Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-1

Examen

9 de Julio 2024

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 de selección múltiple 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.
- Solo podrás retirarte de la sala una vez hayan transcurrido 20 minutos desde que inició de la evaluación.
- Durante la evaluación se realizarán 2 rondas de preguntas. Estas preguntas únicamente serán respondidas por los profesores del ramo.
- En caso de que sea necesario, podrás solicitar hojas blancas para apoyar al desarrollo de la prueba. Para esto levanta la mano y espera que un ayudante se acerque a tu puesto.
- 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*.

Selección múltiple

1. Dado el siguiente código en Python, ¿cuál es el correcto orden de salida resultante de los llamados a `print`? Para esta pregunta, considera que en vez de saltos de línea usaremos una coma por temas de espacio.

```python
import threading

def thread_A(evento, lock):
    evento.wait()
    lock.acquire()
    print("A")

def thread_B(lock):
    lock.acquire()
    print("B")

def thread_C(evento, lock):
    evento.set()
    print("C")
    lock.release()

evento = threading.Event()
lock = threading.Lock()

thread_a = threading.Thread(target=thread_A, args=(evento, lock))
thread_b = threading.Thread(target=thread_B, args=(lock, ))
thread_c = threading.Thread(target=thread_C, args=(evento, lock))

thread_a.start()
thread_b.start()
thread_b.join()
thread_c.start()
```

A) `B, C`

B) `A, B, C`

C) `B, A, C`

D) `B, C, A`

E) No se imprime nada

2. Respecto a *threading*, ¿cuál o cuáles afirmaciones son **correctas**?

   I. Se puede utilizar para modelar sucesos que puedan ocurrir concurrentemente en el tiempo.

   II. Es posible hacer que varios *threads* esperen hasta tener acceso a un mismo recurso protegido.

   III. En una arquitectura cliente-servidor, cada cliente conectado se debe modelar como un *thread*.

   A) Solo I

   B) Solo II

   C) I y II

   D) I y III

   E) II y III

3. Según las siguientes definiciones asociadas a los principios de diseño de *software* de calidad:

   Definición A Cada uno de los componentes de *software* debe realizar solo las tareas para las que fue creado.

   Definición B La modificación de un componente, implica que es necesario modificar otro componente para que la implementación del cambio sea correcta y completa.

   Indique qué concepto corresponde cada definición y qué debemos buscar al momento de diseñar un *software* de calidad:

A)	Def A: Cohesión	Def B: Acoplamiento	Baja cohesión y alto acoplamiento.
B)	Def A: Cohesión	Def B: Acoplamiento	Alta cohesión y bajo acoplamiento.
C)	Def A: Acoplamiento	Def B: Cohesión	Bajo acoplamiento y alta cohesión.
D)	Def A: Acoplamiento	Def B: Cohesión	Alto acoplamiento y baja cohesión.
E)	Def A: Acoplamiento	Def B: Cohesión	Alto acoplamiento y alta cohesión.

4. Según los contenidos del curso, ¿cuál de la siguiente información **no** se incluye en un diagrama de clases?

   A) Los atributos y *properties* de cada clase.

   B) Las líneas de código que ejecutan cada método.

   C) Si una clase se relaciona con otra mediante herencia o contención.

   D) Las señales de `PyQt` definidas en cada clase.

   E) Los métodos de cada clase.

5. En el contexto de la correcta implementación de una Interfaz Gráfica y respetando las convenciones del curso expuestas en los contenidos, ¿cuál o cuáles de las siguientes funcionalidades deben ir en el *back-end* del programa?

   I. Enviar y recibir mensajes del servidor.

   II. Mostrar imágenes del programa.

   III. Recibir las acciones del usuario.

   IV. Leer y actualizar el contenido de un archivo.

   A) Solo I

   B) Solo III

   C) II y III

   D) I y IV

   E) I, II y IV

6. ¿En cuál caso se debe ocupar exclusivamente `QTimer` en vez de `QThread`?

   A) Para desplazar un elemento en la pantalla.

   B) Para detectar en todo momento si una tecla es presionada.

   C) Para detectar en todo momento la posición del cursor/*mouse*.

   D) Para cronometrar el tiempo de un juego.

   E) Ninguna de las anteriores.

7. Según las aplicaciones realizadas en el curso, seleccione la alternativa que relaciona **correctamente** las clases de `Threading` con su **análogo** de `PyQt`.

   A) `Lock`: `QMutex` `Thread`: `QTimer` `Timer`: `QThread`

   B) `Lock`: `QMutex` `Event`: `pyqtSignal` `Timer`: `QTimer`

   C) `Lock`: `QMutex` `Thread`: `QThread` `Timer`: `QTimer`

   D) `Event`: `QMutex` `Thread`: `QThread` `Timer`: `QTimer`

   E) `Event`: `pyqtSignal` `Thread`: `QThread` `Timer`: `QThread`

8. En el contexto de Programación Funcional, ¿qué es lo **primordial** que debe cumplir una función para que sea considerada una **función generadora**?

   A) Utilizar el comando **`yield`**.

   B) Utilizar estructuras por comprensión.

   C) No utilizar el comando **`return`**.

   D) No utilizar los comandos **`for`** y **`while`**.

   E) Retornar el resultado tras ejecutar las funciones `map`, `filter` y/o `reduce`.

9. Sea un código que contiene las clases `Iterador` e `Iterable`, las cuales fueron programadas correctamente y son coherentes con sus nombres.

   Indique los métodos y el orden en que se ejecutan en el siguiente código:

```
for elemento in Iterador:
    print(elemento)
```

   I. `Iterador.__iter__()`

   II. `Iterador.__next__()`

   III. `Iterable.__iter__()`

   IV. `Iterable.__next__()`

   A) I y II

   B) III y IV

   C) I, III y IV

   D) III, I y II

   E) I, II, III y IV

10. ¿Cuáles de estas afirmaciones respecto a `map` son **correctas**?

   I. Necesita recibir una función que retorna un booleano (`True`/`False`).

   II. Aplica una función acumulativamente hasta quedarse con un solo valor.

   III. Retorna un iterador.

   A) Solo I

   B) Solo II

   C) Solo III

   D) I y II

   E) II y III

11. Se te entrega el siguiente código donde se definen las clases `Nodo` y `X`:

```
class Nodo:
    def __init__(self, id, siguiente=None):
        self.id = id
        self.siguiente = siguiente

class X:
    def __init__(self):
        self.cabeza = None

    def add(self, nodo_id):
        cabeza = self.cabeza
        nodo = Nodo(nodo_id, cabeza)
        self.cabeza = nodo

    def extract(self):
        nodo = self.cabeza
        self.cabeza = nodo.siguiente
        return nodo.id
```

A partir del código anterior, identifica qué estructura busca simular la clase `X`:

A) Lista (de Python)

B) Stack

C) Cola

D) Deque

E) Ninguna de las anteriores

12. Indique cuál de las siguientes afirmaciones respecto a la serialización mediante `pickle` es **incorrecta**:

A) El resultado de la serialización puede escribirse en un archivo.

B) Los objetos serializados pueden ser fácilmente leídos y entendidos por humanos.

C) Permite serializar más tipos de objetos que JSON.

D) Los objetos serializados pueden ser enviados a través de sockets.

E) Es posible personalizar el resultado de la serialización.

13. Considere el siguiente código en Python:

```
from utils import funcion

def caso_general():
    try:
        funcion()
        print("LLAMADO")
    except:
        print("ERROR")

def caso_exception():
    try:
        funcion()
        print("LLAMADO")
    except Exception:
        print("ERROR")

def caso_value():
    try:
        funcion()
        print("LLAMADO")
    except ValueError:
        print("ERROR")
```

Suponga que `funcion()` levanta un **`ValueError`**. ¿Qué funciones imprimirán lo mismo?

A) Solo `caso_general()` y `caso_exception()`.

B) Solo `caso_general()` y `caso_value()`.

C) Solo `caso_exception()` y `caso_value()`.

D) Todas imprimirán lo mismo.

E) Todas imprimirán cosas diferentes.

14. Respecto a la codificación en *bytes*, es **correcto** afirmar que:

A) Solo puede realizarse utilizando `JSON` y `pickle`.

B) Al codificar un *string* de largo n, siempre se obtendrán n *bytes*.

C) No usar el mismo *encoding* para codificar y decodificar siempre levanta una excepción.

D) Un mismo *byte* puede codificar diferentes caracteres.

E) Existen *bytes* que no tienen una representación numérica.

15. Considere el siguiente código en Python:

```
try:
    resultado = 10 / 0
    print("TRY")
except ZeroDivisionError:
    print("EXCEPT")
else:
    print("ELSE")
finally:
    print("FINALLY")
```

Sin considerar saltos de línea, ¿qué imprimirá el código anterior?

A) `EXCEPT FINALLY`

B) `ELSE FINALLY`

C) `EXCEPT FINALLY TRY`

D) `TRY EXCEPT FINALLY`

E) `TRY ELSE FINALLY`

16. ¿Cuál de las siguientes afirmaciones es **correcta** sobre los puertos de un computador?

A) Es posible usar cualquier puerto que queramos al momento de levantar un servidor.

B) Es posible tener un servidor que utilice varios puertos a la vez.

C) Siempre debemos usar un puerto distinto para cada cliente que se conecta a nuestro servidor.

D) Los computadores poseen una cantidad infinita de puertos disponibles.

E) Para levantar dos servidores en el mismo computador, debo usar el mismo puerto para ambos.

17. ¿Cuál de las siguientes es **correcta** respecto a la modelación de Servidores y Clientes usando *sockets* en Python?

A) Es posible tener un programa que sea Servidor y Cliente al mismo tiempo.

B) Un Cliente puede estar conectado máximo a un Servidor al mismo tiempo.

C) Mientras esté activo, un Servidor siempre aceptará a todos los Clientes que se conecten.

D) Una conexión entre Servidor y Cliente solo puede ser cerrada cuando uno de los dos programas finaliza su ejecución.

E) En caso de fallo en el Cliente, el Servidor automáticamente cierra la conexión a dicho Cliente.

18. Suponga que se le pide programar un juego de UNO en Python, haciendo uso de *networking* para lograr que este sea multijugador.

    Siguiendo una correcta modelación de la arquitectura Cliente - Servidor, ¿cuál o cuáles de las siguientes tareas deberían ser responsabilidad **exclusiva** del servidor?

    I. Almacenar a todos los nombres de usuarios y datos de *log-in*.

    II. Administrar las diferentes salas de juego disponibles y su capacidad.

    III. Detectar la carta seleccionada por el usuario en función de la posición del *mouse* al momento de hacer *click*.

    IV. Decidir a qué usuario le corresponde el siguiente turno.

    A) I y II

    B) III y IV

    C) I, II y IV

    D) I, III y IV

    E) I, II, III y IV

19. Suponga que se tiene un servidor remoto cuyo código no conoce. Se le encarga construir un programa de Python cuyo objetivo sea comunicarse con dicho servidor mediante el protocolo TCP/IP.

    Indique cuál o cuáles de las siguientes son las características **mínimas** a conocer **del servidor** para que su programa sea capaz de enviar y/o recibir mensajes correctamente con el servidor:

    I. Lenguaje de programación en que está hecho el servidor.

    II. Dirección IP y puerto que está usando el servidor.

    III. Método de codificación y decodificación que usa el servidor.

    IV. Cuántos clientes ya están conectados al servidor.

    A) I y II

    B) II y III

    C) I, II y III

    D) II, III y IV

    E) I, II, III y IV

20. Se tiene un programa que simula un cliente que recibe *strings* codificados con `utf-8` desde un servidor, haciendo uso de *sockets* mediante el siguiente código:

```
1  import socket
2
3  sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
4
5  host = "localhost"
6  port = "6000"
7
8  sock.connect((host, port))
9  data1 = sock.recv(100)
10 data2 = sock.recv(100)
11
12 data = data1 + data2
13 print(data.decode("utf-8"))
```

¿Cuál de las siguientes afirmaciones es **correcta** respecto a posibles problemas que podrían encontrarse al ejecutar este código?

A) Si el servidor solo envía un mensaje de 80 *bytes*, el programa se quedará estancado en la línea 9 hasta que el servidor envíe suficientes mensajes para completar al menos 100 *bytes*.

B) Si el servidor envía un solo mensaje de 120 *bytes*, el programa lanzará error en la línea 9 debido a que el mensaje sobrepasa el tamaño del `recv`.

C) Si el servidor envía un mensaje de 120 *bytes*, y luego uno de 80 bytes, el cliente no podrá decodificar correctamente la línea 13.

D) Si el servidor envía un mensaje de 120 *bytes* y luego uno de 100 *bytes*, el cliente no obtendrá 20 *bytes* del primer mensaje.

E) Si el servidor envía un mensaje de 120 *bytes*, y luego uno de 100 *bytes*, el cliente no obtendrá 20 *bytes* del segundo mensaje.

21. Se te entrega un conjunto de grupos de estado HTTP y un conjunto de definiciones:

Grupo de estados HTTP	Definiciones
I. `2XX`	a. Error del cliente
II. `4XX`	b. Error del servidor
III. `5XX`	c. Solicitud exitosa

Relaciona cada grupo de estado HTTP con la definición que le corresponde:

A) I a, II b y III c

B) I b, II a y III c

C) I b, II c y III a

D) I c, II a y III b

E) I c, II b y III a

22. Se dispone de un servidor con una API de 3 *endpoints*. Cada *endpoint* fue programado correctamente, pero no fue conectado a la solicitud correspondiente.

    El código de la API es el siguiente:

```python
from flask import Flask, request

app = Flask(__name__)
DATABASE = [{"nombre": "One Piece", "caps": "?"},
            {"nombre": "Gintama", "caps": 215}]

@app.route("/anime/<int:index>", methods=["XXX"]) # ¡No olvidar conectar!
def funcion_a(index: int):
    body_data = request.get_json(force=True)
    caps = body_data["caps"]
    DATABASE[index]["caps"] = caps
    return DATABASE[index]

@app.route("/anime", methods=["XXX"]) # ¡No olvidar conectar!
def funcion_b():
    body_data = request.get_json(force=True)
    caps = body_data["caps"]
    nombre = body_data["nombre"]
    DATABASE.append({"nombre": nombre, "caps": caps})
    return DATABASE[-1]

@app.route("/anime/nombre/<str:nombre>", methods=["XXX"]) # ¡No olvidar conectar!
def funcion_c(nombre: str):
    return list(filter(lambda d: d["nombre"] == nombre, DATABASE))

if __name__ == "__main__":
    app.run(host="localhost", port=4444)
```

    Indique la alternativa que, al reemplazar los `"XXX"`, conecta **correctamente** y respetando el protocolo HTTP, cada uno de los *endpoint* con la solicitud correspondiente.

    A) GET: `funcion_b` POST: `funcion_a` PATCH: `funcion_c`

    B) GET: `funcion_c` POST: `funcion_a` PATCH: `funcion_b`

    C) GET: `funcion_c` POST: `funcion_b` PATCH: `funcion_a`

    D) GET: `funcion_b` POST: `funcion_c` PATCH: `funcion_a`

    E) Ninguna alternativa asocia correctamente cada función con la solicitud correspondiente.

23. Desde el cliente, se necesita consultar una API, que cumple el protocolo HTTP, para actualizar **únicamente** el número de teléfono del usuario. Con este objetivo, el cliente accede al *endpoint* `"/{nombre_usuario}/datos-personales"`. ¿Cuál de los siguientes métodos deberá estar diseñado desde el servidor de dicha API para realizar esa acción?

    A) `POST`

    B) `PUT`

    C) `DELETE`

    D) `PATCH`

    E) `GET`

24. Respecto a las expresiones regulares, ¿cuál de las siguientes afirmaciones es **correcta**?

    A) Son secuencias especiales de caracteres que permiten encontrar únicamente *strings* que presenten formatos particulares como las fechas o el RUT chileno.

    B) Cualquier cadena de texto es considerada una expresión regular válida.

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

    D) Solamente pueden capturar patrones que no contengan símbolos numéricos.

    E) Todas las afirmaciones anteriores son incorrectas.

25. Según los contenidos del curso, ¿en cuál de los siguientes casos de uso es **más apropiado** utilizar principalmente expresiones regulares?

    A) Extraer el puntaje numérico más alto de una tabla de ganadores.

    B) Encontrar y eliminar todos los dígitos de una cadena de texto alfanumérica.

    C) Para buscar un valor en un diccionario con tipos de datos heterogéneos.

    D) Contar la cantidad de faltas de ortografía.

    E) Recorrer los elementos de cualquier objeto iterable.

26. Sea una lista de Python, `lista = ['a', 'b', 'c', 'd']`. Indique cuál de las siguientes expresiones permite imprimir la lista de forma que en la terminal aparezca exactamente: `a b c d`

    A) `print(lista)`

    B) `print(*lista)`

    C) `print(**lista)`

    D) `print(str(lista))`

    E) `print(tuple(lista))`

27. Respecto a las estructuras de datos vistas en el curso, ¿cuál de las siguientes estructuras **no** permite alterar el largo de esta una vez instanciada, y solo permite contener elementos inmutables?

    A) Tupla

    B) *Set*

    C) Generador

    D) Diccionario

    E) Ninguna de las anteriores.

28. En función del siguiente código, ¿cuál o cuáles clases se pueden instanciar sin provocar un error al momento de ejecutar el código?

```
1  from abc import ABC, abstractmethod
2
3  class Animal(ABC):
4      @abstractmethod
5      def saludar(self):
6          pass
7
8      @abstractmethod
9      def despedir(self):
10         pass
11
12 class Perro(Animal):
13     def saludar(self):
14         print("Wenomechainsama")
15
16 class RusselTerrier(Perro):
17     def despedir(self):
18         print("Tumajarbisaun")
```

    A) Solo la clase `Animal`.

    B) Solo la clase `Perro`.

    C) Solo la clase `RusselTerrier`.

    D) Las clases `Perro` y `RusselTerrier`.

    E) Ninguna se puede instanciar.

29. ¿Cuál de las siguientes reglas **no pertenece** a las reglas de estilo (PEP8) solicitadas en las tareas del curso?

    A) Espacio después de la coma.

    B) No exceder un máximo de caracteres por línea.

    C) El nombre de los archivos deben estar en ingles.

    D) Uso adecuado de `snake_case` y `CamelCase`.

    E) Uso de variables aclarativas.

30. Dentro del contexto del curso, ¿cuál de las siguientes alternativas relacionadas a un entorno de trabajo de desarrollo puede variar dependiendo en qué sistema operativo se trabaje?

    A) Comandos de Git disponibles.

    B) Separadores de *paths*.

    C) Reglas de PEP8.

    D) Extensión del archivo `.gitignore`.

    E) Forma de importar módulos y librerías de Python.

Preguntas 31 (Bonus)

Debes responder solo una de las siguientes preguntas. Esta corresponderá a la pregunta pregunta 31 del examen que otorga 1 décima a la nota final del examen en caso de tenerla correcta.

Cada pregunta, corresponde a una de las 5 secciones del ramo y todas presentan la misma alternativa correcta, por lo cual puedes escoger la que estimes conveniente.

No es obligación escoger la pregunta de tu sección.

S1. Con respecto a la Sección 1, complete la oración con la alternativa **correcta**:

Al profesor Hernán le gusta incluir referencias sobre _____ *en clases.*

A) Sus Crocs

B) El sueldo de "pobresor"

C) Sus disfraces inflables

D) Japón y/o anime

E) Perry el Ornitorrinco

S2. Con respecto a la Sección 2, ¿qué entrega la profesora Dani como premio a los estudiantes que participan en clase?

A) PepaPuntos

B) Patos de hule

C) Décimas

D) Dulces

E) Fotos de sus mascotas (Pepa y Luna)

S3. Con respecto a la Sección 3, ¿qué personaje aparece siempre en las diapositivas de la profesora ~~Fran~~?

A) Frutillita

B) Bulbasaur

C) Snoopy

D) Hello Kitty

E) El Nóctulo

S4. Con respecto a la Sección 4, ¿cuál de estos temas **NO** es mencionado recurrentemente por el profesor Dante?

A) La antigüedad de los memes de las clases.

B) Las diferencias entre Python y otros lenguajes de programación.

C) La edad del profesor.

D) Las ventajas de Google Slides.

E) La teoría detrás de los contenidos.

S5. Con respecto a la Sección 5, ¿qué objeto/instrumento toca la profesora Paqui cuando se pone triste en clases?

A) Batería Chiquita

B) Trombón Chiquito

C) Oboe Chiquito

D) Violín Chiquito

E) Trompeta Chiquita