interrumpe.
- b. El Algoritmo Ávido sugerido para el problema del Cambio de Monedas funciona correctamente para cualquier conjunto de valores nominales de monedas, siempre y cuando ese conjunto incluya a la moneda de 1 centavo.
- c. Sea cual sea el algoritmo que se emplee, es exigible que exista la moneda de 1 centavo, pues de otro modo no habrá solución posible para muchos valores de cambio.
- d. Si el Problema de Cambio de Monedas no puede resolverse en forma óptima para un conjunto dado de monedas que incluya a la de 1 centavo, mediante el Algoritmo Ávido propuesto, entonces el problema no tiene solución.

Pregunta **13**

Finalizado

Puntúa 1 sobre 1

Analice el siguiente programa básico controlado por un menú de opciones. ¿Hay algún error en el planteo del mismo?

```
__author__ = 'Catedra de AED'


op = 1
while op != 3:
    # visualizacion de las opciones...
    print('1. Opcion 1')
    print('2. Opcion 2')
    print('3. Opcion 3')
    print('4. Salir')
    op = int(input('Ingrese el numero de la opcion elegida: '))

    # chequeo de la opcion elegida...
    if op == 1:
        print('Eligió la opcion 1...')
    elif op == 2:
        print('Eligió la opcion 2...')
    elif op == 3:
        print('Eligió la opcion 3...')
```

Seleccione una:

- a. No. No hay ningún error en el programa.
- b. Sí. El error es que dentro del bloque de acciones del ciclo, no hay una condición que controle si *op* es 4, por lo que si se ingresa un 4 el programa se interrumpirá con un mensaje de error.
- c. Sí. El error es que el ciclo no llega a hacer ninguna ejecución del bloque de acciones, ya la variable *op* comienza valiendo 1 y en ese momento la condición de control de ciclo se hace falsa.
- d. Sí. El error es que en la lista de opciones la opción de salida está marcada con el número 4, pero el ciclo corta cuando se ingresa la 3.
- e. Sí. El error es que el ciclo no controla si el valor ingresado en *op* es un número menor a 1 o mayor a 4, lo cual hace que si se carga un número incorrecto, el programa se interrumpirá con un mensaje de error.

Pregunta **14**

Finalizado

Puntúa 1 sobre 1

El siguiente programa crea y carga un arreglo *temp* con *n* valores de temperaturas medidas en diferentes momentos y luego procesa ese arreglo mediante la función *amplitud()* ¿Qué hace exactamente esa función?

```
__author__ = 'Cátedra de AED'


def read(temp):
    n = len(temp)
    for i in range(n):
        temp[i] = int(input('Temperatura[' + str(i) + ']: '))


def amplitud(temp):
    n = len(temp)
    my = mn = temp[0]
    for i in range(1, n):
        if temp[i] > my:
            my = temp[i]
        elif temp[i] < mn:
            mn = temp[i]

    return my - mn


def test():
    n = int(input('Cantidad de temperaturas a cargar: '))
    temp = n * [0.0]
    read(temp)

    d = amplitud(temp)

    print('Amplitud térmica:', d)


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

Seleccione una:

- a. Calcula y retorna la diferencia entre la mayor y la menor temperatura del arreglo *temp*.
- b. Calcula y retorna el promedio entre la mayor y la menor temperatura del arreglo *temp*.
- c. Calcula y retorna la la mayor temperatura del arreglo *temp*.
- d. Calcula y retorna la menor temperatura del arreglo *temp*.

Pregunta **15**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes es claramente falsa respecto de las características y propiedades de un objeto para manejar archivos en Python (un *file object*, tal como lo crea y lo retorna la función *open()*?

Seleccione una:

a. Un *file object* (o un *file-like object*) contiene métodos que permiten tanto grabar como leer datos del archivo representado, independientemente de que eso también puede hacerse con funciones de serialización incluidas en módulos separados (como *pickle* o *json*).

b. Un *file object* (o un *file-like object*) en última instancia permite interpretar el contenido de un archivo como si se tratase de un arreglo de bytes en memoria externa.

c. Un *file object* (o un *file-like object*) representa archivos en los que es posible acceder en forma directa a cualquiera de sus bytes mediante el método *seek()*.

d. Un *file object* (o un *file-like object*) representa sólo archivos de texto (y nunca archivos binarios).

Pregunta **16**

Finalizado

Puntúa 0 sobre 1

¿Qué efecto produce la ejecución del siguiente script, tal y como se muestra (exactamente con los valores 3 y 0 indicados como parámetro al invocar a la función *calcular()*)?

```
__author__ = 'Catedra de AED'


def calcular(a, b):
    if b != 0:
        c = a // b
        r = a % b
        return c, r

# script principal
a, b = calcular(3, 0)
print('Cociente:', a, 'Resto:', b)
```

Seleccione una:

a. Se produce un error al intentar ejecutar la instrución `a, b = calcular(3, 0)`.

b. Ejecuta sin problemas. Muestra el mensaje "*Cociente: None Resto: None*"

c. Ejecuta sin problemas. Muestra el mensaje "*Cociente: 0 Resto: 0*"

d. Ejecuta sin problemas. Muestra el mensaje "*None*"

Pregunta **17**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones con correctas en relación a conceptos elementales del análisis de algoritmos? (Más de una respuesta puede ser válida, por lo que marque todas las que considere correctas).

Seleccione una o más de una:

- [x] a. Los dos factores de eficiencia más comúnmente utilizados en el análisis de algoritmos son el tiempo de ejecución de un algoritmo y el espacio de memoria que un algoritmo emplea.
- [x] b. El análisis del *peor caso* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar la configuración más desfavorable posible de los datos que recibe.
- [x] c. El análisis del *caso promedio* es aquel en el cual se estudia el comportamiento de un algoritmo cuando debe procesar una configuración de datos que llegan en forma aleatoria.
- [ ] d. La notación Big O se usa para expresar el rendimiento de un algoritmo en terminos de una función que imponga una cota inferior para ese algoritmo en cuanto al factor medido (tiempo o espacio de memoria).

Pregunta **18**

Finalizado

Puntúa 1 sobre 1

¿En cuáles de las siguientes situaciones el uso de *recursividad* está efectivamente recomendado? (Más de una respuesta puede ser válida. Marque todas las que considere correctas).

Seleccione una o más de una:

- [ ] a. Nunca.
- [x] b. Recorrido y procesamiento de estructuras de datos no lineales como árboles y grafos.
- [ ] c. Siempre que se pueda escribir una definición recursiva del problema.
- [x] d. Generación y procesamiento de imágenes y gráficos fractales (figuras compuestas por versiones más simples de la misma figura original).

Pregunta **19**

Finalizado

Puntúa 0 sobre 1

El siguiente es un programa simple, que carga por teclado un número y usa una función para chequear si el mismo es cero o positivo (en cuyo caso, avisa con un mensaje) ¿Hay algún problema en el programa mostrado?

```
__author__ = 'Cátedra de AED'


def mostrar(n):
    if n < 0:
        return
    print('El número', n, 'es válido')


def test():
    n = int(input('Ingrese un número: '))
    mostrar(n)
    print('Programa terminado...')


# script principal
test()
```

Seleccione una:

a. No hay ningún problema: el programa hace exactamente lo esperado y no tiene elementos extraños en su planteo.

b. El programa lanza un error de intérprete: una función que usa un *return* sin valor indicado, no puede tomar parámetros.

c. El programa ejecuta sin problemas, pero SIEMPRE muestra el mensaje que indica que el número es válido (debería mostrarlo sólo si el número es mayor o igual a cero).

d. El programa lanza un error de intérprete: no se puede usar *return* en una función sin indicarle el valor a retornar.

Pregunta **20**

Finalizado

Puntúa 0 sobre 1

Considere el programa para el *Juego del Número Secreto* que se presentó en la Ficha 8. En ese programa se usa una bandera para marcar en el algoritmo si el número secreto fue encontrado o no. Nos proponemos tratar de eliminar el uso de esa bandera y simplificar la estructura del programa, y sugerimos el que se muestra más abajo emplando una *instrucción **break** para cortar el ciclo apenas se encuentre el número secreto*. ¿Funciona correctamente el programa que estamos sugiriendo? *Seleccione la respuesta que mejor describa lo que está pasando con el programa propuesto.*

```
__author__ = 'Catedra de AED'

import random


print('Juego del Número Secreto... Configuración Inicial...')
limite_derecho = int(input('El número secreto estará entre 1 y: '))
cantidad_intentos = int(input('Cantidad máxima de intentos que tendrá disponible: '))

# limites iniciales del intervalo de búsqueda...
izq, der = 1, limite_derecho

# contador de intentos...
intentos = 0

# el numero secreto...
secreto = random.randint(1, limite_derecho)

# el ciclo principal... siga mientras no
# haya sido encontrado el número, y la
# cantidad de intentos no llegue a 5...
while intentos < cantidad_intentos:
    intentos += 1
    print('\nEl numero está entre', izq, 'y', der)

    # un valor para forzar al ciclo a ser [1, N]...
    num = izq - 1

    # carga y validación del número sugerido por el usuario...
    while num < izq or num > der:
        num = int(input('[Intento: ' + str(intentos) + '] => Ingrese su numero: '))
        if num < izq or num > der:
            print('Error... le dije entre', izq, 'y', der, '...')

    # controlar si num es correcto, avisar y cortar el ciclo...
    if num == secreto:
        print('\nGenio!!! Acertaste en', intentos, 'intentos')
        break

    # ... pero si no lo es, ajustar los límites
    # del intervalo de búsqueda... y seguir...
    elif num > secreto:
        der = num
    else:
        izq = num

print('\nLo siento!!! Se te acabaron los intentos. El número era:', secreto)
```

Seleccione una:

a. Sí. Funciona correctamente para todos los casos.

b. No. No funciona bien: en la forma en que está planteado, se muestran en forma incorrecta los mensajes informando los límites del intervalo que contiene al número secreto en cada vuelta del ciclo, ya que las variables *izq* y *der* se actualizan en forma incorrecta.

c. No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *break*. Pero como *break* corta el ciclo y no el programa comppleto, entonces el programa continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.

d. No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto se mostrará el mensaje

avisándole que ganó pero el ciclo continuará pidiendo que se ingrese un número, hasta agotar el número de intentos disponible.

Pregunta **21**

Finalizado

Puntúa 0 sobre 1

¿Cuál es la mejora esencial que el algoritmo *Quicksort* realiza sobre el algoritmo *Bubblesort* o *Burbuja*?

Seleccione una:

- a. *Quicksort* no implementa ninguna mejora sustancial sobre *Bubblesort*.
- b. En las versiones analizadas en clases, *Quicksort* sólo usa un ciclo para recorrer el arreglo, mientras que *Bubblesort* usa dos.
- c. *Quicksort* acelera el cambio de posición tanto de los elementos menores como de los mayores, mientras que *Bubblesort* solo acelera a los mayores (o a los menores, dependiendo de la forma de implementación).
- d. *Quicksort* primero determina qué tan desordenado está el arreglo, mientras que *Bubblesort* procede directamente a ordenarlo

Pregunta **22**

Finalizado

Puntúa 1 sobre 1

¿Qué significa decir que un algoritmo dado tiene un tiempo de ejecución **O(1)**?

Seleccione una:

- a. El tiempo de ejecucíón es constante, sin importar la cantidad de datos.
- b. El tiempo de ejecución es lineal: si aumenta el número de datos, aumenta el tiempo en la misma proporción.
- c. El tiempo de ejecución siempre es de un segundo, sin importar la cantidad de datos.
- d. El tiempo de ejecución es logarítmico: a medida que aumenta el número de datos, aumenta el tiempo pero en forma muy suave.

Pregunta **23**

Finalizado

Puntúa 1 sobre 1

¿De qué forma se representa un dato que se graba en un archivo?

Seleccione una:

- a. Se representa en sistema alfanumérico y por cada dato se utilizan tantos bytes como sea necesario para representar ese dato en forma de cadena que contenga números y caracteres.
- b. Se representa en sistema decimal (base 10) y por cada dato se utilizan tantos bytes como sea necesario para representar ese dato en base 10.
- c. Se representa en formato de caracteres y por cada dato se utilizan tantos bytes como sea necesario para representar ese dato en forma de cadena de caracteres.
- d. Se representa en sistema binario (base 2) y por cada dato se utilizan tantos bytes como sea necesario para representar ese dato en base 2.

Pregunta **24**

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:

- a. 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.
- b. 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.
- c. 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.
- d. 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.

Pregunta **25**

Sin contestar

Puntúa como 1

Considere la siguiente función (vista en clases) basada en **backtracking** para resolución del Problema de las Ocho Reinas, e indique cuál de las opciones que se muestran es correcta:

```
def intend(col):
    global rc, qr, qid, qnd

    fil, res = -1, False
    while not res and fil != 7:
        res = False
        fil += 1
        di = col + fil
        dn = (col - fil) + 7
        if qr[fil] and qid[di] and qnd[dn]:
            rc[col] = fil
            qr[fil] = qid[di] = qnd[dn] = False

            if col < 7:
                res = intend(col + 1)
                if not res:
                    qr[fil] = qid[di] = qnd[dn] = True
                    rc[col] = -1
            else:
                res = True
    return res
```

Seleccione una:

- a. La función no es correcta porque la diagonal normal debe almacenar valores **(columna + fila)**.
- b. La función no es correcta porque no debe asignarse a **rc[col]** el valor de -1.
- c. La función es correcta y funciona sin problemas.
- d. La función no es correcta porque **qr** debe señalar las columnas disponibles y no las filas.

◄ Examen Final - Registro de Calificación Final (Libres, Regulares y Promocionados - Cualquier año) MODALIDADES PRESENCIAL Y A DISTANCIA

Ir a...

Examen Final Práctico [Regulares de cualquier año] MODALIDAD A DISTANCIA ►