1. You are given a stack machine with the following instruction set:

- `push value` Pushes the `value` to the stack
- `pop` Pops top of the stack
- `load loc` Pushes value of the data location `num` to the stack
- `store loc` Pops a value from the stack and stores it in data location `loc`
- `ifeq label` Pops a value from the stack, if it is equal to zero jumps to statement `label`
- `ifneq label` Pops a value from the stack, if it is not equal to zero jumps to statement `label`
- `goto label` Jumps to statement `label`
- `add` Pops two values from the stack, adds them, and pushes the result back onto the stack
- `cmplt label` Pops the top two values off the stack, if the value popped last is less than the value popped first then it jumps to statement `label`
- `cmlte label` Pops the top two values off the stack, if the value popped last is less than or equal to the value popped first then it jumps to statement `label`

The stack machine instructions can be labeled as “Label : instruction”.

(a) Write the semantic rule for generating stack machine code for if-then-else statements defined by the following production:

```
IfStmt → if ( Expr ) then Stmt else Stmt
```

where `IfStmt`, `Expr` and `Stmt` are nonterminal symbols and `if`, `(`, `)`, `then` and `else` are terminal symbols. Assume that `Expr` and `Stmt` nonterminals have `code` attributes which generate code for them. The code generated by boolean expressions store a value in the top of the stack which is 1 if the expression is true, 0 if it is false. You can use operator `||` to concatenate generated code, function `gen` to generate an instruction or a label (you have to call it with appropriate arguments) and `newlabel` to get a new label.

(b) Write the stack machine code (based on the semantic rules you gave in part (a)) for the following program segment assuming that variable `x` is stored in data location 8, and variable `y` is stored in data location 12.

```java
if (x <= y) then
  x = y;
else
  x = x + 2;
```

(c) Write the semantic rule for generating stack machine code for while statements defined by the following production:

```
WhileStmt → while ( Expr ) do Stmt
```

Assume that `Expr` and `Stmt` nonterminals have `code` attributes which generate code for them. The code generated by boolean expressions store a value in the top of the stack which is 1 if the expression is true, 0 if it is false.
(d) Write the stack machine code (based on the semantic rule you gave above) for the following program segment assuming that variable $x$ is stored in data location 10, and variable $y$ is stored in data location 14. Here you must also write the $Expr.code$ and $Stmt.code$.

```plaintext
while (x < y) do {
    x = x + 3;
}
```

2. Consider the three-address code with the following instructions:

- $x := y$ Assignment statement
- $x := y \ op \ z$ Assignment statement where $\ op$ is a binary arithmetic or logical operation
- $x := \ op \ y$ Assignment statement where $\ op$ is a unary arithmetic or logical operation
- goto $label$ Jumps to $label$
- if $x \ relop \ y$ goto $label$ If the condition holds ($\ relop$ is a relational operator) it jumps to $label$

The instructions can be labeled as “Label : instruction”. You can use operator $||$ to concatenate the generated code, function $gen$ to generate an instruction or a label and $newlabel$ to get a new label.

(a) Assuming implicit representation of boolean expressions, write the semantic rule for generating three-address code for if-then-else statements defined by the following production:

```plaintext
Stmt \rightarrow \ if \ Expr \ then \ Stmt \ else \ Stmt
```

Assume that $Expr$ and $Stmt$ nonterminals have $code$ attributes which store the generated code. Also assume that $Expr$ nonterminal has a $true$ attribute denoting the label that the generated code should jump to if the expression evaluates to true, and a $false$ attribute denoting the label that the generated code should jump to if the expression evaluates to false. Finally, assume that the $Stmt$ nonterminal has a $next$ attribute denoting the label of the statement that should be executed immediately after $Stmt$.

(b) Write the generated three-address code for the following expression (based on the semantic rule you gave above). Here you should also show the code generated for $Expr$ and $Stmts$.

```plaintext
if (a < b + 4) then a = b + 3 + d; else c = d;
```