Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2023-2

Examen

14 de diciembre 2023

Instrucciones

- La evaluación consta de 30 preguntas de alternativas. Para obtener el 7.0 en la evaluación, se deben tener 28 preguntas correctas.
- Recibirás una hoja de respuestas que deberás rellenar con tus datos y las respuestas correspondientes a cada alternativa. Sólo se corregirá la hoja de respuestas. **Ten mucha precaución de anotar correctamente tus datos**.
- Cada pregunta tiene únicamente 1 alternativa correcta. Responder con 2 o más alternativas implicará dejar inválida esa pregunta y se considerará incorrecta. Además, cada pregunta presenta el mismo puntaje y no se descontará por respuesta incorrecta.
- Para quienes justificaron su ausencia en el *midterm* con su Unidad Académica y fue aceptada su justificación, deben tener las 30 preguntas correctas para lograr el 7.0 equivalente al apartado de alternativas del *midterm*. Luego, deberán quedarse después del examen para rendir una evaluación oral e individual que será en reemplazo del apartado de desarrollo del *midterm*.

1. En el contexto de un cliente y servidor que se comunican a través de *networking*, ¿cuáles de las siguientes acciones son las **mínimas** que se deben implementar para asegurar un **correcto** manejo de los mensajes?

   I. Codificación

   II. Decodificación

   III. Encriptación

   IV. Desencriptación

   A) I y II

   B) I y III

   C) II y IV

   D) III y IV

   E) I, II, III y IV

2. Un programador utilizó un bloque de `try/except` para capturar una excepción del tipo `ValueError` y registrar dicho error en un archivo `.txt`. Ahora, además de registrar el error, desea volver a levantar la misma excepción con sus respectivos argumentos.

   ¿Cuál afirmación es **correcta** respecto a este caso?

   A) Una vez capturado el `ValueError`, no es posible levantar la misma excepción.

   B) Usando `print` se puede lograr este objetivo.

   C) Usando `raise` se puede lograr este objetivo.

   D) Si el bloque `try/except` está dentro de una función, con `return` se puede lograr el objetivo.

   E) Usando `emit` se puede lograr este objetivo.

3. Respecto a excepciones, ¿cuál o cuáles de los siguientes afirmaciones son **correctas**?

   I. La sentencia `finally` permite ejecutar un segmento de código independiente de si ocurre o no un error dentro del bloque del `try`.

   II. Al capturar excepciones, es posible no especificar la excepción que se desea capturar.

   III. Los errores de sintaxis únicamente se pueden detectar mientras el programa ya está ejecutando cada línea del código.

   A) Solo I

   B) Solo II

   C) Solo III

   D) I y II

   E) II y III

4. Según los contenidos vistos en el curso, ¿en cuál o cuáles de estos casos es **recomendable** el uso de **TCP**?

   I. Subida de cambios de un repositorio local a uno remoto en Github.

   II. Una videollamada entre dos o más participantes.

   III. Edición de documentos de texto en la nube entre dos o más personas, como Word Online o Google Docs.

   A) Solo I

   B) Solo II

   C) I y II

   D) I y III

   E) I, II y III

5. Dado el caso de un cliente que se desea conectar a un servidor mediante *networking*, ¿cuál de las siguientes afirmaciones es **correcta** respecto al siguiente código?

```
1 sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
2 sock.connect(("localhost", 3000))
```

   A) El argumento `"localhost"` indica la dirección IP en que se está ejecutando el **servidor**, y el argumento `3000` indica el puerto que está usando el **servidor** para la comunicación.

   B) El argumento `"localhost"` indica la dirección IP en que se está ejecutando el **cliente**, y el argumento `3000` indica el puerto que está usando el **cliente** para la comunicación.

   C) El argumento `"localhost"` indica la dirección IP en que se está ejecutando el **servidor**, y el argumento `3000` indica el puerto que está usando el **cliente** para la comunicación.

   D) El argumento `"localhost"` indica la dirección IP en que se está ejecutando el **cliente**, y el argumento `3000` indica el puerto que está usando el **servidor** para la comunicación.

   E) El argumento `"localhost"` indica la dirección IP en que se está ejecutando tanto el **servidor** como el **cliente**, y el argumento `3000` indica el puerto que está usando tanto el **servidor** como el **cliente** para la comunicación.

6. ¿Cuál de las siguientes afirmaciones es **incorrecta** respecto a la información enviada mediante *sockets* en Python?

   A) Solo es posible enviar *bytes* o *bytearray*.

   B) Es posible enviar directamente instancias de objetos serializadas con `pickle`.

   C) Es posible enviar directamente instancias de diccionarios serializadas con `json`.

   D) Los mensajes de gran tamaño pueden ser enviados por partes de menor tamaño cada una.

   E) Para enviar *floats*, primero hay que convertirlos a otro tipo de dato.

7. Respecto al método `.recv()` de *socket* en una comunicación TCP desde el cliente al servidor, este cumple la función de:

   A) Recibir información desde el servidor.

   B) Convertir toda la data del servidor a una cadena de texto.

   C) Capturar los errores de conexión, retornando un objeto de la clase `ConnectionError`.

   D) Limitar la cantidad máxima de datos a recibir desde el servidor durante la conexión.

   E) Enviar información al servidor.

8. El siguiente *script*, en Python, busca que un cliente utilice *sockets* para conectarse a `'iic2333.ing.puc.cl'`, una página web programada en otro lenguaje. Luego, si logra conectarse, enviará una solicitud para obtener el contenido de la página web.

```python
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
sock.connect(('iic2333.ing.puc.cl', 80))
sock.sendall('GET / HTTP/1.1\nHost: iic2333.ing.puc.cl\n\n'.encode('utf-8'))
data = sock.recv(4096)
print(data)
sock.close()
```

   Asumiendo que no existen errores de sintaxis, ¿cuál afirmación es **correcta** sobre este código?

   A) El *socket* será incapaz de conectarse al servidor porque este último no utiliza Python.

   B) El cliente está haciendo una solicitud del tipo `POST` al servidor.

   C) Si el servidor nunca responde a la solicitud del cliente, este se quedará esperando hasta que forcemos el cierre al programa.

   D) Si el contenido de la página web es mayor a 4096 *bytes*, fallará el código.

   E) Si el contenido de la página web es menor a 4096 *bytes*, fallará el código.

9. ¿Cuál o cuáles de las siguientes acciones pueden realizarse mediante el módulo `re` para expresiones regulares de Python?

   I. Verificar si algún *substring* de un texto cumple con la expresión regular.

   II. Listar todos los *substrings* de un texto que cumplan con una expresión regular.

   III. Reemplazar todos los *substrings* de un texto que cumplan con una expresión regular por un texto diferente.

   A) Solo I

   B) Solo II

   C) I y II

   D) I y III

   E) I, II y III

10. Respecto a las expresiones regulares, es **incorrecto** afirmar que:

    A) Son secuencias especiales de caracteres que permiten buscar *strings* dentro de otro *string*.

    B) Poseen una sintaxis específica para poder ser expresiones regulares válidas.

    C) Poseen un conjunto de caracteres especiales para definir patrones más generales. Por ejemplo: `.`, `$`, `?`.

    D) Una limitación que tienen, es que no pueden encontrar los caracteres especiales dentro de otro *string*. Por ejemplo, no puede encontrar `.`, `$` o `?`.

    E) Una limitación que tienen, es que no pueden capturar patrones que tengan una cantidad aleatoria de caracteres.

11. Respecto a los *strings* es **correcto** afirmar que:

    I. Son mutables.

    II. Son indexables.

    III. Pueden ser utilizados como llaves de diccionarios.

    IV. Un *string* al ser codificado y luego decodificado, puede verse distinto si es que se eligen distintos *encodings*.

    A) Solo I

    B) Solo II

    C) I y III.

    D) II y III

    E) II, III y IV

12. En expresiones regulares usamos: `()` para agrupar, `*` para indicar que el carácter o grupo está 0 o más veces y `|` para definir un *or*. Dado lo anterior, ¿**cuántos** *substrings* hacen *match* entre la Frase y la Expresión?

    Frase: `'¡Aaahhh! La cama de mi casa está ocupada por mi gata'`
    Expresión: `r'a(s|m)*a'`

    A) 3

    B) 5

    C) 0

    D) 4

    E) 2

13. Según los contenidos del curso, ¿en cuál o cuáles de los siguientes escenarios se recomienda implementar un *context manager*, usando la sentencia `with` para acceder al recurso?:

    I. Actualizar la información visualizada a través de una interfaz gráfica.

    II. Leer y/o escribir datos en un archivo.

    III. Manejar excepciones en el código.

    IV. Manejar el acceso a un recurso compartido por múltiples *threads*.

    A) Solo I

    B) Solo II

    C) I y II

    D) II y IV

    E) I, II, III y IV

14. En el contexto de las consultas que se realizan por medio una API, ¿cuál o cuáles elementos de una *response* no están presentes en una *request*?

    I. *Status code*

    II. *Header*

    III. *Params*

    IV. *Body*

    A) Solo I

    B) Solo III

    C) Solo IV

    D) I y IV

    E) I, II, III y IV

15. ¿Cuál de la siguientes alternativa relaciona **incorrectamente** el caso descrito con el grupo de códigos de estado HTTP que respondería una API?

    A) Solicitar el horóscopo para `"Acuario"` y que la API responda con el texto asociado a dicho signo. Grupo de códigos de estado HTTP: `2XX`.

    B) Solicitar el horóscopo para `"Siuu"` y que la API responda que no encuentra dicho signo en la base de datos. Grupo de códigos de estado HTTP: `4XX`.

    C) Agregar un nuevo signo a la base de datos, pero la API responde que no tengo permisos para realizar dicha acción. Grupo de códigos de estado HTTP: `5XX`.

    D) Solicitar el horóscopo para `"Siuu"` y que la API se cae por un error interno. Grupo de códigos de estado HTTP: `5XX`.

    E) Cambiar el horóscopo para `"Sagitario"` por otro texto y que la API responde que el texto fue actualizado correctamente. Grupo de códigos de estado HTTP: `2XX`.

16. Si se decide utilizar un *endpoint* de una API que cumple el protocolo HTTP, ¿cuál de los siguientes métodos no modifica la base de datos de la API?

    A) `PUT`

    B) `POST`

    C) `PATCH`

    D) `GET`

    E) `DELETE`

17. Se dispone un servidor con la siguiente API:

```python
class API:
    def __init__(self):
        self.database = [{"id": "A", "nombre": "Anya"},
                         {"id": "B", "nombre": "Yor"}]

    def metodo_a(self, id):
        personas = list(filter(lambda d: d["id"] == id, self.database))
        persona = personas[0]
        return persona

    def metodo_b(self, id, nombre):
        self.database.append({"id": id, "nombre": nombre})

    def metodo_c(self, id, nombre):
        personas = list(filter(lambda d: d["id"] == id, self.database))
        persona = personas[0]
        persona["nombre"] = nombre
```

    Cuando se hace una solicitud, al servidor, del tipo `POST`, `GET` o `PATCH`, se ejecutará el método correspondiente al tipo de solicitud. Puedes asumir que cada método está asociado a solo un tipo de solicitud distinta y que cumplen con el protocolo HTTP.

    Considerando la base de datos actual (`self.database`) que contiene 2 elementos, ¿cuál afirmación es **incorrecta** sobre este servidor y su API?

    A) Hacer una solicitud del tipo `GET` con el siguiente dato: `id="A"`, hará que la API retorne `{"id": "A", "nombre": "Anya"}`.

    B) Hacer una solicitud del tipo `GET` con el siguiente dato: `id="F"`, hará que la API retorne un código de estado HTTP del grupo `5XX`.

    C) Hacer una solicitud del tipo `POST` con los siguientes datos: `id="H"` y `nombre="Nightfall"` hará que se incluya un nuevo dato a la base de datos.

    D) Hacer una solicitud del tipo `PATCH` con los siguientes datos: `id="A"` y `nombre="Sylvia"` hará que el dato con `id="A"` modifique su nombre por `"Sylvia"`.

    E) Hacer una solicitud del tipo `POST` con los siguientes datos: `id="A"` y `nombre="Loid"` hará que el dato con `id="A"` modifique su nombre por `"Loid"`.

18. ¿Cuál alternativa es **correcta** respecto a los algoritmos BFS y DFS?

    A) BFS es un algoritmo para recorrer únicamente listas ligadas, mientras que DFS es un algoritmo para recorrer cualquier tipo de grafo.

    B) BFS es un algoritmo para recorrer por profundidad, mientras que DFS es para recorrer un grafo por amplitud.

    C) DFS es un algoritmo para recorrer únicamente listas ligadas, mientras que BFS es un algoritmo para recorrer cualquier tipo de grafo.

    D) DFS es un algoritmo para recorrer un grafo por profundidad, mientras que BFS es para recorrer un grafo por amplitud.

    E) Son 2 nombres distintos para el mismo algoritmo.

19. Se dispone de una lista ligada, que fue modelada de la siguiente forma:

```
1  class Nodo:
2      def __init__(self, id, siguiente=None):
3          self.id = id
4          self.siguiente = siguiente
5
6  lista_ligada = Nodo("A", Nodo("B", Nodo("C", Nodo("D"))))
```

    ¿Cuál afirmación es **incorrecta** respecto a este grafo?

    A) Aplicar DFS y BFS entre el nodo `"A"` y el nodo `"D"` entregará el mismo resultado.

    B) Esta lista ligada se puede modelar como un grafo.

    C) La lista ligada está formada por 4 nodos.

    D) Si queremos agregar un nuevo nodo `"X"`, lo podemos hacer solo en la cabeza o solo a la cola de la lista ligada.

    E) El atributo `siguiente` del nodo `"D"` es `None`.

20. Considerando un grafo en donde todo par de nodos posee un camino que los une, y todas las aristas poseen pesos distintos, ¿cuál de las siguientes afirmaciones es **correcta**?

    A) DFS siempre encontrará primero el camino más largo entre dos nodos.

    B) DFS siempre encontrará primero el camino cuya suma de sus pesos es la más pequeña posible.

    C) BFS siempre encontrará primero el camino más largo entre dos nodos.

    D) BFS siempre encontrará primero el camino cuya suma de sus pesos es la más pequeña posible.

    E) Ninguna de las anteriores es correcta.

21. Dado el siguiente grafo representado como una matriz de adyacencia:

```
matriz = [
    [0, 1, 0, 0],
    [1, 0, 0, 0],
    [0, 0, 0, 0],
    [0, 0, 0, 0]
]
```

¿Cuál de las siguientes operaciones se puede lograr sin uso de iteraciones (*for*/*while*) o recursión para ejecutarse?

I. Eliminar un enlace del grafo.

II. Agregar un enlace al grafo.

III. Verificar conexión directa entre 2 nodos.

IV. Encontrar un camino entre dos nodos que no están directamente conectados.

A) Solo IV

B) I y II

C) III y IV

D) I, II y III

E) I, II, III y IV

22. ¿Cuál de las siguientes afirmaciones es **incorrecta** respecto a la separación *frontend-backend*?

A) Esta separación busca una baja cohesión y alto acoplamiento.

B) El *backend* y el *frontend* pueden estar en distintos lenguajes de programación.

C) Según lo implementado en el curso, una responsabilidad de *backend* debería ser verificar que la contraseña ingresada por un usuario sea la correcta.

D) Mostrar los *outputs* al usuario es responsabilidad del *frontend*.

E) Un programa puede funcionar sin la necesidad de recurrir a esta separación.

23. Considerando la siguiente representación de un grafo direccional utilizando una matriz de adyacencia que tiene los nodos 0, 1, 2, 3 y 4 respectivamente:

```
grafo = [
    [0, 1, 0, 1, 0]
    [1, 0, 1, 0, 0]
    [0, 0, 0, 1, 0]
    [0, 1, 0, 0, 0]
    [1, 1, 1, 1, 1]
]
```

Decimos que un nodo X está directamente conectado al nodo Y si en (fila X, columna Y) existe un `1`. Dada esta información, te piden modificar el grafo para cumplir con la siguiente condición: un nodo X quedará directamente conectado a un nodo Y si es que existe un camino de largo 3 o menos que une ambos nodos en el grafo original. A modo de ejemplo, en el siguiente grafo:

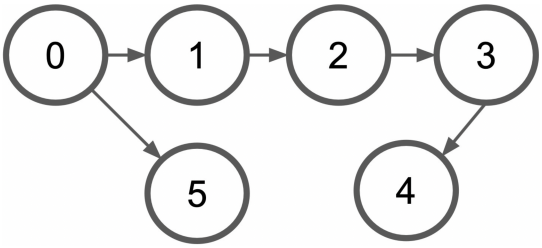

Figura 1: Grafo ejemplo.

Al nodo 0 habría que agregarle aristas que lo conecten directamente con el nodo 2 y 3, al nodo 1 habría que agregarle aristas que lo conecte directamente con el 3 y el 4, y al nodo 2 habría que agregarle una arista que lo conecte directamente con el nodo 4.

Considerando esto, los destinos del nodo 0 en la matriz de adyacencia se verán de la siguiente forma:

A) `[1, 1, 0, 1, 1]`

B) `[1, 1, 1, 1, 0]`

C) `[0, 1, 0, 1, 1]`

D) `[0, 1, 1, 1, 0]`

E) `[1, 1, 1, 1, 1]`

24. ¿Cuál es el *output* del siguiente código?

```
data = [1, 2, 3]
print(reduce(lambda x, y: str(x) + str(y), map(lambda x: x * 2, data)))
```

A) 246

B) 12

C) 123

D) 642

E) 321

25. Respecto a las estructuras de datos *built-in*, ¿cuál o cuáles de las siguientes afirmaciones son **correctas**?

    I. Verificar que un elemento pertenece a un conjunto de datos es más rápido en un `set` que en una lista.

    II. Los diccionarios son estructuras mutables.

    III. Acceder al i-ésimo elemento de una tupla es más rápido que encontrar la llave de un diccionario teniendo únicamente el valor asociado a dicha llave.

    A) Solo I

    B) I y II

    C) I y III

    D) II y III

    E) I, II y III

26. Respecto a Programación Orientada a Objetos (OOP), ¿qué afirmación es **incorrecta**?

    A) Se pueden guardar datos en cada instancia de una clase.

    B) Las clases abstractas permiten forzar la implementación de ciertas funcionalidades a sus subclases.

    C) Mediante herencia se pueden tener clases especializadas y otras más generales.

    D) La herencia permite reutilizar y/o re-implementar ciertas funcionalidades de otra clase.

    E) Las instancias de una o más clases no pueden interactuar entre ellas.

27. Luego de clonar tu repositorio personal y hacer cambios, deseas subir estos a la nube. ¿Cuál sería la secuencia correcta de comandos a utilizar para lograr este objetivo?

    I. `git commit`

    II. `git pull`

    III. `git push`

    IV. `git add`

    A) IV, I y III

    B) III, IV y I

    C) II, IV y II

    D) I, II y III

    E) I, II y III, IV

28. En base a la ejecución del código presentado a continuación, ¿cuántas veces se llama al *setter* de la *property* `valor`?

```python
class CajaSecreta:

    def __init__(self, valor):
        self._valor_secreto = valor

    @property
    def valor(self):
        self.valor = self._valor_secreto + 1
        print(self._valor_secreto)
        return self._valor_secreto

    @valor.setter
    def valor(self, valor):
        print(valor)
        self._valor_secreto += valor

caja = CajaSecreta(5)
caja.valor
caja.valor = caja.valor + 5
```

A) 1 vez

B) 2 veces

C) 3 veces

D) 4 veces

E) 5 veces

29. ¿Cuál o cuáles de los siguientes tipos de datos *built-int* de Python son **inmutables**?

I. `byte`

II. `list`

III. `tuple`

IV. `dict`

V. `bytearray`

A) Solo II

B) Solo III

C) I y III

D) III, IV y V

E) En Python no existen estructuras de datos inmutables.

30. Imagina que tienes un programa que simula una cafetería, el cual cuenta con un *thread* principal y varios *threads* secundarios, para cada mesero, chef y cliente. Además, el café cuenta con un almacén con una cantidad limitada de ingredientes para preparar los platos, y con un único horno que permite preparar un plato a la vez.

    ¿Cuál de las siguientes acciones **no necesita** de *locks* para evitar problemas de concurrencia?

    A) Cambiar el sueldo de un mesero en particular.

    B) Agregar las ventas de un mesero al total de ventas del café.

    C) Asignar la mesa a un cliente recién llegado.

    D) Sacar un ingredientes del almacén.

    E) Utilizar el horno para preparar el plato de un cliente.