Sample Problems for CS 152 Final

Q1. Below is the specification of a control flow construct and its grammar:

Example:

```plaintext
If x < y then
    <otherstatements>
elsif a > b then
    <otherstatements>
else
    <otherstatements>
endif
```

Relevant Grammar Productions:

```
<S>  →  if <condt> then <otherstatements> <rest>
<rest>  →  elseif <condt> then <otherstatements> <rest>
        |  else <otherstatements> endif
<condt>  →  id relop id
```

Provide the semantic rules that generate code for the above control flow construct. The generated code should be available in the `<S>.code` attribute. Assume that semantic rules specified by other productions not shown here will place the code generated for `<otherstatements>` in the attribute `<otherstatements>.code`.

Solution:

```plaintext
<condt>  →  id1 relop id2
    { 
        truelabel = newlabel();
        <condt>.falselabel = newlabel();
        <condt>.code = gen("if" id1.place "relop" id2.place "go to" truelabel)
        || gen("go to" <condt>.falselabel) || gen(truelabel":")
    }

<S>  →  if <condt> then <otherstatements>
    { 
        <rest>.iflabel = <condt>.falselabel;
        <rest>.exit = newlabel();
        <rest>.code = <condt>.code || <otherstatements>.code ||
        gen("go to" <rest>.exit)
        gen("go to" <rest>.iflabel)
    }
<rest> { <S>.code = <rest>.scode }
```
Q2. Construct the control flow graph for the following intermediate code sequence.

1. Read X
2. Read Y
3. $X = X + Y$
4. If $Z < 0$ go to 7
5. $X = X + 1$
6. Go to 8
7. $X = X - 1$
8. $Y = Y + 1$
9. $T = X + Y$
10. If $Z = T$ go to 4
11. Print $Z$

Q3. For the following code sequence: construct the control flow graph; perform liveness analysis for all variables; construct live ranges; and construct the interference graph. Color the interference graph using (a) 2 colors; and (b) 3 colors. Assume all variables are live at the end of the code sequence.

Read X
Read Y
If $X < Y$ go to L1
$Z = X + Y$
$X = Y$
Go to L2
L1:
$Z = X - Y$
$X = Y$
L2:
Print X
Print Z
Q4. For the program given below show the contents of the runtime stack (activation records, local variables, control links, and access links) for the following call sequence: Main → F → G → H → F

Main () {
    Int a, b;

    F() {
        Int a, c;
        Call G();
    }

    G() {
        Int a, e;
        H() {
            Int a, d;
            Call F();
        }
    }
    Call H();
}
Call F()

(a) Show how the access links are computed and setup; and
(b) Show how various non-local variables are accessed within each function using the access links.

8. access variable b from H.

3 - 1 = 2  (2, offset)