Nombre: Catalina González

Certamen 2 ELO-329 Diseño y Programación Orientada a Objetos (1s2023)

En este certamen usted no podrá hacer preguntas. Si algo no está claro, indíquelo en su respuesta, haga una suposición razonable y resuelva conforme a ella.

Primera parte, **sin apuntes** (40 minutos):
Primera pregunta (30 pts.)

A. Complete en el siguiente diagrama los nombres de al menos 5 de los elementos que componen el stack Android.

Arquitectura en capas de Android:

B. En el código proporcionado, se han declarado varias funciones miembro dentro de la clase MiClase con diferentes modificadores const.
Indique cuáles de estas funciones miembro generarán errores de compilación y cuáles no. Para aquellas que generen errores, justifique su respuesta.

```
class MiClase{
   public:
      void add1(int *pinput, int x);
      void add2(int *pinput, int x) const;
      void add3(int *const pinput, int x);
      void add4(const int *pinput, int x);
      void add5(const int *const pinput, int x) const;
   private:
      int *px = new int(5);
};
```

toma un puntero a un entero y un entero como param. para operar
la función no modificará los miembros de la clase
no se puede modificar para apuntar a otra dir. de memoria
el valor al que apunta no se puede modificar
el puntero y su valor son ctes. la fx no modifica los miembros de la clase

```
void MiClase::add1(int *pinput, int x){
   this->px += *pinput + x;
   std::cout << *px << std::endl;
}
//R: no genera errores
```

```
~~void MiClase::add3(int *const pinput, int x){~~
   ~~this->px += *pinput + x;~~
   ~~std::cout << *px << std::endl;~~
~~}~~
~~//R:~~
```

ELO329 1s2023, 2do certamen. 01.07.23

Nombre: Catalina González

```
void MiClase::add2(int *pinput, int x) const{
    this->px += *pinput + x;
    std::cout << *px << std::endl;
}
```
//R: Error de compilación la función no modifica los miembros de la clase, se intenta modificar *px y px es de la clase

```
void MiClase::add3(int *const pinput, int x){
    this->px += *pinput + x;
    std::cout << *px << std::endl;
}
```
//R: No genera errores porque se apunta correctamente

```
void MiClase::add4(const int *pinput, int x){
    this->px += *pinput + x;
    std::cout << *px << std::endl;
}
```
//R: No genera errores porque no se modifica el valor a que apunta *pinput

```
void MiClase::add5(const int *const pinput, int x) const{
    this->px += *pinput + x;
    std::cout << *px << std::endl;
}
```
//R: Error de compilación porque la función no modifica los miembros de la clase

C. Sabiendo que el compilador ubica la variable **i** en la dirección 2345 y **j** en la dirección 2357 ¿qué debería imprimir el siguiente código?

```
int i[3] = {5, 7, 11};
int * j = i;
int * &k = j;
cout << "i =" << i << endl;          // →
cout << "&j =" << &j << endl;        // →
cout << "k =" << k << endl;          // →
cout << "*k =" << *k << endl;        // →
cout << "*(j+2)=" << *(j+2) << endl; // →
```

Aquí ponga sus respuestas
i = 2345
&j = 2357
K = 2345
*K = 5
*(j+2) = 11

→ dir. de memoria del 1° elem. del array

→ &j = dirección de la variable j.

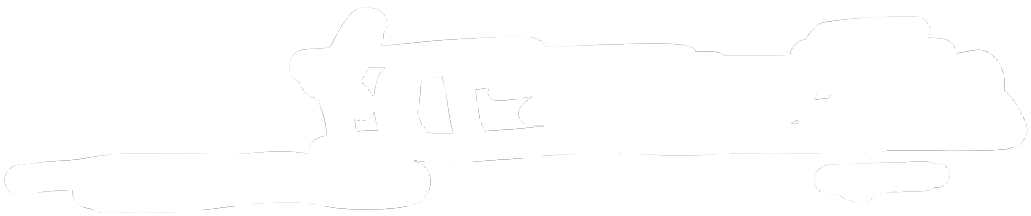
→ K referencia a j , j apunta a i

→ *K desreferencia a K, accedemos a K K apunta al 1° elem de i

↳ *(j+2) accede al 3er elem de i (i[2] = 11)

ELO329 1s2023, 2do certamen. 01.07.23

Nombre: Catalina González



D. Considerando las declaraciones de clases dadas, en columna “Con virtual” indique qué aparece por consola al ejecutar el código de la derecha.

Si eliminamos la palabra virtual en el método foo() de cada clase, en columna “Sin virtual” indique qué aparece por consola al ejecutar el código de la derecha.

polimorfismo dinámico / sin polimorfismo dinámico

```
class A {
   public:
      virtual void foo() { cout << “A” << endl; }
};
class B: public A {
   public:
      virtual void foo() { cout << “B” << endl; }
};
class C: public B {
   public:
      virtual void foo() { cout << “C” << endl; }
};
```

```
B b;
A *a=&b;
a ->foo();
C c;
A aa=c;
aa.foo();
a=&c;
B *bb = (B*) a;
bb->foo();
```

a apunta a A, pero apunta a una instancia de B // →

aa es una inst. de A, pero se inicializó con c

no es puntero, no hay polim. //→

puntero a B

// →

bb apunta a B se inicializa con a que apunta a A realmente apunta a B

Con virtual	Sin virtual
B	A
A	A
B	B

E. En el siguiente ejemplo, se tienen las tres funciones: ‘funcionA’, ‘funcionB’ y la función principal ‘main’. Al ejecutar la función main mostrada, índique qué mensaje se muestra al ingresar por consola el valor negativo **-2** y qué se muestra al ingresar el valor positivo **3**.

```
void funcionA(int valor) {
   if (valor < 0)
        throw "El valor no puede ser negativo";
   cout << "El valor es: " << valor << endl;
}
void funcionB(int valor) {
    try {
         funcionA(valor);
    } catch (const char* mensaje) {
        cout << "Se produjo una excepción: ";
        cout << mensaje << endl;
    }
}
```

```
int main()  {
    int valor;
    cin >> valor;
     try {
        funcionB(valor);
     } catch (...) {
        cout << "Se produjo una excepción";
        cout<< "desconocida" << endl;
    }
}
```

Si se ingresa el valor -2

- main llama a funcionB(-2)
  - ↳ llama a funcionA(-2)
    - ↳ lanza excepción: "El valor no puede ser negativo"
      - ↳ capturada por catch en funcionB

"Se produjo una excepción:
El valor no puede ser negativo"

Nombre: Catalina González

Si se ingresa el valor 3
- main llama a funcionB(3)
  ↳ llama a funcionA(3)
    ↳ no lanza excepción
      ↳ imprime "El valor es: 3"
  ↳ funcionB(3) no imprime nada porque no hay excepción

F. Mencione 1 semejanza y 1 diferencia entre casos de uso e historias de usuario.

Semejanza:
Capturan requisitos del sistema desde la perspectiva del usuario final

Diferencia:
C.U = descripciones detalladas de cómo interactúan los usuarios con un sistema
H.U = descripciones breves de una funcionalidad del punto de vista del usuario