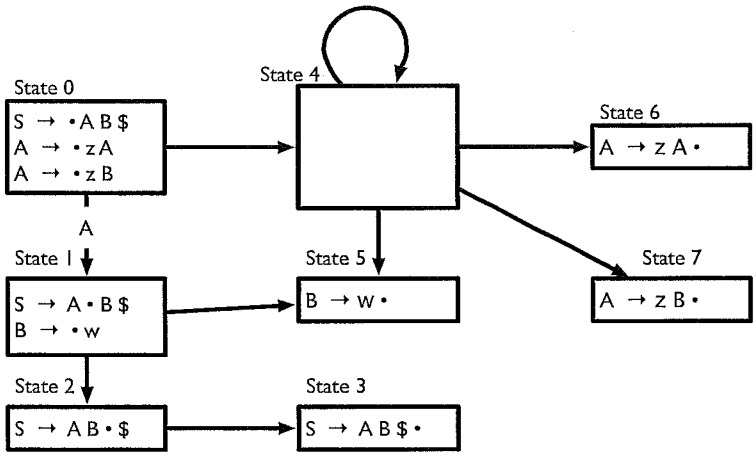
1. **(5 points)** Consider the language $L_1$ defined by the expression $a^n b^{m-n} c^m$, $m \geq 0$, $m \geq n$. (i.e., any string with $m$ c's, $n$ a's, and as many b's as the difference between the two). Can a finite state machine recognize $L_1$? Give a one-sentence argument for why or why not.
2. Consider the following grammar for the following questions:

$$\begin{array}{|l} \hline S \rightarrow AxAy\$ \\ A \rightarrow zA \\ A \rightarrow \lambda \\ \hline \end{array}$$

   a. **(5 points)** What are the terminals and non-terminals of this grammar?
   b. **(5 points)** Draw the *first state only* of the LR(1) machine used to parse this grammar.
   c. **(5 points)** Could this grammar be parsed by an LR(0) parser? Why or why not?
3. For the next questions, consider the following grammar and associated partial CFSM:

$$\begin{array}{|l} \hline S \rightarrow AB\$ \\ A \rightarrow zA \\ A \rightarrow zB \\ B \rightarrow w \\ \hline \end{array}$$




   a. **(5 points)** In your blue book, show the configuration set that would be in State 4. Also, show what symbol would be on the self-edge from State 4 to itself.
   b. **(5 points)** Suppose the state stack is 0 4 4 5, and the next symbol in the input is a w. What action will the parser take? If it is a shift action, say what state the parser will shift to. If it is a reduce action, say which rule is recognized, and what state the parser will end up in. Give the state stack after the action is taken.

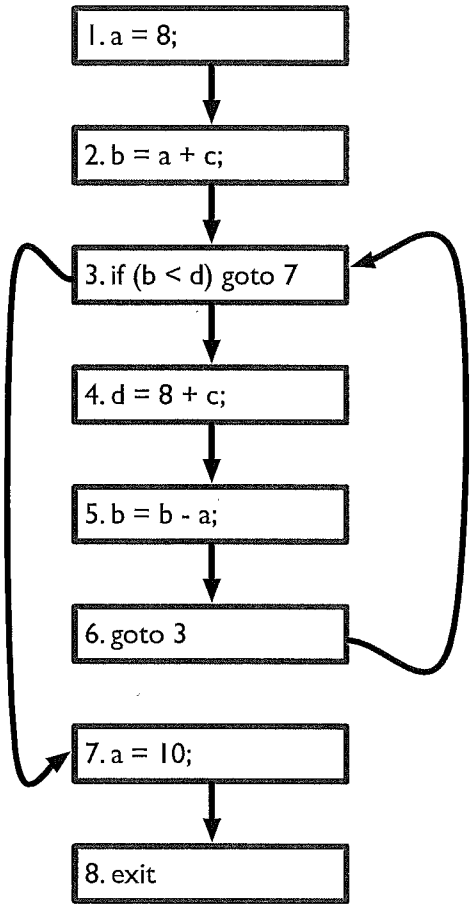
4. This question pertains to *reaching-defs* analysis. For each statement below, we want to calculate which definitions *reach* a statement. Assume this is the entire code.

a. **(16 points)** ***In your blue books,*** fill in a table like the one below, with one row for each statement in the program. ***Do not fill in the table below.*** Assume there is no aliasing.

b. **(4 points)** Show the GEN and KILL sets for statement 5 if variable b *may* be aliased to variable d.

c. **(10 points)** Explain (in one or two sentences) how you would use the results of this analysis to find dependences in a program. Your explanation should be 30 words or fewer. Give an example of this by listing which statement(s) statement 5 depends on and explaining why.

| Statement | GEN | KILL | IN | OUT |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| ... | | | | |




Write in Exam Book Only

5. Given the loop nest:

```
for (k = 0; k < n; k++) {
  for (i = 0; i < n; i++) {
    a[i, k] = b[i + 1, k + 3];
    b[i + 3, k] = a[i + 2, k + 1];
  }
}
```

   a. **(5 points)** Describe the dependences, if any, on the "**a**" array in the loop nest above. Either say "no dependence", or, if one or more dependences exists, give the type(s) (flow or true, output or anti), the direction and the distance.
   b. **(5 points)** Describe the dependences, if any, on the "**b**" array in the loop nest above. Either say "no dependence", or, if one or more dependences exists, give the type(s) (flow or true, output or anti), the direction and the distance.
   c. **(5 points)** Can the loops be interchanged? Why or why not?
   **d. (5 points)** Can loop distribution be applied to the inner loop? Why or why not?

6. Acme's first generation computers only had one ALU. Their latest computers have two ALUs. Are the following optimizations more or less important now?

   a. **(5 points)** Loop unrolling
   b. **(5 points)** Register allocation
   c. **(5 points)** Common subexpression elimination
   d. **(5 points)** Instruction scheduling

7. **(10 points)** For the following code, show what the code would look like after performing strength reduction *and* linear test replacement.

```
1:  X = 10;
2:  Y = A;
3:  if (A < 2 * X + 10) goto 9
4:    R = 4 * X + 7;
5:    X = X - A;
6:    S = A * Y + 10;
7:    Y = Y + 5;
8:    goto 3
9:  exit;
```