COMP 506, Spring 2018
Project 1 Questionnaire

Name: _______Yuanqing Zhu_________________________

NetId: ________yz120________________________

1. Briefly discuss your experience building the scanner and parser. What things were easy to do? What things were hard?

   First of all, after reading the pdf, I tried to figure out the outline. I started from the flex and bison tools, read the document and started to code. Just by following the rules and browsed the lecture ppts, I basically constructed the .l and .y file. One problem is that there’re specific orders for the tokens to be scanned, which should be noticed. Then, I debugged and tested, to find the bugs in my codes. At last, I did the error handling, following the instructions in ppt, by testing with the error samples.

   --

   1. Using the flex and bison tools are not hard. The document is clear, and I had experience using these two tools.
   2. Makefile is not hard, I never wrote one before, but it’s clear to create by following the online tutorial.
   3. Creating a driver is not hard, the only problem is to use argc and argv parameters, which was not familiar to me. BTW, comparing whether the first parameter is “-h” takes me much time, due to the comparison string with char* in C is tricky.

   --

   1. The error handling is tricky, we follow the error cases and the syntax patterns, to find where the errors are. Then, add an error token and handle. But we cannot find all the errors, we still need more test cases and lots of effort and time to solve most of the errors.

2. Are there topics that we should cover in class to better prepare you for this project?

   No.

3. What, if anything, would you do differently if you could start over on this project?
   1. I would use a separate file driver.c to include the main function, which will make my code much clear.
   2. I won’t correspondent each token to a specific word in first part of my .l file. It takes me much time and useless. We can just use the original token in the second part.

      Eg. //useless

      ```
      AND "and"
      %%
      ```

```
{AND} {return AND;}


//easy and clear
%%
“and” {return AND;}
```