PROJECT

In this project, you will write a yacc program that will make function inlining. Function inlining is basically to replace a function call by the body of the callee (figure 1).

input	output
void f()	void main()
{	{
int x;	{
x=x*2;	int x;
}	x=x*2;
	}
void main()	}
{	
f();	
}	

Figure 1: Example function inlining.

Input file will contains:

- if and while operations.
- declaration operations.
- assignment operations.
- function
- function call.
- variable types will only integer.

if and while operation:
- The input file will contain if and while statements just like the prelab 2 and 3. You can use your grammar or the one I have uploaded to the coadsys.

declaration operations:
- The input file will contain declaration statements.
- Each declaration statement can contain multiple variables.
- All declarations will be integer.
- There will be no assignment/initialization in the declaration statement.
    - int a,b,c;
    - ~~int a=5,b;~~

assignment operations:
- The input file will contain assignment statements.
- Assignment statements can contain, multiplication, subtraction, summation and division operations.
    - a = a*2+5/y-x;

functions:

- The input file will contain multiple functions.
- A function has a return type, function name, and body.
    - return type will be always void.
    - function name contains alphanumeric characters.
    - function will never have an input parameter (except the bonus part)
    - a function body can contain if, while, declaration and assignment operations.
    - Since the function is void, there will be no return statement.
- There will be no recursive function.
- A function takes place in the file before the function call. Just like the c language function mechanism.

function call:

- A function is called in another function.
- The function call is carried out just like in the c programming language.
- All functions will take place before the function call.
- A function can be called from different functions multiple times.
- You should check if the called function exists or not. If not, print an error message.

OUTPUT:

- Your program will take every function and open it in the place where it is called. The function will be opened by copying everything in the function between { and } symbols.
- In the end there will be only one function which is main.
- if the called function does not exist, print an error message.
- The indentation in the output is not important.
- The number of tab, newline and space is not important.
- Try the example input and output files provided to you. Your output file should be exactly the same with the one provided. Use the “diff -wB” command to test it.

BONUS (+20 Points):

- In the bonus part, you should consider that a function can have input parameters. In such cases, inlining will require extra operation.
    - The values passed to the function should be assigned to the variables.
    - The variables should have the same name and order with the function parameters.
- There can be an infinite number of input parameters.

input	output
void f(int x)	void main()
{	{
x=x*2;	{
}	int x = 3;
void main()	x=x*2;
{	}
f(3);	}
}	

Figure 2: Bonus Example function inlining.