This exam has six (6) questions.

You may use one single-sided 8.5x11” piece of paper with whatever you want written on it. Apart from that, this exam is closed book. No notes, books, calculators or electronic devices, 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 spot provided below on the exam.

State any assumptions you make.

Show all work.

Name: 

Student I.D.: 
Question 1 (4/38)

The general form of immediate left recursion is

\[ A \to A_1 A_2 \cdots A_m \beta_1 \beta_2 \cdots \beta_n \]

Show how to transform this to remove the immediate left recursion.

Question 2 (6/38)

The Python Language Reference\(^1\) defines the lexical structure of octal (base 8) literals using the following EBNF-like definition:

```
octinteger ::= "0" ("o" | "O") (["_"] octdigit)+
octdigit ::= "0"..."7"
```

They provide two examples of octal integer literals: `0o177` and `0o377`. In addition, they add that ‘Underscores are ignored for determining the numeric value of the literal. They can be used to group digits for enhanced readability. One underscore can occur between digits, and after base specifiers like 0x.’

Using any method, draw a deterministic finite automaton (DFA) that accepts octal literals as defined above.

\(^1\)Version 3.8.2, Section 2.4.5.
**Question 3 (8/38)**

Construct the LL(1) parse table for the following grammar. As in lectures, $\lambda$ denotes the empty string. Remember to show your work.

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

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Question 4 (12/38)

Construct the SLR(1) parse table for the grammar below, which has already been augmented with the start symbol S.

\[
\begin{align*}
0 & \quad S \rightarrow T \\
1 & \quad T \rightarrow T F \\
2 & \quad T \rightarrow F \\
3 & \quad F \rightarrow F * \\
4 & \quad F \rightarrow a
\end{align*}
\]

Use the diagram on the opposite page for your state machine, and be sure to use the state numbers from the diagram and the rule numbers from the grammar when building your table.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>*</th>
<th>$</th>
<th>S</th>
<th>T</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>$</td>
<td>S</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Question 5 (4/38)

For the grammar in Question 4, could an LR(0) table be built without conflicts? Explain your answer and, if there would be conflicts, clearly identify what/where they would be.
This diagram is for use with Question 4. There are no marks for filling this diagram out, but this gives you a way to check part of your answer, and gives me a way of marking this question and getting you your grade relatively quickly.

Question 6 (4/38)

You succeed in building an LL(1) parse table without any table entries being multiply-defined. What does this tell you about the grammar? Circle all that apply.

- The grammar is unambiguous.
- The grammar is LL(1).
- The grammar is LL(2).
- The grammar has no left recursion.
- The grammar is LR(0).