

# School of Computer Science and Software Engineering

## CSE3322 Programming Languages and Implementation

### Assignment 3

Due 12 noon Friday 17th of October

The purpose of this assignment is to write a compiler for a very simple language using `flex` and `bison`. You are to work in pairs.

Your boss has developed a new desktop calculator programming language called `dec` for calculating with floating point numbers. It has the following commands:

**input** which reads a number from standard input into a variable,

**output** which writes the value of a numerical expression to standard output,

**=** which assigns the value of a numerical expression to a variable, and

**skip** which does nothing.

It has control statements `;` and *if-then-else* and *while* which behave much like the corresponding `C` control statements.

*if-then-else* and *while* take a relational expression as an argument. This is built from the relational operators

`>` strict greater than

`>=` greater than or equal to

`<` strict less than

`=<` less than or equal to

`=` equal to

`<>` not equal

using the standard Boolean connectives *not*, *and*, *or* and parenthesis.

She wants you to develop a UNIX utility called `decc` which compiles program in this language to `C`.

For example, a `dec` program for computing the maximum of two input floats is

```
{ input x;  
  input y;  
  if ( x >= y ) then {output x} else {output y}  
}
```

The language has partial grammar

```
<program> ::= <stmtseq>
<stmtseq> ::= { <stmts> }
<stmts> ::= <statement> | <statement> ; <stmts>
<statement> ::= <ident> = <expr>
                | if ( <relexpr> ) then <stmtseq> else <stmtseq>
                | while ( <relexpr> ) <stmtseq>
                | input <ident>
                | output <expr>
                | skip
<expr> ::= <expr> + <expr>
          | <expr> - <expr>
          | <expr> * <expr>
          | <expr> / <expr>
          | ( <expr> )
          | <number>
          | <ident>
```

Note that `<program>`, `<stmtseq>`, `<stmts>`, `<statement>`, `<expr>`, and `<relexpr>` are non-terminal symbols. The terminal symbol `<ident>` represents an identifier which is a non-empty sequence of lower or upper case letters while the terminal symbol `<number>` is a non-empty sequence of digits with an optional “`~`” sign at the front to indicate a negative number and an optional decimal point in the middle.

The different parts are:

- (a) Complete the grammar by giving production rules to specify relational expressions. That is, define the non-terminal `<relexpr>`. [1 mark]
- (b) Write a file `dec.flex` which is to be used by `flex` to generate a lexical analyzer for this simple language. [2 marks]
- (c) Write a file `dec.y` which is to be used by `bison` to generate a parser for this language. The generated parser should be called `decc`. [3 marks]
- (d) Extend your grammar `dec.y` so that it gives meaningful syntax error messages including the line number of the error and continues parsing after finding a syntax error. [Hint: You may want to modify your lexical analyzer so it keeps the current line count in a global variable.] [3 marks]
- (e) Extend your grammar `dec.y` so that it generates the appropriate C code for the input program. That is, write a **compiler** for this language. [5 marks] [Hint: remember to appropriately declare all identifiers in the generated C program. You may assume identifiers are less than 10 characters in length and that there are no more than 100 in the input program.]
- (f) Write a `Makefile` for “making” your compiler `decc`. The command `make decc` should appropriately use `flex`, `bison` and `gcc` to create the utility `decc`. [1 mark]

For example, when given

```
{
    input k;
    input n;
    m = 1.0;
    while ( m >= n ) {
        if ( k = n or (k+1 = n and m=n )) then {
```

```

        skip
    } else {
        output m
    };
    m = m + 1
}
}

```

as input `decc` should generate something like:

```

#include <stdio.h>

main()
{
    double k,n,m;

    {
        scanf("%lf", &k);
        scanf("%lf", &n);
        m=1;
        while (m >= n){
            if (k == n || (k+1 == n && m == n )){
                ;
            } else {
                printf("%f", m);
            }
            m=m+1;
        }
    }
}

```

Note that indentation and white spaces are not important. However, for valid input programs, `decc` should generate valid C programs with the same behaviour as the input program. For invalid input programs it should stop with a syntax error.

You should work in pairs. If this is not possible then you need to obtain written permission from the lecturer to work by yourself. In this case you do not need to do Part (e) and your assignment will be marked out of 10.

## Submission Instructions

The above exercises contribute 15% to your total CSE3322 mark.

Submission will be electronic. You should use `/cs/cc/bin/submit` to submit a directory containing `dec.flex`, `dec.y`, the `Makefile` as well as any additional C files and test data files. The assignment name is `ass3`.

Each pair need only submit one copy of the assignment. Clearly indicate in comments in the files if the assignment was done in a pair or singly and who are the authors of the assignment (both name and student ID).

The assignment is due **12 noon Friday 17th of October**.

Your programs will be marked on correctness, style, efficiency and clarity.

Assignments handed in after the due date will attract a late penalty of 5% per day unless special consideration

applies or there has been prior agreement in writing from the lecturer. No submission will be accepted after Friday 24th of October.

Your submission must include the following declaration at the top of **each** file. If it does not your assignment will not be marked and you will receive 0 for the assignment.

(\* MONASH UNIVERSITY, School of Computer Science and Software Engineering  
Student Declaration for CSE3322 Submission

We #YOUR NAMES#, ID: #YOUR ID NUMBERS# declare that this submission is our own work and has not been copied from any other source without attribution.

We acknowledge that severe penalties exist for any copying of code without attribution, including a mark of 0 for this assessment.

\*)

Unless otherwise advised your assignment marks for CSE3322 are to be listed on the 3rd year notice board next to your student ID. Of course your name will not be on the list, only your ID and assignment mark. If you do not wish your mark for assignment 3 and student ID to appear on such a list please contact your lecturer for the subject by Friday 17th October to make alternative arrangements.