CS3300: Compiler Design **Odd Sem**

Lab #13

Optimization Deadline: 04/11/2018, 11:55PM

Important Instructions

- Program should read from stdin and write to stdout.
- Follow submission guidelines carefully.
- Violating any of the above instructions will incur penalty.

Function Inlining

Function Inlining is an optimization wherein the functions are inlined by expanding the body of the function instead of calling the function. This reduces the overhead of calling and returning from a function and also exposes additional opportunities for further optimizations.

Task

The aim of the assignment is to output a function inlined C code for user given C code as input using lex/yacc. The following are the specifications:

- The only data type supported is **int**.
- The statements include local variable declarations, assignment statements with basic expressions, return statements, user defined function calls and printf statements. Examples of statements with basic expressions are as follows:
  - a = b, a = -b (b could be variable or non-negative integer)
  - a = b + c, a = b - c, a = b * c, a = b / c (b,c could be variable or non-negative integers)
- printf statements will contain at most one format specifier (as below).
  - printf("Hello %d World ", a);
  - printf("Hello World");
- Only the main function will contain function calls. Hence, there will be only one level of function inlining.
- The main function will not contain any parameters and the return type will be void.
- Every user defined function will contain one parameter of type int and one return value of type int.
- User defined functions may contain printf statement(s).
- The main function won't call itself.
- Variable/function names can contain underscore.
- Only one header - <stdio.h> will be present.

Input

A valid C program with a subset of statements as mentioned above.

Output

A valid, function inlined C code which contains only the optimized **main** function. All other user defined functions should not be present.

Evaluation

The output code will be tested as follows:

- The output of your optimized code will be checked for program correctness against the given input program.
- The output code will be checked for presence of any user defined functions/function calls.

Examples

Input:

```
#include <stdio.h>
int addByFive(int a){
    int num;
    int five;
    five = 5;
    num = a + five;
    return num;
}

int mulBy2(int b){
    int num;
    num = b * 2;
    return num;
}

void main(){
    int num;
    num = 3;
    num = addByFive(num);
    num = mulBy2(num);
    printf("num value: %d\n",num);
}
```

Output:

```
#include <stdio.h>
 void main(){
    int num;
    num = 3;
    int add;
    add = num;
    int i;
    int five;
    five = 5;
    i = add + five;
    num = i;
    int mul;
    mul = num;
    int j;
    j = mul * 2;
    num = j;
    printf("num value: %d\n",num);
}
```

Note: i and j are arbitary variable names. You may generate any non-conflicting temporary names.

Submission Guidelines

Submit a tar.gz file with filename as <ROLLNO>.tar.gz (eg. CS12B043.tar.gz) containing the following structure:

- CS12B043/
  - *.l
  - *.y
  - Makefile

The Makefile should generate an executable a.out.