Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2026-1

Interrogación 2

28 de Mayo 2026

Selección múltiple [20 % nota final]

Cada alternativa vale lo mismo.

1. B
2. D
3. A
4. C
5. A

Desarrollo

Pregunta 1. Interfaces gráficas [20 % nota final]

Distribución de puntaje (máx. 6 puntos)

Parte A.

- **(0.5 puntos)** Se emite una señal desde VentanaSemanal a Controlador que indica el día seleccionado. Puede ser un string o un int indicando el día.

- **(0.5 puntos)** El controlador almacena internamente el día seleccionado.

- **(0.5 puntos)** Luego el Controlador emite una señal a VentanaDiaria que contiene la información de los eventos del día seleccionado. Esta información debe incluir descripciones y horas.

- **(0.5 puntos)** VentanaDiaria se actualiza con la información recibida.

Solución propuesta: Se emite una señal desde VentanaSemanal a Controlador que contiene un string con el día de semana ("Lunes", "Martes"...). El controlador guarda el día recibido en una variable, obtiene los eventos del día correspondiente y envía otra señal a VentanaDiaria con una lista de eventos que incluye la hora y descripción de cada uno. Finalmente la VentanaDiaria muestra los eventos del día seleccionado.

Parte B.

- **(0.5 puntos)** Se emite una señal desde VentanaFormulario a Controlador que indica toda la información rellenada en el formulario (Día, hora y descripción).

- **(0.5 puntos)** Controlador actualiza la información interna de los eventos.

- **(0.5 puntos)** Luego el Controlador emite una señal a VentanaSemanal indicando el día del nuevo evento (string o int). La señal también puede incluir opcionalmente el nuevo número de eventos de ese día. Ambos son correctos.

- **(0.5 puntos)** VentanaSemanal actualiza el número en el día indicado.

- **(0.5 puntos)** En el caso de que el día del nuevo evento coincida con el día seleccionado, se debe enviar una señal a VentanaDiaria que contenga la información del evento (descripción y hora)

- **(0.5 puntos)** VentanaDiaria actualiza su información agregando el nuevo evento.

Solución propuesta: VentanaFormulario emite una señal a Controlador que contiene un str para el día, un str con la hora, y un str con la descripción. El controlador guarda la información recibida por la señal y ve si el día seleccionado corresponde al día del evento creado. Luego emite una señal a VentanaSemanal con un str indicando el día al que se le agregó un evento. VentanaSemanal actualiza la interfaz sumando 1 al contador del día correspondiente.

Finalmente, si el día del evento nuevo corresponde al día seleccionado, el controlador emite una señal a VentanaDiaria con los datos del evento nuevo (str, str, str)

VentanaDiaria se actualiza agregando el nuevo evento.

Parte C.

- **(0.5 puntos)** Controlador debe guardar la información de todos los eventos. Incluido el día, hora y descripción.

- **(0.5 puntos)** Además debe tener una variable que indica el día seleccionado actualmente.

Solución propuesta: El controlador almacena la información de cada evento: Día, Fecha y Descripción, además tiene una variable que indica el día seleccionado.

Pregunta 2. Excepciones [20 % nota final]

Parte A — Análisis de casos (3 puntos)

Importante: En los casos de `try/except`, no basta con mencionar que "puede ocurrir un error" o "una excepción". Se espera justificar por qué corresponde utilizar `try/except`.

- **Caso 1**
  - **(0.5 puntos)** Escoge correctamente `if/else`.
  - **(0.5 puntos)** Justifica que las condiciones pueden validarse previamente.

  Ideas esperadas en la justificación:

  - "Se puede verificar el largo antes."
  - "Se puede revisar si contiene letras y números."
  - "Las condiciones son conocidas previamente."

  Respuesta esperada (ejemplo):

  > Corresponde utilizar `if/else` porque las condiciones que debe cumplir la contraseña (largo mínimo de 8 caracteres y ser alfanumérica) pueden verificarse **antes** de cualquier operación, directamente sobre el string ingresado. No se trata de un error que ocurra durante la ejecución, sino de validaciones conocidas y comprobables previamente.

- **Caso 2**
  - **(0.5 puntos)** Escoge correctamente `try/except`.
  - **(0.5 puntos)** Justifica que no se puede conocer el contenido del generador previamente y que la situación ocurre durante la ejecución de `sum()`.

  Ideas esperadas en la justificación:

  - "No se puede saber de antemano qué tipos entregará el generador."
  - "Validar previamente obligaría a consumir el generador."
  - "El problema ocurre **durante** la ejecución de `sum()`, al intentar sumar un valor no numérico."
  - "El tipo de cada elemento se conoce sólo al momento de ser generado."

  Respuesta esperada (ejemplo):

  > Corresponde utilizar `try/except` porque no existe forma de validar previamente los elementos del generador: hacerlo implicaría consumirlo y dejarlo inutilizable para `sum()`. El posible error (sumar un valor no numérico) ocurre **durante** la ejecución de `sum()`, por lo que conviene capturarlo cuando efectivamente se produzca.

- **Caso 3**
  - **(0.5 puntos)** Escoge correctamente `try/except`.
  - **(0.5 puntos)** Justifica que pueden ocurrir problemas durante el procesamiento del archivo.

  Ideas esperadas en la justificación:

  - "Pueden ocurrir errores al leer o procesar el archivo."
  - "La situación ocurre durante la operación."

- "No se puede garantizar el estado del archivo (existencia, permisos, formato) antes de intentar procesarlo."

Respuesta esperada (ejemplo):

Corresponde utilizar `try/except` porque los problemas (archivo inexistente, sin permisos, contenido mal formado, etc.) ocurren **durante** la lectura o procesamiento del archivo. Estas condiciones dependen del entorno y no se pueden verificar de forma confiable antes de intentar la operación.

Parte B — Análisis de código (3 puntos)

Output esperado:

```
Precio final: 80
Cierre de operación
Reporte generado
Cierre de operación
ValueError: No puede ser negativo
Cierre de operación
TypeError: Debe ser un número
```

Distribución de puntaje (máx. 3 puntos):

- **Si identifica correctamente todas las líneas del output: 3 puntos.**
- **Si tiene errores en 2 o 3 líneas: 1.5 puntos.**
- **Si tiene errores en más de 3 líneas: 0 puntos.**

Pregunta 3. Iterable e Iterador [20 % nota final]

Distribución de puntaje (máx. 7 puntos)

- **(2 puntos)** En cada llamado de `IteradorManoCartas_._next__()`, se busca correctamente una carta que pueda ser descartada.

- **(2 puntos)** Al descartar una carta, se actualiza correctamente la Lista Ligada de `IteradorManoCartas` (`self.mano`).

  Si no se maneja correctamente el caso de que la carta a descartar sea la cabeza de la Lista Ligada, se opta a mitad de puntaje.

- **(2 punto)** Se mantiene un estado de la última carta jugada en un atributo de `IteradorManoCartas`. Este estado se actualiza en cada llamado del `IteradorManoCartas.__next__()`.

  El valor inicial de este atributo queda a criterio del alumno. Algunas alternativas son: alguna carta arbitraría de la Lista Ligada (p.ej. la primera carta de la mano), algún valor arbitrario (p.ej. `None`).

- **(0.5 punto)** Si se acaban las cartas de la mano, se levanta un `StopIteration`.

- **(0.5 punto)** Si no se puede jugar ninguna de las cartas de la mano, se levanta un `StopIteration`.

Para cada ítem: se entrega puntaje total si se cumple el ítem al 100 %, mitad de puntaje si presenta errores menores, puntaje si se logra el menos del 50 % esperado para el ítem.

Se evaluará la pregunta con nota mínima, si no se cumplen las indicaciones del enunciado:

- Utilizar únicamente la Lista Ligada como estructuras de datos, es decir, no haber utilizado otras estructuras de datos (p.ej. listas, tuplas, diccionarios, etc).
- No modificar el funcionamiento de los métodos y atributos definidos inicialmente en las clases `Carta`, `ManoCarta`, `IteradorManoCarta`.

Posible solución

```python
from copy import deepcopy


class Carta:
    def __init__(self, color: str, numero: int) -> None:
        self.color = color
        self.numero = numero
        self.siguiente = None

    def __eq__(self, value: Carta) -> bool:
        return self.color == value.color or self.numero == value.numero

```

```python
class ManoCartas:
    def __init__(self) -> None:
        self.cabeza = None
        self.cola = None

    def agregar_carta(self, carta: Carta) -> None:
        if self.cabeza is None:
            self.cabeza = carta
            self.cola = carta
            return
        self.cola.siguiente = carta
        self.cola = carta

    def __iter__(self) -> IteradorManoCartas:
        return IteradorManoCartas(deepcopy(self.cabeza))

class IteradorManoCartas:
    def __init__(self, cabeza: Carta) -> None:
        self.mano = cabeza
        self.pozo = cabeza

    def __next__(self) -> Carta:
        carta_anterior = None
        carta_actual = self.mano

        # Si no quedan cartas se termina la iteración
        if carta_actual is None:
            raise StopIteration('No quedan cartas en el mazo')

        # Mientras existan cartas por ser revisadas
        while carta_actual is not None:
            # Comprobamos si la carta puede ser jugada (mismo color o número)
            if carta_actual == self.pozo:
                # Si la carta es la cabeza, se cambia la referencia
                if carta_actual is self.mano:
                    self.mano = carta_actual.siguiente
                # Sacamos la carta de la ListaLigada
                if carta_anterior is not None:
                    carta_anterior.siguiente = carta_actual.siguiente
                # Actualizamos el pozo
                self.pozo = carta_actual
                return carta_actual

            carta_anterior = carta_actual
            carta_actual = carta_actual.siguiente

        # Si no se puede jugar una carta, se termina la iteración
        raise StopIteration('No se puede jugar ninguna carta')
```

Pregunta 4. Implementación y ejecución de código [20 % nota final]

Parte A. Tabla de valores [2 puntos]

Distribución puntaje (máx. 2 puntos)

- **(1 punto)** `elemento` posee los valores correctos
- **(1 punto)** `paso_actual` posee los valores correctos

Solución

Valor entregado	None	"salchicha"	"pan"	"arroz"	"salchicha"	"DCComponenteSecreto"	"pan"
paso_actual	0	0	1	1	2	0	1
elemento	None	"salchicha"	"pan"	"arroz"	"salchicha"	"DCComponenteSecreto"	"pan"

Nota: Puede haber una columna antes del primer `None`, que represente lo que se obtiene desde que se instancia el generador hasta el primer `yield`. El puntaje no difiere con su ausencia o presencia.

Parte B. Contar DCCompletos [2 puntos]

Distribución puntaje (máx. 2 puntos)

- **(0.5 puntos)** Instancia correctamente el generador antes de usarlo.
- **(0.5 puntos)** Usa primero `next` o `send` con valor `None`.
- **(0.5 puntos)** Envía correctamente los valores de `gen_ingredientes_dccompleto` usando `.send`.
- **(0.5 puntos)** La cuenta final de dccompletos es correcta.

Posible solución

```
mi_generador = cadena_generadora()
next(mi_generador)
dccompletos = 0
for elemento in gen_ingredientes_dccompleto:
    dccompletos += mi_generador.send(elemento)
print(dccompletos)

```

Parte C. Contar ingredientes perdidos [2 puntos]

Distribución de puntaje (máx. 2 puntos)

- **(0.5 puntos)** Consume correctamente `resultados_cadena` considerando que es un generador.
- **(1 puntos)** El código es correcto para cuando el generador posee más de un elemento.
- **(0.5 puntos)** El código es correcto para cuando el generador posee menos de dos elementos.

Nota: Si se responde usando algún elemento prohibido, se entregan 0 puntos, Los elementos prohibidos son:

- loops (`for`, `while`, etc.).
- estructuras de datos (`list`, `tuple`, etc.).
- `class`.

Posibles soluciones

`reduce`:

```
reduce(lambda acc, x: acc - 2 if x == 1 else acc + 1, resultados_cadena, 0)
```

`map + reduce + lambda`

```
reduce(lambda acc, x: acc - x,
    map(
        lambda x: 2 if x == 1 else -1, resultados_cadena
    ), 0)
```

`map + reduce`

```
def auxiliar(valor):
    if valor == 1:
        return 2
    else:
        return -1

reduce(lambda acc, x: acc - x, map(auxiliar, resultados_cadena), 0)
```

`sum + map`:

```
sum(map(lambda x: -2 if x == 1 else 1, resultados_cadena))
```