Compiladores - LCC
Práctica 0.
Introducción a Tiger

1.

El record `lista = {item: int, resto:lista}` puede usarse para representar una lista de enteros. Implemente las siguientes dos funciones:

- `agrega` que recibe dos argumentos, un entero y una lista. Devuelve la lista formada por el entero seguido de la lista
- `filtra` que recibe un entero n y una lista l. Devuelve una lista formada por los elementos de l que no son iguales a n, manteniendo el orden de l.

```
let
    type lista = {item:int, resto:lista}
    /* Definir agrega */
    /* Definir filtra */
in
    nil
end
```

2.

Definir las siguientes funciones:

- **cantprints:** dado un *tigerabs.exp* devuelve la cantidad de llamadas a la función *print* cuyo argumento no es un *StringExp*.
- **cantplus:** dado un *tigerabs.exp*, contar la cantidad de veces que se utiliza la operación binaria de la suma.

3.

Encuentre el AST que corresponde a los siguientes fragmentos de código. Puede usar la primera versión del compilador, comentando la línea
`val _ = findEscape(expr)` en *tigermain.sml*.

a) `a := 10`

b) `for i := 0 to c do print (".")`

c) `f[a+1].data[0]`

d)
```
let
    var f := 10
in
    f(f, f); f
end
```

e) `type lista = {item:int, resto:lista}`

f) `if row[r]=0 & a<b then g(r)`

¿Todos los fragmentos pueden ser parte de un programa válido? ¿Por qué hay un problema al copiar directamente el fragmento e? Descomente la línea comentada anteriormente. ¿Qué error detecta ahora el compilador?

4.

Encuentre el código que genera los siguientes ASTs. No tome en cuenta el valor del campo `pos`.

a) `ArrayExp({init = IntExp(5, 0), size = IntExp(10, 0), typ = "a"}, 0)`

b) `VarExp(SubscriptVar(SimpleVar "a", IntExp(7, 0)), 0)`

c) `AssignExp({exp = NilExp 0, var = SimpleVar "a"}, 0)`