Ejercicios de examen: segundo parcial

1. (0.5 ptos) Escribir las acciones semánticas necesarias para la propagación de tipos en la siguiente gramática:

$$Declarac \rightarrow NombTipo : ListaIDs$$
$$ListaIDs \rightarrow ident$$
$$ListaIDs \rightarrow ListaIDs, ident$$
$$NomTipo \rightarrow float$$
$$NomTipo \rightarrow int$$

   con las siguientes restricciones:

   (a) Sólo pueden usarse atributos sintetizados

   (b) Se asumen las funciones típicas de la TDS: $insertarTDS(nombre, Tipo)$ y $buscarTDS(nombre)$.

   (c) Sólo se considerarán atributos, referenciados por $NombreSimbolo_{subind}.NombreAtributo$.

2. (0.75 ptos) Explicar brevemente cuales son las posibilidades de evaluación de atributos respecto a la decoración del árbol sintáctico, indicando las ventajas e inconvenientes de cada una.

3. (0.5 ptos) ¿Permite un analizador LR(1) la decoración del árbol durante el análisis para gramáticas S-atribuídas y LC-atribuídas? Razonar la respuesta.

4. (0.75 ptos) ¿Cuales de las siguientes variables tienen tipos equivalentes según el modelo de equivalencia estructural? ¿Por qué?

```
type
  t_vector: array [0..9] of real;
  t_matriz : array [0..9] of array [0..9] of real;
  t_registro: record
              X: array[0..9] of real;
              end;
var
  uno:    array [0..9] of t_vector;
  dos:    array [0..9] of t_registro;
  tres:   array [0..9, 0..9] of real;
  cuatro: t_matriz;
```

5. (0.75 ptos) Escribir las acciones semánticas para la conversión implícita de tipos en la siguiente gramática.

$$Exp \rightarrow num\_int$$
$$Exp \rightarrow num\_real$$
$$Exp \rightarrow id$$
$$Exp \rightarrow Exp + Exp$$

6. (0.5 ptos) En qué circunstancias se puede incorporar la TDS al código objeto?

7. (0.75 ptos) Describir las opciones existentes para la gestión de tipos registro en TDSs.

8. (1 pto) **Explicar** cual es la mejor opción (tablas hash o árboles) para la implementación de la TDS en lenguajes con estructura de bloques:

   - usando una única tabla.
   - mediante una pila de tablas.

9. (0.75 ptos) Describir las ventajas derivadas del uso de representaciones intermedias en compiladores.

10. (0.75 ptos) Describir la RI basada en notación polaca inversa, enumerando sus ventajas e inconvenientes.

11. (1 pto) Dado el siguiente código intermedio de 3 direcciones:

```
        ...
        valor = 1000
        acc1 = 0
        acc2 = 0
        i = 1
test:   if i > 100 goto E3
        t1 = 4*i
        t2 = a[t1]
        t3 = 2 * t2
        acc1 = acc1 + t3
        if i <> 30 goto E1
        t4 = 4 * i
        t5 = acc1 + acc2
        c[t4] = t5
        goto E2
E1:     t6 = 5 * valor
        t7 = 9 * i
        acc2 = t6 + t7
E2:     i = i + 1
        goto test
E3:     suma = acc1+acc2
        ...
```

(a) (0.4 ptos) Dividir el C.I. en bloques básicos y dibujar el diagrama de flujo.

(b) (0.6 ptos) Señalar las posibles optimizaciones locales sobre el C.I. y escribir el C.I. optimizado.

12. (1.25 ptos) Dado el siguiente fragmento de código C:

```
int A[200];
int B[100];
...
 suma = 0;
 for(i=0, j=100; i < 100; i++, j--) {
     if ((A[j*2]+B[i]) > 350) {
         k = 0;
         while (k < i ) {
             B[i] = B[i] + A[k*2];
             k++;
         }
     }
     suma = suma + B[i];
 }
 final = (suma - 100);
...
```

(a) Generar su representación intermedia usando código de tres direcciones. (NOTA: Tamaño enteros: 4 bytes)

(b) Señalar sus bloques básicos y construir el diagrama de flujo.

13. (0.75 ptos) Describir los tres tipos de representaciones de código de tres direcciones, señalando sus ventajas e inconvenientes.

14. (1.25 ptos) Dado el siguiente fragmento de código en tres direcciones, parcialmente optimizado (p.e. en las variables de inducción):

```
...
       suma=0
       i=0
       j=100
etq1: if i<100 goto etq5
       goto etq7
etq2: t1 = j<<1
```

```
        t2 = t1<<2
        t3 = A[t2]
        t4 = i<<2
        t5 = B[t4]
        t6 = t3+t5
        if t6 > 35 goto etq3
        goto etq6
etq3:   k=0
etq4:   if k<i goto etq5
        goto etq6
etq5:   t7 = B[t4]
        t8 = k<<1
        t9 = t8<<2
        t10 = A[t9]
        t11 = t7+t10
        B[t4]=t12
        k=k+1
        goto etq4
etq6:   t12 = B[t4]
        suma=suma+t12
        i=i+1
        j=j-1
        goto etq1
etq7:   final = suma-100
...
```

(a) (0.75 ptos) Señalar sus bloques básicos y construir el diagrama de flujo.

(b) (0.25 ptos) Construir el código inicial en lenguaje C.
NOTA: A y B son arrays de 100 enteros, y el tamaño de los enteros en memoria es 4 bytes

(c) (0.25 ptos) Construir el GDA para el BB que comienza en `etq2`

15. (0.5 ptos) ¿Qué condiciones deben cumplirse para que un compilador genere código objeto sólo con asignación estática de memoria, incluso admitiendo procedimientos? Razone la respuesta.

16. (1 pto) Describir la disposición típica de los programas en memoria.

17. (0.75 ptos) Enumerar los pasos de una secuencia de llamada típica a un procedimiento/función.

18. (1.5 ptos) Razonar si las siguientes afirmaciones son verdaderas o falsas:

    (a) *Si al analizador es precedencia simple y la gramática de atributos L-atribuída, la decoración del árbol puede realizarse durante el análisis.*

    (b) *Es posible generar código objeto sin pila para un lenguaje con funciones y/o procedimientos.*

    (c) *La mejor opción para implementar múltiples TDS en un lenguaje con estructura de bloques es mediante árboles binarios.*

19. (0.5 ptos) ¿Qué condiciones deben cumplirse para que un compilador genere código objeto sólo con asignación estática de memoria, incluso admitiendo procedimientos? Razone la respuesta.

20. (0.75 ptos) Describa los campos típicos de un registro de activación

21. (0.75 ptos) Describa los modelos de desasignación de espacio en el montículo (*heap*)?