FINAL LENGUAJES DE PROGRAMACIÓN I – 1era. Diciembre 2019

Criterio de corrección: **para aprobar el examen el alumno debe:**

- · Calificar con **bien** el ejercicio **1** o el ejercicio **2**
- · **No** puede tener **mal** ni el ejercicio **1** ni el ejercicio **2**
- · **No** puede calificar con **mal dos ejercicios** en todo el examen

Ejercicio 1. Dada la siguiente gramática en la que las reglas comienzan con '?' seguidas del no terminal de definición, seguido del símbolo ':' (se define como) y a continuación una secuencia de no terminales en minúscula y terminales entre comillas dobles. Suponer definiciones convencionales para los no terminales: id, iteración, seleccion y asignación, por ejemplo las del lenguaje Java.

1. ?programa: declaraciones instruccionesejecutables
2. ?declaraciones: declaracionesvariables declaracionesfunciones
3. ?declaracionesvariables: tipo id | declaracionesvariables tipo id
4. ?declaracionesfunciones: definicionfuncion | declaracionesfunciones definicionfuncion
5. ?definicionfuncion: tipo id "(" parametrosformales ")" cuerpo
6. ?cuerpo: "{" declaraciones instruccionesejecutables retorno "}"
7. ?retorno: "return" id
8. ?parametrosreales: parametroreal | parametrosreales ";" parametroreal
9. ?parametroreal: id | invocacionfuncion
10. ?invocacionfuncion: id"("parametrosreales")"
11. ?tipo: "int" | "float" | "auto"
12. ?parametrosformales: parametroformal | parametrosformales ";" parametroformal
13. ?parametroformal: parametroformalvariable | parametroformalfuncion
14. ?parametroformalvariable: modificadorsemantica tipo id
15. ?parametroformalfuncion: encabezadofuncion
16. ?encabezadofuncion: tipo id "(" parametrosformales ")"
17. ?modificadorsemantica: "in" | "out" | "inout" | "ref" | "name"
18. ?instruccionesejecutables: iteracion | seleccion | asignacion

Responder los siguientes incisos, indicando el conjunto de reglas que lo permiten. En caso contrario, modificar la gramática y/o proponer reglas nuevas para que el lenguaje lo permita:

i)
- **a)** *El lenguaje admite que una función no retorne un resultado*
- **b)** El lenguaje posee diferentes semánticas de pasaje de parámetros para variables
- **c)** El lenguaje posee diferentes semánticas de pasaje de parámetros para funciones
- **d)** El lenguaje permite recibir una función como parámetro
- **e)** El lenguaje permite retornar el valor resultante de ejecutar una función
- **f)** El lenguaje permite intercalar declaraciones de funciones y variables en cualquier lugar del programa

ii) Para los siguientes programas indicar si son aceptados o no con la gramática original, fundamentando por qué. En caso de no ser aceptado proponer modificaciones a las reglas para que se acepte.

a) int a(in int b;int a(out int b)) {int a x=3 return a(a)} x=3

b) int b(int a(int b)) {auto a x=3 return b} x=3

Ejercicio 2. Responder, acerca de pila de ejecución y memoria en lenguajes tipo Algol, si las siguientes afirmaciones son verdaderas o falsas, **justificando todas las respuestas.**

a) No puede existir en la pila de registros de activación, una variable cuyo alcance sea mayor que su tiempo de vida pues las variables desaparecen al desapilarse el registro de activación que la contiene

b) Los punteros de las variables dinámicas con nombre pueden producir punteros colgados si el lenguaje posee asignaciones entre punteros

c) Es posible que se produzcan punteros colgados tanto desde como hacia la pila de registros de activación

d) Si un lenguaje permite hacer declaraciones y asignaciones con aritméticas de punteros, es posible hacer que un bloque del heap apunte a una dirección de la pila. Esto podría producir, potencialmente, más garbage y/o punteros colgados

e) Una unidad anónima puede estar al alcance de las unidades padre e hija pero no puede ser alcanzada por una unidad nieta

f) El alcance de una variable contenida en el registro de activación de una unidad anónima siempre es menor que si esa unidad fuese nombrada

g) Siempre las variables dinámicas anónimas y las variables estáticas tienen un tiempo de vida mayor que su alcance

h) Si una variable estática de tipo puntero apunta a otra variable estática, entonces el alcance de ambas variables coincide y también el tiempo de vida

Ejercicio 3. Considerando un lenguaje con orientación a objetos (al estilo C++) y un programa que posee un objeto p1 de la clase P y un objeto h1 de la clase H que hereda de P, responder los siguientes incisos **fundamentando las respuestas.**

a) h1 tiene en su almacenamiento los atributos de la clase H y un puntero a un objeto de la clase P para poder acceder a los atributos heredados.

b) Si un método es declarado como estático, su tiempo de ligadura para la llamada es el mismo que el de una función convencional.

c) A h1 puede asignarse p1 porque este último posee todos los atributos necesarios para la asignación y además cumple con la relación "es un" por lo tanto la compatibilidad por nombre es válida.

d) p1 puede ser asignado a h1 porque la estructura del primero está contenida en la del segundo por lo tanto al ser una variable menos abarcativa no hay problema en la asignación.

e) La indirección debida a la tabla de despacho permite implementar la ligadura dinámica de llamado a métodos y reducir el espacio necesario para almacenar las direcciones de los métodos.

Ejercicio 4. Dado el siguiente fragmento de programa escrito en C++ 11, del cual se sabe que compila correctamente

```
1.  static int t = 5;
2.  void XX(string A){
3.    int t = t;
4.    int arreglo[t];
5.    int cantidad = sizeof(arreglo)/sizeof(int);
6.    imprimir(msg,
        "Th "<<A<<" 1er Elem. -> "<<&arreglo[0]<<endl
        <<"Th "<<A<<" Ult. Elem. -> "<<&arreglo[cantidad-1]<<endl
      );
7.    if (t-->0){
8.      this_thread::sleep_for(std::chrono::milliseconds(50));
9.      XX(A);
10.   }
11. }
12. int main (){
13.     thread th0(XX,"0");
14.     thread th1(XX,"1");
15.     th0.join();
16.     th1.join();
17.     return 0;
18. }
```

El código ha sido testeado por distintos programadores y los mismos aseguran que han obtenido las siguientes salidas

Salida 1	Salida 2	Salida 3
Th 0 1er. -> 0x70000bf6ddd0 Th 0 Ult. -> 0x70000bf6ddd0 Th 1 1er. -> 0x70000bff0dd0 Th 1 Ult. -> 0x70000bff0dd0 Th 1 1er. -> 0x70000bff0cd0 Th 1 Ult. -> 0x70000bff0ccc Th 0 1er. -> 0x70000bf6dcd0 Th 0 Ult. -> 0x70000bf6dccc	Th 1 1er. -> 0x70000c5e6dc0 Th 1 Ult. -> 0x70000c5e6dd0 Th 0 1er. -> 0x70000c563dc0 Th 0 Ult. -> 0x70000c563dd0 Th 1 1er. -> 0x70000c5e6ca0 Th 1 Ult. -> 0x70000c5e6cb0 Th 0 1er. -> 0x70000c563ca0 Th 0 Ult. -> 0x70000c563cb0 Th 0 1er. -> 0x70000c563b80 Th 0 Ult. -> 0x70000c563b90 Th 1 1er. -> 0x70000c5e6b80 Th 1 Ult. -> 0x70000c5e6b90 Th 0 1er. -> 0x70000c563a60 Th 0 Ult. -> 0x70000c563a70 Th 1 1er. -> 0x70000c5e6a60 Th 1 Ult. -> 0x70000c5e6a70	Th 0 1er. -> 0x700004f21d30 Th 0 Ult. -> 0x700004f3dd2c Th 1 1er. -> 0x700004fa4d30 Th 1 Ult. -> 0x700004fc0d2c Th 0 1er. -> 0x700004f21b80 Th 0 Ult. -> 0x700004f21b7c Th 1 1er. -> 0x700004f88b80 Th 1 Ult. -> 0x700004fa4b7c Th 1 1er. -> 0x700004f6c9d0 Th 1 Ult. -> 0x700004f889cc Th 1 1er. -> 0x700004f50820 Th 1 Ult. -> 0x700004f6c81c Segmentation fault: 11

a) Explique brevemente lo que ocurre en cada salida y los motivos por los cuales pueden producirse.

b) Si ninguna de las salidas anteriores fuese posible ¿cree que podría hacer modificaciones mínimas ***(sin modificar declaraciones de variables)*** para que lo fueran? Fundamente su respuesta.

c) ¿Existen riesgos de conflicto de acceso debido a la concurrencia en las salidas expuestas? En caso de que existan ¿cómo los resolvería? Si tiene que emplear semáforos, indique entre qué líneas los ubicaría.

d) Si la línea 3 cambia por estas dos instrucciones

```
int *x= new int(t);
int t = *x;
```

¿En qué cambiaría la ejecución del programa y sus resultados? Justifique brevemente

e) Clasifique las variables, de acuerdo a su almacenamiento, que intervienen e indique en qué lugar se almacenan las mismas, teniendo en cuenta las modificaciones del inciso anterior.