PRELAB 3

In this prelab, you will implement a lex and yacc code that will read a C code snippet that contains declarations and comparisons. In the input code, there can be any number of if statements and declarations. Your program should cover all the valid inputs, and reject the invalids.

ATTENTION: Zip your lex, yacc and submit to Yulearn. Your code can be compiled with the Makefile provided to you. Be sure that Makefile works properly.

1) **If statement**
   - Contains a comparison part and body part
   - The comparison part can contain 1 comparison.
     - There will be no "&&" and "||" operation like we did in prelab2.
   - Comparison will be only "==".
   - If the body can contain other "if" statements or can contain declarations.
   - Note: There will be no "else if" or "else" In the input file

2) **Declaration statement**
   - There can be 2 types of declarations: "int" and "string".
     - You can assume that there will be only 1 declaration per line.
       - int a;
       - string b;
     - In other words, the input file will not contain declarations like:
       - int a,b,c;
       - string a,b;
3) **Invalid comparison:** The input file can contain invalid comparisons. You should detect and print an error message for that.
   - The comparisons should be between the same types. If it is not print an error message: "Type mismatch in line x"

Example 1	Example 2

int a;
string b;
if (a==b)
{
 	
}
	string b;
	if (b==4)
	{
	
	}

GO TO THE NEXT PAGE

4) **Output:**
    - For each int declaration, you should print:
        - integer variable X is created in line Y
    - For each string declaration, you should print:
        - string variable X is created in line Y
    - For each “If” comparison, you should print:
        - There is a comparison in line 3
    - Look at the example input and output files to understand better.

GRADING POLICY:

IMPORTANT: If you cannot run at least one of the files below correctly, you will not get a grade from the prelab.

IMPORTANT: Those files are an example. The number of the “if” statements, comparisons or depth of the nested ifs can increase or decrease. So design your program carefully.

- **Input 1 (70 pts):** This file contains only valid inputs.
    - If you can write the grammar that accepts the input you will get 30/70 points.
    - If your program can create the correct output you will get (40/70)
    - Your program should produce the same output with exampleOutput1.txt to get a full grade.
- **Input 2 (30 pts)**: This file contains invalid input.
    - Your program should produce the same output with exampleOutput2.txt to get a full grade.

How to test your code:

- Your code should be compiled with the Makefile given to you.
- Do not change the Makefile.
- Run your code and redirect output to a file. Example:
    - ./prelab3 input.txt > output.txt
- Check whether your output and our output file are the same or not by using the diff command
    - diff output.txt exapleOutput.txt -wB
    - if your output and our output are the same there will be no message on the screen
    - Otherwise, the difference will be shown on the screen. Look at the example below.