Assignment-9

Due : 5^{th} April, 2023

In the assignment, you have to generate intermediate code corresponding to a conditional expression. For the present assignment, you may assume that only $if - else$ statements are 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()
{
    int a, b;
    a = 6;
    b = 7;
    if(a > b || b > a)
    {
        a = a + 1;
    }
    else
    {
        a = a - 1;
    }
    a = a + 5;
}
```

Output (out.ll) :

```
define i32 @main() #0 {
%1 = alloca i32, align 4
%2 = alloca i32, align 4
%3 = alloca i32, align 4
store i32 0, i32* %1, align 4
store i32 6, i32* %2, align 4
store i32 7, i32* %3, align 4
%4 = load i32, i32* %2, align 4
%5 = load i32, i32* %3, align 4
%6 = icmp sgt i32 %4, %5
br i1 %6, label %11, label %7
```

```
; <label>:7:                                              ; preds = %0
  %8 = load i32, i32* %3, align 4
  %9 = load i32, i32* %2, align 4
  %10 = icmp sgt i32 %8, %9
  br i1 %10, label %11, label %14

; <label>:11:                                             ; preds = %7, %0
  %12 = load i32, i32* %2, align 4
  %13 = add nsw i32 %12, 1
  store i32 %13, i32* %2, align 4
  br label %17

; <label>:14:                                             ; preds = %7
  %15 = load i32, i32* %2, align 4
  %16 = sub nsw i32 %15, 1
  store i32 %16, i32* %2, align 4
  br label %17

; <label>:17:                                             ; preds = %14, %11
  %18 = load i32, i32* %2, align 4
  %19 = add nsw i32 %18, 5
  store i32 %19, i32* %2, align 4
  %20 = load i32, i32* %1, align 4
  ret i32 %20
}
```

Note that the sample output does not have header and footer for a typical llvm IR (.ll) file. The intermediate code has to be generated using backpatching method. As this method requires to insert labels to the branch statements, the entire file content has to be written to a file buffer. The buffer will be written to the file at the end. 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 the relational operator '>' only for generating intermediate code.