5CCS2PLD: Programming Language Design Paradigms

Coursework 1

Submission deadline: 1 March 2019

ANSWER ALL THE QUESTIONS

Question 1:

The abstract syntax of an imperative language has been defined using the grammar:

$$\begin{array}{lll} C & ::= & skip \mid l := E \mid C; C \mid if\ B\ then\ C\ else\ C \mid while\ B\ do\ C \mid do\ E\ times\ C \\ E & ::= & !l \mid n \mid E\ op\ E \\ op & ::= & + \mid - \mid * \mid / \\ B & ::= & True \mid False \mid E\ bop\ E \mid \neg B \mid B \wedge B \\ bop & ::= & > \mid < \mid = \end{array}$$

The commands are similar to those of SIMP, with the addition of the command

$$do\ E\ times\ C$$

The following rules define the big step semantics for this new command.

$$\frac{\langle E, s\rangle \Downarrow \langle 0, s'\rangle}{\langle do\ E\ times\ C, s\rangle \Downarrow \langle skip, s'\rangle}$$

$$\frac{\langle E, s\rangle \Downarrow \langle n, s_1\rangle \quad \langle C, s_1\rangle \Downarrow \langle skip, s_2\rangle \quad \langle do\ (n-1)\ \ times\ \ C, s_2\rangle \Downarrow \langle skip, s_3\rangle}{\langle do\ E\ times\ C, s\rangle \Downarrow \langle skip, s_3\rangle}\ \texttt{if}\ n \neq 0$$

1. Consider the program P:

$$do\ \ (!x - 1)\ \ times\ \ y := !y + 1$$

   (a) Assume s is a memory where x and y have both the value 100. What are the values of x and y after the execution of the program P defined above?

   (b) Now assume that, in the memory s, x has the value 0 and y has the value 100. Describe the behaviour of the program P when executed in s.

2. Write a SIMP program equivalent to the program P defined above. Your program can use more variables, but should produce the same results for all the variables mentioned in P. Justify your answer.

Question 2:
We add to the language SIMP, studied in the course, a new class of expressions that will be used to represent lists of numbers. The abstract syntax of list expressions is defined by the grammar:

$$L ::= \texttt{nil} \mid \texttt{cons}(E, L) \mid \texttt{concat}(L_1, L_2)$$

where `nil` denotes an empty list and `cons`(E,L) represents a list where the first element is the result of evaluating the expression E and the rest of the list is defined by L. For example, the list containing the numbers 1, 2, and 3 is represented by `cons`(1, `cons`(2, `cons`(3, `nil`))). The operator `concat` concatenates two lists, so the expression `concat`(L,S) represents the concatenation of the lists denoted by L and S.
In addition, a new selector command is included in the language to discriminate between an empty list and a non-empty one:

```
case L
   empty:    C1
   nonempty: C2
```

where the list expression L will be evaluated, and if the result is an empty list then C_1 will be executed otherwise C_2 will be executed.

1. Give a formal definition of the semantics of this command (you can give transition rules for the abstract machine, or rules to extend the small-step semantics or the big-step semantics of SIMP).

2. A compiler for this language has been optimised so that the machine code produced for commands of the form

   ```
   case nil
      empty:    C1
      nonempty: C2
   ```

   is identical to the machine code generated for C_1. Is this correct? Justify your answer.