Escuela de Ciencias Aplicadas e Ingeniería
Ingeniería de Sistemas
ST0244 Lenguajes de Programación
Segundo parcial (20 %)

Nombre: _____ **Código:** _____ **Nota:** _____

Profesor: Alexánder Narváez Berrío **Clase n.º:** 6143 **Fecha:** 2024-09-27 **Duración:** 90 m.

1. (25%) **Smalltalk:**

   (a) (15%) Por favor indique de qué forma el siguiente código en Smalltalk está implementando el polimorfismo.

   (b) (10%) Favor indicar en qué línea se está aplicando la herencia, qué métodos tienen retorno y cuáles no.

```
Object subclass: Animal [

  Animal >> expresarse [
    "Este método será sobrescrito por las subclases"
  ]
]

Animal subclass: Perro [
  Perro >> expresarse [
    ^ 'Guau Guau'
  ]
]

Animal subclass: Gato [
  Gato >> expresarse [
    Transcript show: 'Miau Miau'; cr.
  ]
]
]
```

2. (25%) **Prolog**: Dada la siguiente base de hechos en Prolog:

(a) (10%) Indique las consultas respectivas que escribirías en el prompt de Prolog para saber quiénes son los padres de la Chilindrina, cuáles son los hijos del Profesor Jirafales.

(b) (15%) Cree una regla para hermana y hermano.

```
parent(don_ramon, quico).
parent(marina, chilindrina).
parent(senor_barriga, ñoño).
parent(profesor_jirafales, pedro).
parent(profesor_jirafales, ernesto).
parent(florinda, quico).
parent(padre_quico, quico).

% Relaciones de género
male(chavo).
male(quico).
male(senor_barriga).
male(profesor_jirafales).
male(ñoño).
male(padre_quico).
female(chilindrina).
female(marina).
female(florinda).

% Definiciones de relaciones
father(X, Y):-parent(X,Y), male(X).
```

mother(X, Y):-parent(X,Y), female(X).

3. (25%) **C++:** Dado el siguiente código en C++, argumente:

   (a) (10%) ¿Qué es una clase virtual en C++?

   (b) (15%) ¿Cuál es la diferencia entre un método virtualmente puro vs un método virtual regular?

```
class Mamifero {
public:
   virtual void amamantar();
};

class Oviparo {
public:
   virtual int ponerHuevos()  = 0;
};
```

4. (25%) **Punteros en C++:** Considera el siguiente código en C++:

```
#include <iostream>
using namespace std;

int main() {
   int x = 10;
   int* p = &x;

   *p = 20;
   cout << "Valor de x: " << x << endl;

   *p = 30;
   cout << "Valor de x: " << x << endl;

   int y = 40;
   *p = y;
   cout << "Valor de x: " << x << endl;
   cout << "Valor de y: " << y << endl;

   return 0;
}
```

Preguntas:

(a) (15%) ¿Cuál es la salida del programa y por qué?

(b) (5%) Después de ejecutar este código, ¿a qué dirección de memoria apunta el puntero p?

(c) (5%) ¿Qué pasaría si agregamos **p = &y**; antes de modificar ***p** por tercera vez? Explica detalladamente el comportamiento.

"Los buenos programadores saben qué escribir. Los grandes saben qué reescribir y reutilizar."

— Eric S. Raymond