This exam has eight (8) questions.

This exam is closed book. No notes, books, calculators, or other assistance may be used.

Write all your answers directly in the exam.

Write your full name and student I.D. number in the spots provided in the exam.

State any assumptions you make.

Show rough work.

Name: ________________________________
FOR MARKING USE ONLY:

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>/58</td>
</tr>
</tbody>
</table>
Question 1 (4/58)

Define “backpatching.”

Question 2 (2/58)

For the declarations below:

```pascal
type
  foo: integer;
  bar: integer;
var
  i, j: foo;
  k: integer;
  n: bar;
```

A. Which variables are structurally equivalent?

B. Which variables are name equivalent?

Question 3 (4/58)

Define “constant folding” and give a short example using pseudocode.
Question 4 (8/58)

An LALR(1) parser generator reports a conflict when given the grammar below. (As in lectures, $\lambda$ represents the empty string.) Why?

$$\begin{align*}
S & \rightarrow \ A \ B \ c \\
A & \rightarrow \ \lambda \ | \ a \\
B & \rightarrow \ \lambda \ | \ a
\end{align*}$$

Question 5 (8/58)

Draw an AST for the following code. Make sure your AST nodes are labeled so that the correspondence to the code is evident.

```c
do {
    x = recalc(x, 2 + 3 * 5);
} while (x < 5);
```
Question 6 (4/58)

Transform the grammar below to remove immediate left recursion.

\[
\begin{align*}
E & \rightarrow \text{EEg} \\
E & \rightarrow \text{a} \\
E & \rightarrow \text{b}
\end{align*}
\]

Question 7 (8/58)

Consider a variant of the usual `while` loop which has an `else` part. The code in the `else` part is executed only if the `while` loop exits normally, i.e., not with a `break` statement.

Show a translation of the following code into pseudo-assembly code:

```
S1
while (expr) {
    S2
    break
    S3
} else {
    S4
}
S5
```
Question 8 (20/58)

Construct the LALR(1) parse table for the grammar below. Show the alternate grammar and FOLLOW sets as well.

```
0  S'  \to  S
1  S    \to  A B C
2  S    \to  A C
3  A    \to  a C
4  B    \to  b
5  C    \to  b
```

Use the diagram on the opposite page for your state machine, and be sure to use the state numbers from the diagram when building your table.
Remember to fill in the diagram fully, including labeling the edges.