CS 152: Exam-II (June 8, 2021) **Name:_____**

/24

1. (4 x 6 = 24 points)

i. The storage for *static* variables declared in a *c* function
   a) cannot be allocated along with global variables
   b) cannot be allocated along with local variables in the stack frame
   c) cannot be allocated in a heap block pointed to by a global pointer

ii. In a programming language without recursion the activation frames
   a) must be statically allocated once at the start of execution
   b) must be allocated in the runtime stack
   c) must be allocated in the runtime heap
   d) none of the above

iii. Backpatching is applied when instruction number of the branch target
   a) is less than the instruction number of the branch itself
   b) is greater than the instruction number of the branch itself
   c) backpatching is always necessary
   d) backpatching is never necessary

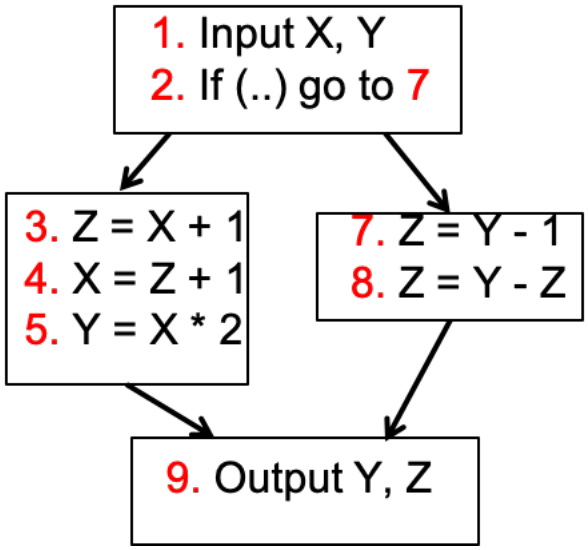
iv. In the production A → B A D, a semantic rule for evaluating a synthesized/inherited attribute of non-terminal A must be placed
   a) at the end/beginning of the right-hand-side of the production
   b) immediately following/somewhere preceding A on the right-hand-side
   c) immediately following/preceding A on the right-hand-side
   d) somewhere following/immediately preceding A on the right-hand-side

v. An intermediate code instruction *i* is a leader of a basic block if:
   a) *i* is the first instruction in a function
   b) *i* is the target of a branch instruction
   c) *i* immediately follows a branch instruction
   d) all of the above are true

vi. None of the program variables are live
   a) at the start of a function
   b) at the end of a function
   c) at both the start and end of a function
   d) none of the above is true

/30

2. **(3 x 10 =** 30 points) Live Ranges & Interference Graphs. Given the following control flow graph:



i. How many live ranges are there for variable X?

ii. How many live ranges are there for variable Y?

iii. How many live ranges are there for variable Z?

iv. Do all live ranges of X interfere with a live range of Y?

v. Do all live ranges of X interfere with a live range of Z?

vi. Does a live range of Y interfere with a live range of Z?

vii. What is the highest degree value in the interference graph?

viii. What is the lowest degree value in the interference graph?

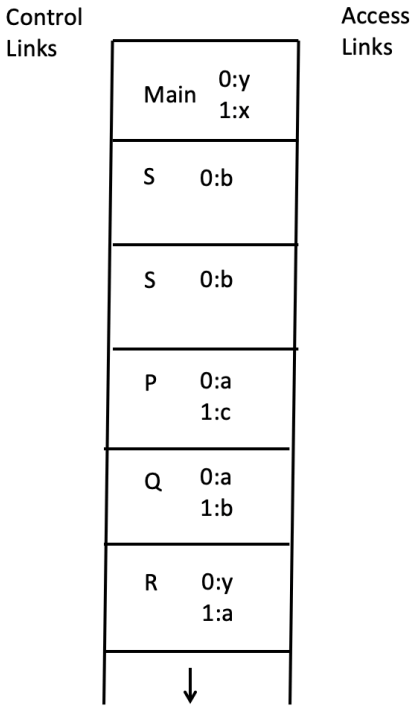
ix. What is the minimum number of colors needed to color all the live ranges in the interference graph?

x. If only one color is available to color the interference graph, what is the minimum number of live ranges that will not be assigned a color?

/26

3. (2x13 points) Runtime Management: Given call sequence Main → S → S → P → Q → R

```
Main () {
   integer y, x;
   Procedure P () {
      integer a, c;
      Procedure Q () {
         integer a, b;
         Call R ();
      }
      Procedure R () {
         integer y, a;
         Call S ();
      }
      Call Q ();
   }
   Procedure S () {
      integer b;
      if (..) Call S ()
      else Call P() endif
   }
   Call S ();
}
```



In the runtime stack shown:

i. Control link of R points to _____
ii. Control link of P points to _____
iii. Access link of R points to _____
iv. Access link of Q points to _____
v. Access link of P points to _____
vi. Access link of the first frame of S points to _____
vii. Access link of the second frame of S points to _____
viii. Accessing c from R requires traversal of how many access links?
ix. Accessing x from R requires traversal of how many access links?
x. Accessing y from R requires traversal of how many access links?
xi. Setting up access link of R requires traversal of how many access links?
xii. Setting up access link of Q requires traversal of how many access links?
xiii. Setting up access link of P requires traversal of how many access links?

/20

4. (2x10 points) Intermediate Code Generation: The grammar below corresponds to a control construct with the following semantics. When the value of variable corresponding to **id** is less than zero, equal to zero, or greater than zero, the code in the **<less>**, **<equal>**, or **<greater>** part is executed.

<S> → switch id do <less> <equal> <greater> endswitch { "Compute <S>.code" }
<less> → less <S> endless { <less>.code = <S>.code }
<equal> → equal <S> endequal { <equal>.code = <S>.code }
<greater> → greater <S> endgreater { <greater>.code = <S>.code }

(a) Provide the intermediate code that implements the following use of the construct:

```
switch a do
      less     print -a   endless
      equal    print  0   endequal
      greater  print  a   endgreater
endswitch
```

(b) The semantic rules for all but the first production are provided. Provide the missing code "Compute <S>.code" in the first production that generates the code for the entire construct and places it in <S>.code.