Compiladors: Examen de laboratori de Asl

4 d'abril de 2016

Duració de l'examen: fins les 9:55h (no es podrà fer cap entrega més enllà d'aquesta hora).

Cal enviar l'intèrpret complet (tot el directori `Asl`) en un fitxer `tgz` i fer l'entrega a través del Racó (examens.fib.upc.edu).

Al Racó trobareu un fitxer `examen.tgz` que conté:

- Aquest enunciat
- Intèrpret ASL original (Asl.tgz)
- Jocs de proves per validar els exercicis proposats a l'examen, amb les sortides esperades de la seva execució i els arbres abstractes corresponents.
- Un script `avalua.sh` que executa l'ASL sobre els jocs de proves calcula la nota de l'examen.
- Un script `empaqueta.sh` que crea un fitxer `entrega.tgz` per pujar al Racó

Notes importants:

1. Els fitxers `.ast` dels jocs de proves es donen com a referència, no és necessari que la vostra solució els creï exactament igual. Per tant, si l'avaluador dona `OK` per a `Output`, el problema es considera correcte, encara que la columna `AST` sigui `NO`.
2. Al fer el problema 3, canvia l'estructura de l'arbre de les assignacions, causant que l'avaluador doni `NO` a la columna `AST` dels problemes 1 i 2. Això no suposa cap diferència en l'avaluació.
3. Els tres problemes son independents i es poden fer en qualsevol ordre. De tota manera, es recomana deixar el tercer problema (assignació múltiple) pel final. Sigui quin sigui l'ordre, cal *acumular* les solucions en el mateix codi, i no fer-les per separat.
4. La sortida dels jocs de prova generada per la vostra solució ha de coincidir *exactament* amb els fitxers `.out` donats. Altrament, l'avaluador no considerarà correcte l'exercici. Pareu especial atenció a l'escriure els missatges d'error.

1 Operador factorial

Afegiu un nou operador "`!`" a l'intèrpret, que calculi el factorial d'un enter.Es tracta d'un operador unari postfix (és a dir, va darrera de l'operand), i té més prioritat que qualsevol altre operador.

Modifiqueu l'intèrpret per incloure el nou operador.

Només cal modificar els fitxers `Asl.g` i `Interp.java`.

Joc de proves 1: (1 punt)

```
func main()
  x = 3!;
  write "Factorial de 3 = ";
  write x; write "%n";

  y = 2*2 + 3!;
  write "2*2 + factorial de 3 = ";
  write y; write "%n";

  z = 3!*2 + 1;
  write "factorial de 3 * 2 + 1 = ";
  write z; write "%n";
endfunc
```

Joc de proves 2: (1 punt)

```
func main()
  x = 3;
  y = x!;
  write "El factorial de "; write x; write " es ";
  write y; write "%n";

  z = 9-x!;
  write "9 - el factorial de "; write x; write " es ";
  write z; write "%n";

  z = (y-9)!;
  write "El factorial de "; write y; write " - 9 es ";
  write z;  write "%n";
endfunc
```

Joc de proves 3: (1 punt)

```
func main()
  x = 3;
  z = 1 + 2*(9-x!)!!;
  write "la formula 1 + 2*(9-3!)!! dona: ";
  write z; write "%n";

  n=0;
  while n<3 do
    z = n!!!;
    write "El requetefactorial de "; write n; write " es ";
    write z;  write "%n";
    n = n+1;
  endwhile
endfunc
```

2 Funció predefinida `suma`

Afegiu una funció predefinida `suma` a l'intèrpret, que calculi la suma de la llista d'arguments rebuda.

Modifiqueu l'intèrpret per incloure la nova funció predefinida.

Només cal modificar els fitxers `Asl.g` i `Interp.java`.

Joc de proves 1: (1 punt)

```
func main()
  write "The sum of nothing is ";
  write sum();
  write "%n";

  x = 4;
  write "The sum of 4 and 5 is ";
  write sum(x, 5);
  write "%n";

  write "The sum of 4, 5, and 6 is ";
  write sum(x, 5, x+2);
  write "%n";
endfunc
```

Joc de proves 2: (1 punt)

```
func main()
  x = 2;
  y = 1;
  write "The sum is ";
  write sum(x, y, x*y-1);  write "%n";

  z = sum(2*x,10);
  write "The second sum is ";
  write sum(x, y+2, z>0 or y<2);  write "%n";

  write "Yet another: ";
  write sum(x, y, -z-x-y, z);  write "%n";
endfunc
```

Joc de proves 3: (1 punt)

```
func main()
  x = 4;
  y = 5;
  write "The sum is ";
  write sum(x, y);  write "%n";

  z = sum(2*x,10);
  write "The second sum is ";
  write sum(x, y+2, z);  write "%n";

  t = 14;
  write "Yet another: ";
  write sum(x, y, z, t+sum(1,2,3) );  write "%n"
endfunc
```

Joc de proves 4: (1 punt)

```
func main()
  x = 2;
  y = 4;
  write "Result is ";
  write sum(x,y)! - sum(x!!,y!);  write "%n";

  z = sum(2*x,sum(x+y!));
  write "Second result is ";
  write (sum(z)/15*3!)!;  write "%n";
endfunc
```

3 Assignació múltiple

Generalitzeu la instrucció d'assignació del llenguatge Asl perquè permeti assignar una llista d'expressions a una llista de variables amb el mateix número d'elements.

Podeu reduïr l'assignació simple al cas particular d'una assignació múltiple d'un sol valor a una sola variable, evitant així duplicitat a la gramàtica. Es recomana crear dos nodes a l'arbre a sota de l'assignació: un que tingui com a fills la llista d'identificadors de la part esquerra, i un altre que tingui com a fills la llista d'expressions de la part dreta.

Per crear aquests nodes imaginaris, podeu basar-vos en com la regla `block_instructions` de `Asl.g` crea el node imaginari `LIST_INSTR` que té com a fills la llista d'instruccions.

Modifiqueu l'intèrpret per incloure la nova estructura.

Només cal modificar els fitxers `Asl.g` i `Interp.java`.

Joc de proves 1: (1 punt)

```
func main()
  n, m, p = 2+2, 34, 45;
  n = n + 1;
  write "n: "; write n; write "%n";
  write "m: "; write m; write "%n";
  write "p: "; write p; write "%n";
  write "%n";

  x, y = m+12-p, n*3;
  write "x: "; write x; write "%n";
  write "y: "; write y; write "%n";
endfunc
```

Joc de proves 2: (1 punt)

```
func main()
  n, m = 2+3, 3*4;
  write "n: "; write n; write "%n";
  write "m: "; write m; write "%n";

  x, y, z = m+12-n, n*3;
  write "x: "; write x; write "%n";
  write "y: "; write y; write "%n";
  write "z: "; write z; write "%n";
endfunc
```

Joc de proves 3: (1 punt)

```
func main()
  x, y, z = 5, 10, 15;
  write "x1: "; write x; write "%n";
  write "y1: "; write y; write "%n";
  write "z1: "; write z; write "%n";

  x, y, z = z, x, y;
  write "x2: "; write x; write "%n";
  write "y2: "; write y; write "%n";
  write "z2: "; write z; write "%n";

  x, y, z = z+x, x+y, y+z;
  write "x3: "; write x; write "%n";
  write "y3: "; write y; write "%n";
  write "z3: "; write z; write "%n";
endfunc
```