Compiler Construction Lab Exam (40 Marks)

Thursday (01Hr 05Mins)

BPDC

Upgrading a mini-compiler

You are being provided with a mini-compiler that parses C programs satisfying a fundamental program template (refer project.y). The program may contain standard declaration statements and arithmetic assignment expressions where the only permitted data types are *int* and *char*. Further, the compiler offers type checking. Modify the program so as to have the following enhancements without loosing the existing functionalities.

1. **Task 1(8M): (Should be finished by the first 20 minutes)** Modify the lex file so as to incorporate the following variable naming conventions.
   (a) The variable names should be composed only of lower case alphabets from your first name and last four digits of your roll-number.
   (b) The variable name always starts with the second or third letter of your first name.
   (c) The variable name should never end in a digit.
2. **Task 2(27M):** Modify the yacc/lex files so as to bring in a set of new keywords $\{is, are, ints, chars\}$ and hence a new user friendly syntactic convention to declaration statements as specified below.
   (a) Single variable declaration statement follows the syntax

   type is $<$ *var_name* $>$; where type can be *int* or *char* (keywords *is*, *int*, *char*). For example,

   ```
   int is x;
   char is test;
   ```

   are valid declaration statements.
   (b) Multiple variable declaration statement follows the syntax

   types are $<$ *var_name* $>$; where type can be *int* or *char*. For example,

   ```
   ints are x,y;
   chars are test1,test2;
   ```

   are valid declaration statements.

   Note that all the following,

   ```
   int is x,y;
   chars are test;
   char are test;
   ```

are invalid declaration statements.

(c) **(Advanced)Task 3(5M):** Bring ins a new statement called alias statement which is of the form:

```
var1 alias var2;
```

which means *var*1 refers to the same memory (and hence the same type) location as *var*2. Correspondingly, we either have to modify the symbol table or bring in a new alias table to deal with. For example, following is a valid segment of code:

```
#include<stdio.h>
int main()
{
   ints are x,y;
   z alias x;
   z=x+y;
}
```

General instructions

1. Save you files on a regular basis.
2. Compile your lex/yacc files as follows:
   lex project.l

   yacc −dv project.y

   gcc lex.yy.c y.tab.c −lfl

   ./a.out < input.c
3. Modify the given sample input files (if required) and test your code.