CS3470 Assignment 4 2017/18

This assignment should be submitted in PDF format via Moodle by **2pm** on **Monday 4th December**. Remember solutions must be your own work.

1.[**30 marks**] Use the attribute grammar (syntax-directed definition) given in Section 8.2.1, page 107, of the Notes to create an annotated parse tree (as described in Section 7.4) from the code fragment

```
y = (1 + x) * 3
```

You should assume that there is a lexer which produces the following stream of tokens which is parsed `ID = ( num + ID ) * num`
Then walk the annotated tree and evaluate all the attributes. (Give the values for all attributes not just that last one.)

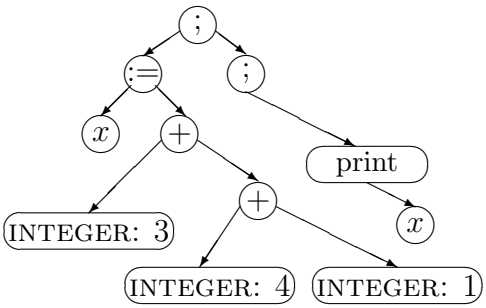
2.[**30 marks**] Consider the grammar

```
TREE
P ::= S ';' [ P ].
S ::= ID ':=' E | 'print' '(' ID ')' .
E ::= INTEGER [ '+' E ] .
```

(i) Add `rdp` promotion operators so that for input

```
x := 3+4+1 ; print(x) ;
```

the following reduced derivation tree is produced by `rdp`.



(ii) (For this part look at Section 7.7 of the Notes and Lab Sheet 2). For the original grammar above add an `rdp` symbol table and semantic actions so that a statement of the form `x := e` computes the value of the expression e and stores it in the symbol table entry for x, and `print(x)` checks that x has been given a value and then prints this value.
Give the output of running your solution through `rdp` with the string
`x := 3+4+1 ; print(x) ;`

PTO/..

3.[**15 marks**] Use loop fusion to improve the following fragment of three address code, writing out the resulting flow graph.

```
     k := 0
     i := 7
     tot := 5
L1 : m := i > 40
     if m goto L2
     k := k + 2
     tot := tot * k
     i := i + 1
     goto L1
L2 : x := 0
     sq := 2
L3 : m := x > 40
     if m goto L4
     sq := sq * sq
     x := x + 1
     goto L3
L4 :
```

4.[**25 marks**] Construct a directed acyclic graph (DAG) for the following block of code and then use it to reconstruct an equivalent block of three address code. (Assume that `t1, t2, t3, t4, t5` are the identifiers that are dead outside the block.)

```
      b := k + 1
     t1 := n + 2
     t2 := k[t1]
     t3 := b
     t4 := t3 * m
      x := m + 2
     t5 := x + n
     t2 := t5
      k := t2 - x
      m := 1 + k
```

PTO/..

Assessment Details And Criteria

This assignment is worth 5% of the final course mark. The questions on the assignment are not all of the same weight. The marks for each question are shown on the sheet.

For Question 1 you should write the semantic rules associated with each node of the derivation tree. Then evaluate the attributes writing out the value of each attribute as it is computed. For Question 3 you should not assume that variables i and m are dead outside the code fragment. For Question 4 you must reconstruct the code from your DAG using the algorithm described in the notes. Do not peform any additional or alternative optimisations before or during the reconstruction.
For all questions, some marks will be given for 'correct working or approach' even when the final answer is not correct.
You should try all the questions and hand in what you have done, even if you are not able to complete some of the questions.

Submission and feedback arrangements

• Solutions must be submitted in PDF format via the links on the course Moodle page. The submissions must be PDFs not jpeg, Word documents or any other format. You may hand write and then scan your solutions to obtain a PDF file.
Write your name on every page and make sure the printed version is legible.
The mark will be set to 0 if the submission is in the wrong format or not legible.

• In the event of central IT issues at the time of the deadline, inform the course lecturer.

• The assignments will be marked and a feedback grade given. Note, the feedback grades are provisional and subject to change by our External Examiners during the examination moderation process.

• General feedback will also be posted on the course webpage in January 2018.

Learning outcomes assessed

The primary purpose of this assignment is formative. It is designed to

- ◇ provide practice in building attribute trees and evaluating the attributes and test your understanding of intermediate code generation using annotated grammars
- ◇ test your ability to improve code using a DAG of operand dependencies
- ◇ test your understanding of reduced derivation tree specification and semantic action use in `rdp`
- ◇ provide you with a way of assessing for yourselves the degree to which you have understood the material

If you do not know how to do a question, begin by reading the corresponding text in the notes and working through related examples discussed in lectures. Please feel free to make an appointment to talk to me about things you don't understand, and remember all solutions must be your own work.