CSC402 – Midterm
Due in Sakai Monday 11/6/17
Version 1.1

1 Part I: Short Answer Problems (30 points)

1. Describe the difference between a high-level programming language and a low-level programming language.

2. Give a regular expression that describes words that start with the letters 'a' or 'z'.

3. Given the grammar:

```
exp : exp '@' exp
    | exp '#' exp
    | prim
prim : '<' W '>'
W : 'a' | 'b' | .. | 'z'
```

   Please answer the following:

   (a) Is this grammar LL(1) compatible? Why? Why not?

   (b) Does the string 'a@b#c' belong to the language of the grammar? Why? Why not?

   (c) If the '@' operator has a higher precedence than the '#' operator, how would you reflect that in grammar?

4. Given this grammar,

```
sen : rambling followed by one_ending
    | rambling followed by another_ending
    ;

rambling : 'a';
followed : 'b';
by : 'c';
one_ending : 'd';
another_ending : 'e';
```

Is the above grammar a LL(1) grammar? Why? Why not? If not, how would you rewrite it to make it a LL(1) grammar?

2 Part II: Programming Problems (70 points)

Basic is a programming language available available on many different hardware platforms. Here is the link from Wikipedia

`https://en.wikipedia.org/wiki/BASIC`

Here we define a subset of that language, call it μBasic, with the following features: [1]

The full grammar of the language is available in the code folder as ubasic_gram.py ane ubasic_lex.py.

- Program – A program is a list of statements
- Assignment Statement – An assignment statement has the form `ID = exp` and sets the value of the variable `ID` to the value of the expression `exp`.
- Input Statement – An input statement has the form `input STRING, ID` and prints out the optional prompt string `STRING`, waits for the user input, and set the value of the variable `ID` to the user value. The user is only allowed to enter integer values. Example:

  ```
  input "Enter a value for variable x: ", x
  ```

- Print Statement – The print statement is the keyword `print` followed by a comma separated list of one or more values. Example:

  ```
  input "Enter a value for variable x: ", x
  print "The value of variable x is ", x
  ```

- End Statement – The end statement terminates the execution of the program.
- If Statement – In the If-statement the conditional expression is followed by one or more then-statements which is optionally followed by one or more else-statements. Example:

[1] μBasic stands for microBasic.

```
input "enter a value: ", a
if a==5 then
   print "a is equal to 5"
else
   print "a is not equal to 5"
endif
```

- While Loop – In the while loop the conditional expression is followed by one ore more statements in the body of the loop. Example:

```
a = 1
while a <= 100
   print a
   a = a + 1
endwhile
```

- For Loop – The for loop initializes the loop variable with a start value and then at each loop iteration increases the loop variable by the amount given in the step value. The loop terminates once the loop variable reaches the end value. If the step value is missing then a step value of 1 is assumed. The step value can also be negative. Note, in the following example the single quote starts a comment line, which is the usual way to write a comment in Basic. Example:

```
' print a list of integers from 1 through 10
for x = 1 to 10
   print x
next x
```

Another example:

```
' print the even numbers between 1 and 10
for x = 2 to 10 step 2
   print x
next x
```

A final example:

```
' print the odd numbers in reverse order between 1 and 10
for x = 9 to 1 step -2
   print x
next x
```

- Expressions – Expressions are best summarized by the grammar productions:

```
exp : exp PLUS exp
    | exp MINUS exp
    | exp TIMES exp
    | exp DIVIDE exp
    | exp EQ exp
    | exp LE exp
    | exp AND exp
    | exp OR exp
    | INTEGER
    | ID
    | '(' exp ')'
    | MINUS exp %prec UMINUS
    | NOT exp
```

  The grammar encodes the standard precedences for these operators. We have the logical operators `&` (and) and `|` (or); the relational operators `==` (equal) and `<=` (less equal); the additive operators `+` (addition) and `-` (subtraction); and finally the multiplicative operators `*` (multiplication) and `/` (division).

  The logical operators work on integer values as follows, anything that is not equal to zero is interpreted as true and a value zero is interpreted as false. For example

$$0\&20 \Rightarrow 0$$

  but

$$-50\&20 \Rightarrow 1$$

  The `!` operator is the logical not operator and sets any value not equal to 0 to the value 0. Conversely it sets a value 0 to the value of 1.

Here is another example program pulling this all together:

```
' compute the factorial of a number
input "Enter a value: ", x
if x <= 0 then
   print "illegal input value"
   end
endif

' input ok - continue computation
y = 1
for i = 1 to x
   y = y * i
next i
print "The factorial of ",x," is ",y
```

Tasks

- Given this language definition together with the grammar write an interpreter for this language.
- If variables are used before they were defined assume a value 0 assigned to them.
- Demonstrate that your interpreter works by showing that it correctly interprets the example programs given above.
- HINT: The interpreter for Cuppa1 developed in class might be used as a framework for this μBasic interpreter.

Deliverables

On Sakai: Hand in a notebook that includes both part I and part II of the midterm. For part two show that your interpreter works on the examples given above and if you like on additional examples of your choosing. Failing to submit the midterm as a notebook will result in a failing grade.