Comenzado el	jueves, 17 de diciembre de 2020, 08:25
Estado	Finalizado
Finalizado en	jueves, 17 de diciembre de 2020, 09:00
Tiempo empleado	34 minutos 33 segundos
Puntos	22/25
Calificación	**9** de 10 (**89**%)

Pregunta **1**

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 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.

- [ ] 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 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.

- [x] 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 **2**

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 no se conoce solución alguna.

c.
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.

d.
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.

Pregunta **3**

Finalizado

Puntúa 1 sobre 1

¿Por qué motivo el algoritmo Bubblesort para ordenamento de un arreglo usa una *bandera de corte* en la versión presentada en las fichas de clase?

Seleccione una:

a.
La bandera de corte se usa para garantizar que el arreglo quede ordenado.

b.
No es cierto que la versión vista en clases use una bandera de corte.

c.
La bandera de corte se usa para terminar el proceso apenas se detecte que en la pasada actual no hubo intercambios, para ahorrar tiempo.

d.
La bandera de corte se usa para determinar si el ordenamento debe hacerse de manor a mayor (bandera = True) o de mayor a menor (bandera = False)

Pregunta **4**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes expone correctamente la estrategia general que debe llevar a cabo un *proceso de alta de registros en un archivo, suponiendo que NO se admiten registros con clave repetida* en ese archivo?

Seleccione una:

a.
1) Abrir el archivo *m* en modo 'ab'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro r que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el archivo.

b.
1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro r que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el arrchivo.

c.
1) Abrir el archivo *m* en modo 'a+b'. 2) Determinar si el archivo ya contiene o no a un registro con la misma clave que el registro r que se quiere agregar. 3) Si ya existe un registro con esa clave, rechazar el alta. Si no existe, asegurarse de marcar el registro como *eliminado* (*activo = False*), y grabar finalmente el registro en el archivo.

d.
1) Abrir el archivo *m* en modo 'a+b'. 2) Asegurarse de marcar el registro como *no eliminado* (*activo = True*), y grabar finalmente el registro en el archivo.

Pregunta **5**

Finalizado

Puntúa 1 sobre 1

Considere la función presentada en clases para calcular mediana entre el primero, el central y el último elemento de una partición del arreglo *v* delimitada por los elementos en las posiciones *izq* y *der*:

```
def get_pivot_m3(v, izq, der):
    # calculo del pivot: mediana de tres...
    central = int((izq + der) / 2)

    if v[der] < v[izq]:
        v[der], v[izq] = v[izq], v[der]

    if v[central] < v[izq]:
        v[central], v[izq] = v[izq], v[central]

    if v[central] > v[der]:
        v[central], v[der] = v[der], v[central]

    return v[central]
```

El tamaño de la partición analizada es entonces *n = der - izq + 1* elementos. ¿Cuál es el tiempo de ejecución de esta función, expresado en *notación O* y de acuerdo a ese valor de *n*?

Seleccione una:

- a. O(1)
- b. $O(n^2)$
- c. O(n)
- d. O(log(n))

Pregunta **6**

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: $O(n^2)$ comparaciones.
- c. Peor caso: O(log(n)) comparaciones.
- d. Peor caso: O(1) comparaciones.

Pregunta **7**

Finalizado

Puntúa 1 sobre 1

¿En cuáles de los siguientes casos es aplicable el algoritmo de *Fusión de Arreglos* que se describió en las fichas de clases para producir un tercer arreglo ordenado? (Seleccione todas las respuestas que considere correctas)

Seleccione una o más de una:

- [ ] a.
  Uno de los arreglos originales debe estar ordenado y el otro arreglo puede estar desordenado.

- [ ] b.
  El proceso es aplicable sin importar si los dos arreglos originales están ordenados o no, ya que el algoritmo de fusión analizado primero ordena esos dos arreglos, y luego procede a la fusión.

- [ ] c.
  Los arreglos originales deben estar desordenados.

- [x] d.
  Los arreglos originales deben estar ordenados de menor a mayor si se quiere producir un tercer arreglo ordenado de menor a mayor.

Pregunta **8**

Finalizado

Puntúa 1 sobre 1

¿Hay algún error en el siguiente script de instrucciones en Python 3?

```
nombre = input('Nombre: ')
edad = int(input('Edad: '))
print('Datos recibidos - Nombre: ', Nombre, 'Edad: ',
Edad)
```

Seleccione una:

- a.
  El error es que la función input() de Python 3 no puede usarse para cargar cadenas de caracteres en forma directa.

- b.
  El error es el uso de la función int() en la segunda carga: no existe tal función en Python 3.

- c.
  No hay ningún error.

- d.
  El error es que las variables *edad* y *nombre* se definieron en minúsculas al hacer la carga, y luego se usaron con mayúscula en la primera letra (*Edad* y *Nombre*) al hacer las visualizaciones.

Pregunta **9**

Finalizado

Puntúa 1 sobre 1

Suponga que se desea desarrollar un programa que cargue dos números, y muestre el mayor de los números pero también el valor de multiplicar por dos y por tres a ese mayor. ¿Está bien planteado el programa que sigue?

```
__author__ = 'Cátedra de AED'


def cargar():
    a = int(input('A: '))
    b = int(input('B: '))
    return a, b


def comparar(a, b):
    if a > b:
        may = a
    else:
        may = b
    return may


def procesar(may):
    doble = 2 * may
    triple = 3 * may
    return doble, triple


def mostrar(may, doble, triple):
    print('El mayor es:', may)
    print('El doble del mayor es:',
doble)
    print('El triple del mayor es:',
triple)


# script principal...
a, b = cargar()
doble, triple = procesar(may)
may = comparar(a, b)
mostrar(may, doble, triple)
```

Seleccione una:

a.
El programa está mal planteado: la visualización de los resultados finales DEBE hacerse en el script principal y NO en una función separada.

b.
Sí. El programa está correctamente planteado.

c.
El programa está mal planteado: la función *comparar()* debería ser invocada antes de invocar a la función *procesar()*. Así como está, lanza un error al intentar ejecutar la función *procesar()* pues no reconoce a la variable may.

d.
El programa lanza un error de intérprete: el script principal no puede consistir sólo en llamadas a funciones.

Pregunta **10**

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 en la lista de opciones la opción de salida está marcada con el número 4, pero el ciclo corta cuando se ingresa la 3.

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 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.

e.
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.

Pregunta **11**

Finalizado

Puntúa 0 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:

- 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.

- b.
  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.

- c.
  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.

- 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 **12**

Finalizado

Puntúa 1 sobre 1

El siguiente programa utiliza variables definidas como globales (**ver Ficha 09**),¿Qué hace exactamente el programa (y por qué hace eso?

```
__author__ = 'Catedra de AED'


def funcion01():
    a = b * 2


def funcion02():
    a = b + c


def test():
    global a, b, c

    a, b, c = 10, 20, 30
    funcion01()
    funcion02()
    print('Valor final de a:', a)


# script principal
test()
```

Seleccione una:

a.
Lanza un error de intérprete al intentar ejecutarlo: una función no puede declarar variables locales con el mismo nombre que una variable definida como global.

b.
Muestra el mensaje: *Valor final de a: 50* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion02()*.

c.
Muestra el mensaje: *Valor final de a: 10* - Porque la variable *a* cuyo valor se muestra en *test()*, es *global* y queda valiendo el valor 10 asignado en la misma *test()*.

d.
Muestra el mensaje: *Valor final de a: 40* - Porque la variable *a* queda valiendo el valor asignado al invocar a *funcion01()*.

Pregunta **13**

Finalizado

Puntúa 1 sobre 1

¿Cuál de los siguientes scripts ***NO*** creará una tupla conteniendo exclusivamente los caracteres de la palabra 'Python', en ese mismo orden'? (Repetimos: la pregunta es cuál de todos ***NO*** creará la tupla pedida...)

Seleccione una:

a.
```
t3 = ('P', 'y', 't', 'h', 'o', 'n')
```

b.
```
t4 = ()
for c in 'Python':
    t4 += c,
```

c.
```
t2 = 'P', 'y', 't', 'h', 'o', 'n'
```

d.
```
t1 = tuple('Python')
```

e.
```
Python = 'Java'
t5 = tuple(Python)
```

Pregunta **14**

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] = 12*.

b.
Se lanzará un error y el script se interrumpirá.

c.
Se mostrará el mensaje: *v[0] = 2*.

d.
Se mostrará el mensaje: *v[0] = 3*.

Pregunta **15**

Finalizado

Puntúa 1 sobre 1

¿Qué elementos son necesarios para que una *función recursiva* se considere *bien planteada*?

Seleccione una:

a.
La función deben tener al menos una invocación a si misma en su bloque de acciones.

b.
La función debe tener al menos un ciclo en su bloque de acciones, y ese ciclo debe estar planteado de forma tal que nunca entre en un lazo infinito.

c.
La función debe tener al menos una invocación a si misma en su bloque de acciones, y antes de esas invocaciones debe tener una o más condiciones de control que permitan interrumpir el proceso recursivo si se ha llegado a alcanzar alguna de las situaciones triviales o base del problema.

d.
La función debe tener al menos una invocación a si misma en su bloque de acciones, y después de esas invocaciones debe tener una o más condiciones de control que permitan interrumpir el proceso recursivo si se ha llegado a alcanzar alguna de las situaciones triviales o base del problema.

Pregunta **16**

Finalizado

Puntúa 1 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.
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.

b.
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.

c.
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.

d.
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.

Pregunta **17**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de las siguientes son características propias del método *write()*, 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 y *cad* es una cadena de caracteres, la invocación *m.write(cad)* graba la cadena contenida en *cad* pero *no agrega* un caracter de salto de línea al final, a menos que *cad* ya lo contenga previamente.

- [ ] b.
  Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación *m.write(cad)* graba la cadena contenida en *cad* en forma directa y simple en un archivo de texto.

- [ ] c.
  Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación *m.write(cad)* graba la cadena contenida en *cad* sin importar en qué modo haya sido abierto el archivo (sólo importa que se haya indicado que es de texto usando una 't' en el modo de apertura, y no una 'b').

- [x] d.
  Si *m* es el *file object* que representa un archivo de texto y *cad* es una cadena de caracteres, la invocación *m.write(cad)* graba la cadena contenida en *cad* y retorna la cantidad de caracteres que efectivamente grabó.

Pregunta **18**

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*) en última instancia permite interpretar el contenido de un archivo como si se tratase de un arreglo de bytes en memoria externa.

- b.
  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()*.

- c.
  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*).

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

Pregunta **19**

Finalizado

Puntúa 1 sobre 1

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

Seleccione una o más de una:

- a.
  El algoritmo Heap Sort arma el heap o grupo de ordenamiento con el que se ordena el vector, pero lo hace en el mismo vector, sin usar memoria extra.

- b.
  El algoritmo Heap Sort tiene es muy eficiente en tiempo de ejecución, tanto para el caso promedio como para el peor caso.

- c.
  El algoritmo Heap Sort utiliza una cantidad de memoria adicional igual al tamaño del arreglo, para armar el heap o grupo de ordenamiento con el que se ordena el vector.

- d.
  El algoritmo Heap Sort se basa en encontrar sucesivamente el menor (o el mayor) de entre los elementos que quedan, para llevar ese valor a su casillero final, pero de forma que la búsqueda del menor (o el mayor) en cada vuelta se haga en forma muy veloz.

Pregunta **20**

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 **qr** debe señalar las columnas disponibles y no las filas.

b.
La función no es correcta porque la diagonal normal debe almacenar valores **(columna + fila)**.

c.
La función es correcta y funciona sin problemas.

d.
La función no es correcta porque no debe asignarse a **rc[col]** el valor de -1.

Pregunta **21**

Finalizado

Puntúa 1 sobre 1

¿Qué hace el siguiente programa en Python?

```
__author__ = 'Cátedra de AED'


def comparar():
    if a > p:
        print('El valor', a, 'es mayor...' )

    if b > p:
        print('El valor', b, 'es mayor...')

    if c > p:
        print('El valor', c, 'es mayor...')


def promedio():
    global p
    p = (a + b + c) / 3


a = int(input('A: '))
b = int(input('B: '))
c = int(input('C: '))

promedio()
comparar()

print('Programa terminado...')
```

Seleccione una:

a.
Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra sólo los valores de de las variables que sean mayores al promedio.

b.
Lanza error de intérprete: las variables *a*, *b*, *c* y *p* están definidas incorrectamente (debieron definirse todas en el script principal, debajo de las funciones).

c.
Calcula el promedio de los valores de *a*, *b* y *c*, y luego muestra los valores de todas las variables.

d.
Muestra los valores de *a*, *b*, *c*, y también el *promedio* de los valores de esas tres variables.

Pregunta **22**

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. El tiempo de ejecución.
- d. La calidad aparente de la interfaz de usuario.

Pregunta **23**

Finalizado

Puntúa 1 sobre 1

¿Cuáles de los siguientes son *factores esenciales* a tener en cuenta cuando se realiza el *análisis de la eficiencia de un algoritmo* (por ejemplo, para efectuar una comparación de rendimiento entre dos algoritmos que resuelven el mismo problema)?

Seleccione una o más de una:

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

Pregunta **24**

Finalizado

Puntúa 1 sobre 1

¿Cuál de las siguientes resume en forma correcta la idea general de la estrategia *Divide y Vencerás* para resolución de problemas?

Seleccione una:

a.
El conjunto de *n* datos se divide en subconjuntos de aproximadamente el mismo tamaño (n/2, n/3, n/4, etc.). Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

b.
El conjunto de *n* datos se divide en subconjuntos de cualquier tamaño, sin importar si los tamaños de cada subconjunto coinciden entre sí. Luego se aplica recursión para procesar cada uno de esos subconjuntos. Finalmente se unen las partes que se acaban de procesar para lograr el resultado final.

c.
Se aplica una regla simple que parece ser beneficiosa, sin volver atrás ni medir las consecuencias de aplicar esa regla, con la esperanza de lograr el resultado óptimo al final.

d.
Se usa una tabla para almacenar los resultados de los subproblemas que se hayan calculado, y luego cuando algún subproblema vuelve a aparecer se toma su valor desde la tabla, para evitar pérdida de tiempo.

Pregunta **25**

Finalizado

Puntúa 1 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* 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 $O(n^{1.5})$.

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

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.
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.

◄ Avisos

Ir a...