Comenzado el	jueves, 19 de noviembre de 2020, 08:28
Estado	Finalizado
Finalizado en	jueves, 19 de noviembre de 2020, 09:02
Tiempo empleado	33 minutos 33 segundos
Puntos	17/25
Calificación	**7** de 10 (**70**%)

Pregunta **1**

Finalizado

Puntúa 1 sobre 1

¿Cuántos caracteres diferentes pueden representarse usando el estándar *ASCII 7*, y cuántos pueden representarse usando *ASCII 8*?

Seleccione una:

a.
32768 con ASCII 7 y 65536 con ASCII 8.

b.
256 con ASCII 7 y 128 con ASCII 8.

c.
128 con ASCII 7 y 256 con ASCII 8.

d.
4294967296 con ASCII 7 y 8589934592 con ASCII 8.

Pregunta **2**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes estrategias de obtención del pivot *es la más recomendable* para evitar que el algoritmo *Quicksort* se degrade en su peor caso $O(n^2)$ en cuanto a su tiempo de ejecución?

Seleccione una:

a.
En cada partición, obtener el pivot por la mediana de tres (ya sea la mediana entre el primero, el último y el central; o bien la mediana entre tres elementos aleatorios la partición).

b.
En cada partición, usar como pivot al valor de la primera casilla.

c.
En cada partición, usar como pivot al valor de la casilla central.

d.
En cada partición, usar como pivot al valor de la última casilla.

Pregunta **3**

Finalizado

Puntúa 1 sobre 1

Desde el punto de vista de la Teoría de la Complejidad los problemas pueden ser clasificados de acuerdo al tipo de ***Máquina Teórica*** para la cual se admitan soluciones para esos problemas. *¿Cuáles de las siguientes afirmaciones sobre estas Máquinas Teóricas* ***son ciertas***? (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

- [ ] a.
  En una Máquina Teórica Determinista el siguiente estado en el que estará esa máquina depende por completo del estado actual de la misma. Por lo tanto, el siguiente estado puede determinarse con precisión.

- [ ] b.
  Las Máquinas Teóricas No Deterministas tienen comportamiento aleatorio: Para un mismo problema estas máquinas podrían obtener resultados diferentes.

- [x] c.
  En una Máquina Teórica No Determinista es imposible determinar el siguiente estado en el que estará esa máquina sólo sabiendo el estado actual de la misma.

- [x] d.
  Todas las computadoras existentes hasta hoy, están basadas en un modelo de Máquina Teórica conocido como *Máquina de Turing*.

Pregunta **4**

Finalizado

Puntúa 1 sobre 1

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

Seleccione una:

- ( ) a.
  Son problemas en los que se pide obtener la mejor solución de entre varias posibles (y no necesariamente *cualquier* solución).

- ( ) b.
  Son problemas en los que se pide obtener una solución aproximada (aunque no necesariamente correcta o exacta)

- ( ) c.
  Son problemas en los que se pide obtener la solución específica para el problema planteado, d forma que la solución puede ser numérica.

- (•) d.
  Son problemas en los que sólo se admite como resultado un valor lógico (o valor de verdad: verdadero o falso).

Pregunta **5**

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 **6**

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 consumo de memoria.

b.
La complejidad aparente del código fuente.

c.
La calidad aparente de la interfaz de usuario.

d.
El tiempo de ejecución.

Pregunta **7**

Finalizado

Puntúa 1 sobre 1

Suponga la función *test()* (que se muestra más abajo) que toma como parámetro una matriz *mat* ya creada y cargada con elementos *int*. ¿Qué hace concretamente *la función test()*?

```
def test(mat):
    n = len(mat)
    m = len(mat[0])

    r = m * [0]
    for i in range(m):
        ac = 0
        for j in range(n):
            ac += mat[j][i]
        r[i] = ac / n

    return r
```

Seleccione una:

a.
Obtiene el valor promedio de cada columna, y retorna un vector con esos promedios.

b.
Obtiene el valor mayor de cada columna, y retorna un vector con esos mayores.

c.
Obtiene el valor promedio de cada fila, y retorna un vector con esos promedios.

d.
Obtiene el valor promedio de la matriz, y retorna ese valor.

Pregunta **8**

Finalizado

Puntúa 1 sobre 1

El siguiente programa crea y carga un arreglos *numeros* con *n* números enteros y luego procesa ese arreglo con la función control() ¿Qué hace exactamente esa función al ejecutarse?

```
__author__ = 'Cátedra de AED'


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


def control(numeros):
    n = len(numeros)
    for i in range(n-1):
        if numeros[i] != numeros[i+1]:
            return False
    return True


def test():
    n = int(input('Cantidad de números a cargar: '))
    numeros = n * [0]
    read(numeros)

    print()
    if control(numeros):
        print('El contenido del arreglo es correcto')
    else:
        print('El contenido del arreglo no es correcto')


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

Seleccione una:

a.
Retorna True si el arreglo está ordenado de mayor a menor.

b.
Retorna True si los números del arreglo no son todos iguales (hay dos o más números diferentes).

c.
Retorna True si todos los números del arreglo son iguales.

d.
Retorna True si el arreglo está ordenado de menor a mayor.

Pregunta **9**

Finalizado

Puntúa 0 sobre 1

Supongamos que se nos pide buscar el mayor de una secuencia de n números, tal como se analizó en la Ficha 7, pero de modo que ahora además de mostrar el mayor, se muestre también en qué orden apareció en la carga (o sea, en qué vuelta del ciclo apareció ese mayor) Por ejemplo, si la secuencia a cargar fuese {2, 5, 1, 3} entonces el mayor sería el 5 y apareció en el lugar 2 (o en el orden 2, o en la vuelta 2 del ciclo) ¿Está bien planteado el siguiente programa para cumplir con este nuevo requerimiento?

```
__author__ = 'Catedra de AED'


print('Determinacion del mayor de una sucesion (variante 1)...')

n = int(input('n: '))
for v in range(1, n+1):
    num = int(input('Numero: '))
    if v == 1:
        may, pos = num, 1
    elif num > may:
        may, pos = num, v

print('El mayor es:', may)
print('Se cargó en la vuelta:', pos)
```

Seleccione una:

a.
No. No está bien planteado: los resultados se están mostrando al revés, ya que el mayor estaría en *pos* y su posición de carga estaría en *may*. Hay que invertir las asignaciones de las tuplas dentro del ciclo.

b.
No. No está bien planteado: siempre muestra que la vuelta en que se cargó el mayor fue la 1 del ciclo.

c.
Sí, está correctamente planteado.

d.
No. No está bien planteado: si el valor mayor apareciese repetido más de una vez en la carga, la variable *pos* quedaría valiendo *None*.

Pregunta **10**

Finalizado

Puntúa 0 sobre 1

Considere el programa para el *Juego del Número Secreto* que se presentó en la Ficha 7. Ese programa se planteó originalmente sin funciones, y aquí lo mostramos redefinido para emplear funciones. Pero además, en la versión original del programa se usaba una *bandera* para marcar en el algoritmo si el número secreto fue encontrado o no; y nos proponemos ahora tratar de eliminar el uso de esa bandera y simplificar la estructura del programa. Sugerimos la modificación que se muestra más abajo empleando una *instrucción* ***return*** *para cortar el ciclo y la función 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.*

R

```
__author__ = 'Catedra de AED'

import random


def play_secret_number_game(limite_derecho,
cantidad_intentos):

    # 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')
            return

        # ... 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)


def test():
    print('Juego del Número Secreto... Configuración
Inicial...')
    ld = int(input('El número secreto estará entre 1 y:
'))
    ci = int(input('Cantidad máxima de intentos que tendrá
disponible: '))
    play_secret_number_game(ld, ci)


# script principal...
test()
```

Seleccione una:

a.
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.

b.
Sí. Funciona correctamente para todos los casos.

c.
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.

d.
No. No funciona bien: en la forma en que está planteado, cuando el jugador adivine el número secreto la función *play_secret_number_game()* mostrará correctamente el mensaje avisando que ganó y cortará el ciclo con la instrucción *return*. Pero luego la función continuará e inmediatamente mostrará también el mensaje avisando que el jugador perdió, provocando ambigüedad.

Pregunta **11**

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.
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.

c.
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.

d.
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.

Pregunta **12**

Finalizado

Puntúa 1 sobre 1

¿En cuáles de los siguientes casos el *algoritmo de divisiones sucesivas* es ***efectivamente muy lento e impráctico*** para determinar la primalidad de un número *n*? (En otras palabras: ¿qué caracterísitcas debe tener el número *n* para que el algoritmo caiga en un *peor caso* o cerca de un peor caso?) (Observación: más de una respuesta puede ser válida. Marque todas las que considere correctas)

Seleccione una o más de una:

- [x] a.
  Que *n* sea un número compuesto (no primo) pero tal que el primero de sus divisores sea a su vez un número muy grande.
- [ ] b.
  Que *n* sea un número par.
- [ ] c.
  Que *n* sea un número impar
- [x] d.
  Que *n* sea ya un número primo muy grande (por ejemplo: $n > 10^{20}$)

Pregunta **13**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes afirmaciones es **CIERTA** respecto de la operación para obtener el tamaño en bytes de un archivo en Python?

Seleccione una:

- (•) a.
  La función *os.path.getsize()* toma como parámetro el nombre físico de un archivo, y retorna la longitud en bytes de ese archivo.
- ( ) b.
  La función *open()* toma como parámetro el nombre físico de un archivo y el modo de apertura deseado, abre el archivo y retorna el tamaño en bytes del mismo.
- ( ) c.
  En Python no hay forma de obtener el tamaño en bytes de un archivo.
- ( ) d.
  El método *tell()* de los objetos manejadores de archivos, no toma parámetro alguno y retorna siempre el tamaño en bytes del archivo para el cual fue invocado.

Pregunta **14**

Finalizado

Puntúa 1 sobre 1

¿Cuál es la diferencia entre la *abstracción de datos y la abstracción funcional*?

Seleccione una:

a.
La abstracción de datos busca captar el conjunto de datos más relevante para representar un tipo abstracto, mientras que la funcional buscar determinar el conjunto de procesos relevante para esos datos.

b.
No existe un mecanismo de abstracción de datos ni un mecanismo de abstracción funcional. Existe sólo un mecanismo de abstracción, sin dividir en asbtracción de datos y abstracción funcional.

c.
Ninguna. Son sólo dos formas de referirse al mecanismo de abstracción.

d.
La abstracción de datos busca captar el conjunto de procesos relevante para el tipo abstracto que se quiere implementar, mientras que la funcional buscar determinar el conjunto de datos más relevante para implementar ese tipo.

Pregunta **15**

Finalizado

Puntúa 1 sobre 1

¿Qué tiene de malo el siguiente script en Python?

```
x1 = int(input('Primer valor: '))
x2 = int(input('Segundo valor: '))

if x1 = x2:
    print('Son iguales')
else:
    print('No son iguales')
```

Seleccione una:

a.
Al ejecutar, la condición sale siempre por verdadero.

b.
Los mensajes que muestra en ambas ramas están al revés.

c.
Al ejecutar, la condición sale siempre por falso.

d.
No compila: usa el operador = para comparar, cuando debió usar el == (doble igual).

Pregunta **16**

Finalizado

Puntúa 0 sobre 1

En la tabla siguiente se muestra un resumen comparativo entre las versiones iterativa (basada en ciclos) y recursiva del algoritmo de cálculo del término n-ésimo de la Sucesión de Fibonacci. Note que se han dejado sin completar los casilleros que corresponden al tiempo de ejecución para ambos casos.

Cálculo del término *n-ésimo* de Fibonacci [F(n)]	**Complejidad código fuente**	**Consumo de memoria**	**Tiempo de ejecución**
Versión iterativa	Aceptable	Constante (no depende de *n*)	????
Versión recursiva	Optima	Proporcional a *n* (lineal)	????

¿Cuál es la estimación para el tiempo de ejecución de ambos algoritmos?

Seleccione una:

a.
Versión iterativa: tiempo proporcional a *n* (lineal) - Versión recursiva: tiempo proporcional a *n* (lineal)

b.
Versión iterativa: tiempo proporcional a *n* (lineal) - Versión recursiva: tiempo proporcional a b^n (exponencial, para algún $b > 1$)

c.
Versión iterativa: tiempo constante (no depende de *n*) - Versión recursiva: tiempo proporcional a b^n (exponencial, para algún $b > 1$)

d.
Versión iterativa: tiempo proporcional a b^n (exponencial, para algún $b > 1$) - Versión recursiva: tiempo proporcional a *n* (lineal)

Pregunta **17**

Finalizado

Puntúa 1 sobre 1

*¿Cuáles de las siguientes afirmaciones **son ciertas** respecto de los problemas NP-Complete?* (Más de una respuesta puede ser válida... marque todas las que considere correctas...)

Seleccione una o más de una:

- [ ] a.
  Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de P al problema p1.

- [ ] b.
  Si p1 es un problema NP-Complete, entonces existe al menos una reducción polinómica que permite reducir el problema p1 a algún problema de NP.

- [x] c.
  Si p1 es un problema NP-Complete, entonces existen reducciones polinómicas que permiten reducir todo problema de NP a al problema p1.

- [ ] d.
  Si p1 es un problema NP-Complete, entonces existe al menos una reducción polinómica que permite reducir el problema p1 a algún problema de P.

Pregunta **18**

Finalizado

Puntúa 1 sobre 1

Cuáles de las siguientes afirmaciones son ciertas en cuanto al uso de ciclos o instrucciones repetitivas en Python? (Aclaración: más de una respuesta puede ser correcta... marque TODAS las que considere válidas...)

Seleccione una o más de una:

- [ ] a.
  Todos los ciclos básicos de Python, son de la forma [0, N].

- [ ] b.
  Todos los ciclos básicos de Python, son de la forma [1-N].

- [x] c.
  El ciclo *while* de Python ***sólo puede aplicarse*** cuando se desconoce la cantidad de repeticiones a realizar. Si la cantidad de repeticiones se conoce de antemano, ***debe*** aplicarse el *for*.

- [x] d.
  Python provee dos tipos de ciclos básicos: el *while* y el *for*.

Pregunta **19**

Finalizado

Puntúa 1 sobre 1

Considere el problema de las *Ocho Reinas* presentado en clases. ¿Cuáles de las siguientes afirmaciones son ***ciertas*** en relación a las ***diagonales 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.
  En cada una de las diagonales que se orientan como la contra-diagonal o diagonal inversa, es constante la suma entre el número de columna y el número de fila de cada uno de sus elementos.

- [x] b.
  En general hay dos tipos de diagonales: las normales (orientadas en la misma forma que la diagonal principal) y las inversas (orientadas en la misma forma que la contra-diagonal o diagonal inversa).

- [x] c.
  La cantidad ***total*** de diagonales que contiene el tablero (sumando todas las diagonales de todos los tipos posibles) es 30.

- [ ] d.
  En cada una de las diagonales que se orientan como la principal, es constante la resta entre el número de columna y el número de fila de cada uno de sus elementos.

Pregunta **20**

Finalizado

Puntúa 1 sobre 1

Analice el siguiente script en Python:

```
v = [2, 4, 1, 6]
v[0] = v[v[0]] * 3
print('v[0]:', v[0])
```

¿Cuál de las siguentes es correcta en relación al script mostrado?

Seleccione una:

- ( ) a.
  Se mostrará el mensaje: *v[0] = 2*.

- ( ) b.
  Se mostrará el mensaje: *v[0] = 12*.

- ( ) c.
  Se lanzará un error y el script se interrumpirá.

- (•) d.
  Se mostrará el mensaje: *v[0] = 3*.

Pregunta **21**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes es ***FALSA*** respecto del *file pointer* en un *file object*?

Seleccione una:

a.
El valor inicial del *file pointer* es asignado por la función *open()* al abrir el archivo.

b.
Cada vez que termina una operación de lectura o grabación en el archivo, el *file pointer* queda apuntando al byte siguiente al último que se leyó o grabó.

c.
La ***única forma*** en que puede cambiar el valor del *file pointer* de un archivo es invocando al método *seek()* del *file object* que lo maneja.

d.
El *file pointer* de un archivo puede apuntar a algún byte que esté fuera del archivo, *a la derecha del final del archivo*, pero no puede contener un valor negativo.

Pregunta **22**

Finalizado

Puntúa 1 sobre 1

Para cada una de las clases de complejidad que aparecen en la columna de la izquierda, seleccione la definición que mejor se le ajuste.

Clase NPC	Conjunto de todos los problemas NP-Complete.
Clase NP	Conjunto de todos los problemas de decisión que pueden ser resueltos
Clase P	Conjunto de todos los problemas de decisión que pueden ser resueltos
Clase EXP	Conjunto de todos los problemas de decisión que pueden ser resueltos
Clase NEXP	Conjunto de todos los problemas de decisión que pueden ser resueltos

Pregunta **23**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes afirmaciones ***son ciertas*** en relación a conceptos asociados con la *recursividad*?

Seleccione una o más de una:

- a.
  Una función recursiva no puede incluir más de una invocación a si misma en su bloque de acciones.

- b.
  Si una función es recursiva, entonces no debe incluir ningún ciclo en su bloque de acciones.

- c.
  A medida que se desarrolla la cascada de invocaciones recursivas, el stack segment se va llenando para darle soporte a cada instancia recursiva, y luego, cuando una instancia logra finalizar y se produce el proceso de vuelta atrás, el stack segment comienza a vaciarse.

- d.
  Cada instancia recursiva que es ejecutada, almacena dos grupos de datos en el stack segment: la dirección de retorno (a la que se debe regresar cuando termine la ejecución de esa instancia) y las variables locales que esa instancia de la función haya creado.

Pregunta **24**

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

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

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: una función que usa un *return* sin valor indicado, no puede tomar parámetros.

Pregunta **25**

Finalizado

Puntúa 1 sobre 1

¿Con qué nombre general se conoce en la *Teoria de la Complejidad* a un problema para el cual sólo se conocen algoritmos cuyo tiempo de ejecución es exponencial (o sea, problemas para los que todas las soluciones conocidas son algoritmos con tiempo $O(2^n)$?

Seleccione una:

- a. Problemas Intratables
- b. Problemas Inmanejables
- c. Problemas Irresolubles
- d. Problemas Imperdonables

◄ Avisos

Ir a...

Enunciado Examen Final Práctico [Regulares - Cualquier año] ►