Q1. Using the Characteristic Finite State Machine (CFSM) above, answer the following questions in your test book. 35 points total.

(a) (2 points) What is the grammar the CFSM is for (there is sufficient information without the missing states to answer this question)? Give the productions.

(b) (1 points) What are the terminal symbols of the grammar?

(c) (1 points) What are the non-terminal symbols of the grammar?

(d) (4 points) What are the contents of state 2?

(e) (4 points) What are the contents of State 5?

(f) (4 points) What are the contents of State 7?

(g) (2 points) What is the edge from State 1 to State 2 labeled with?

(h) (2 points) What is the edge from State 4 to State 5 labeled with?

(i) (2 points) What are the edges into State 7 labeled with?

(j) (2 points) In State 13, what state will be entered when a “(“ is the next input token?

(k) (5 points) Given a symbol parser stack of “( id”, a state stack of “0 8 9” and a next symbol of “+”,
what is the action the parser takes and what are the contents of the symbol and state stack after the action is taken? Note: “id” is the last symbol pushed onto symbol stack, and “9” is the last state pushed onto the state stack.)

(l) (5 points) Given a symbol stack of “( id + id ) *”, a state stack of “0 8 9 10 11 12 13” and a next symbol of “1”, what is the action the parser takes and what are the contents of the symbol and state stack after the action is taken? Note: “*” is the last symbol pushed onto symbol stack, and “13” is the last state pushed onto the state stack.)

(m) (1 points) What is the “accept” state of the CFSM?
Q2 Using the program above (Fig. 1) answer the following questions in your test book. 36 points total.

(a) (6 points) Is there a dependence on the write of array a in line 1 and the read of array a in line 2? If so, give the type of the dependence (flow or true, output or anti), the direction vector and the distance vector for the dependence.

(b) (6 points) Show the loops after performing loop interchange.

(c) (6 points) The programmer notices that interchanging the i and j loops causes the program to run significantly faster. Answer that which is most true: (1) this is consistent with the arrays laid out in column major order, (2) this is consistent with the arrays being laid out in row major order, (3) the effects of loop interchange are not dependent on whether an array is laid out in column major or row major order.

(d) (6 points) Is the loop interchange a legal transformation, i.e., will the program give the same result before and after the transformation?

(e) (6 points) Is it legal to change the code above to the code below (Fig. 2), where the changes are shown in bold. You can assume that the sin function computes the standard trigonometric function.

\[
\begin{align*}
t &= \sin(y);
\end{align*}
\]

\[
\begin{align*}
\text{for (int } i = 0; i < n; i++) \{} \\
\text{ for (int } j = 0; j < n; j++) \{} \\
\quad a[j][i] &= \sin(float(i)); \quad \text{// line 1} \\
\quad \ldots &= a[j-1][i] \quad \text{// line 2} \\
\text{ \} } \\
\text{ \} }
\]

Fig. 2

(f) (6 points) The code of Fig. 1 at the top of the page is changed the code of Fig. 3. Is the transformation from the code of Fig. 3 to the code of Fig. 4 legal, i.e., will the code of Fig. 4 give the same answer as the code of Fig. 3?

\[
\begin{align*}
t &= \sin(float(i)) \\
\text{for (int } i = 0; i < n; i++) \{} \\
\text{ for (int } j = 0; j < n; j++) \{} \\
\quad a[j][i] &= t; \quad \text{// line 1} \\
\quad \ldots &= a[j-1][i] \quad \text{// line 2} \\
\text{ \} } \\
\text{ \} }
\]

Fig. 3

Fig. 4

Write in Exam Book Only
A compiler writer decides to add a conservative points-to analysis dataflow pass to a compiler. For each pointer \( r \), the analysis finds a set \( P_r \) of variables \( v \) that \( r \) might point-to. To keep things simple, assume that this analysis only targets programs where pointers can point to integers, but not other pointers. The analysis works as follows by updating each \( P \) at every statement according to the three rules:

(i) If a pointer \( q \) is assigned the address of \( v \) then \( P_q = \{ v \} \).
(ii) If a pointer \( q \) is assigned the value of a pointer \( r \) (i.e., \( q = r \)) then \( P_q = P_r \)
(iii) At a join point, \( P \) for each variable is the union of the \( P \) sets that flow into the join from each edge.

Q3 Answer the questions below. This questions is worth 29 points total.

(a) (7 points) Is the analysis a forward or backward analysis?
(b) (7 points) For a variable \( v \) to be in \( P_r \) at statement \( s \), is it necessary for the \( r \) to point to \( v \) along all paths from the beginning of the program to \( s \)? Stated differently, is this an ALL-paths or ANY-path algorithm?
(c) (5 points) For many dataflow algorithms, the GEN and KILL sets for a block can be computed using only information about reads and writes in a statement or basic block. We will call these constant dataflow analyses. For others, the GEN and KILL sets are a function of the reads and writes in the statement or basic block, and the current value of the dataflow information being computed. We will call these non-constant dataflow analyses. Is the points-to-analysis of this question a constant or non-constant dataflow analysis?
(d) (10 points total, 1/2 point for each box filled in.) Given the program and control flow graph on the next page, fill in a table in your test books, like the one on the next page, with the

```
L1: int i, j, k;
L2: int* q = &i;
L3: int* r = &j;
L4: int* w = &k;
L5: if (f( )) {
L6: q = &j;
L7: r = &k;
L8: }

<table>
<thead>
<tr>
<th>Statement</th>
<th>( P_q )</th>
<th>( P_r )</th>
<th>( P_w )</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>{ }</td>
<td>{ }</td>
<td>{ }</td>
</tr>
<tr>
<td>L2</td>
<td>{ &amp;i }</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>L8</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
values of $P_r$, $P_q$, and $P_w$ at each statement. The L1: row and the entry for $P_q$ in the L2: row have already been filled in.