Assignment-7

Due : 23^{rd} April, 2021

In the assignment, you have to generate intermediate code corresponding to a toy language code. There are two tasks. The first task is to change the toy language grammar. In the modified grammar, all the occurrence of { will be replaced by the string "beg" and all the occurrence of } will be replaced by the string "end". Hence, the following code snippet will be accepted by the toy compiler.

```
int main()
    beg
        float a, b, c;
        a = b + c;
        if(a>b)
        beg
            a = b + c;
        end
        else
        beg
            a = b - c;
        end
        while(a<b)
        beg
            a = a+c;
        end
    end
```

The second task is to generate intermediate code. For the present assignment, you may assume that $if - else$ and $while$ statements are not present in the input code. The intermediate code is to be written in LLVM IR language. A sample input and output codes for the intermediate code generator is given below.

Input (input) :

```
int main()
beg
    int a, b;
    a = 6;
```

```
    b = a * 7;
end
```

Output (out.ll) :

```
define i32 @main() #0
{
    %1 = alloca i32, align 4
    %2 = alloca i32, align 4
    store i32 6, i32* %1, align 4
    %3 = load i32, i32* %1, align 4
    %4 = mul nsw i32 %3, 7
    store i32 %4, i32* %2, align 4
    ret i32 0
}
```

Note that the sample output does not have header and footer for a typical llvm IR (.ll) file. It is expected that the generated file out.ll can be subjected to the llvm interpreter *lli* without any error. More precisely, the command
lli out.ll
will not output any error. You may allow types integer and double and operators '+' and '*' only for generating intermediate code.