Question 1

(a) If n is the number of characters in the alphabet, then no string can be longer than n repeated characters without repeating. The set of all strings no longer than n is finite, and hence regular.

(b) This is very similar to the C5X multi-character comment:

\( <(>\neg(>)^*) > \)
Question 2

Two ways to show $S$ is not regular

(i) In an exam question we established that if $S_1$ and $S_2$ are regular, so is

$S_1 \cap S_2$ (since $S_1 \cap S_2 = S_1 \cap \overline{S_2}$)

But ($[+]^+ - S$) is balanced brackets, not regular

(ii) Same idea as balanced brackets:

$[[]][][],...[[]]^+$ must all now be rejected.

Read $[],[[],[],[]], etc$ until two distinct prefixes,

$[[]]^+$ and $[[[]]^+$ both reach the same state

$[[[]]^+$ must reach a non-accepting state

But $[[[]]^+$ will reach the same state and

it should be accepting (since $[+]^+$).

A contradiction!

Question 3

(a) There are many possible answers. Here is one:

```
"\\n"
```

(b) 

```
\\\\
```

(c) We'll do the comment in three segments.

Oneline = "{" [^}\n]* "}"

Twolines = "{" [^}\n]* \n [^}\n]* "}"

Threelines = 

"{" [^}\n]* \n [^}\n]* \n [^}\n]* "}"

Answer = {Oneline} | {Twolines} | {Threelines}
4. Below is a context-free grammar for a language of assignments that includes arrays:

1. stmtList → stmt stmtList
2. | λ
3. stmt → ID = exp ;
4. array → [ rowList ]
5. rowList → nonEmpty
6. | λ
7. nonEmpty → row moreRows
8. moreRows → ; nonEmpty
9. | λ
10. row → exp more
11. more → , row
12. | λ
13. exp → term tail
14. tail → + term tail
15. | λ
16. term → ID
17. | INTLIT
18. | array

Here are the FIRST and FOLLOW sets for all of the non-terminals:

<table>
<thead>
<tr>
<th>Non-terminal X</th>
<th>FIRST(X)</th>
<th>FOLLOW(X)</th>
</tr>
</thead>
<tbody>
<tr>
<td>stmtList</td>
<td>ID</td>
<td>EOF</td>
</tr>
<tr>
<td>stmt</td>
<td>ID</td>
<td>ID EOF</td>
</tr>
<tr>
<td>array</td>
<td>[</td>
<td>+ ; ]</td>
</tr>
<tr>
<td>rowList</td>
<td>ID INTLIT</td>
<td>[</td>
</tr>
<tr>
<td>nonEmpty</td>
<td>ID INTLIT</td>
<td>[</td>
</tr>
<tr>
<td>moreRows</td>
<td>;</td>
<td>]</td>
</tr>
<tr>
<td>row</td>
<td>ID INTLIT</td>
<td>; ]</td>
</tr>
<tr>
<td>more</td>
<td>,</td>
<td>; ]</td>
</tr>
<tr>
<td>exp</td>
<td>ID INTLIT</td>
<td>+ ; ]</td>
</tr>
<tr>
<td>tail</td>
<td>+</td>
<td>, ; ]</td>
</tr>
<tr>
<td>term</td>
<td>ID INTLIT</td>
<td>+ , ; ]</td>
</tr>
</tbody>
</table>

(a) Recall that terminal \( t \) is in FOLLOW(\( X \)) if in some partial parse tree with the start non-terminal at the root, \( X \) is one leaf of the tree and \( t \) is the next non-lambda leaf immediately to the right. For example, the following partial parse tree justifies the fact that for the CFG given above, terminal ID is in FOLLOW(stmt):
Complete the partial parse tree below to justify the fact that terminal ; is in FOLLOW(term).
(b) Fill in the parse table below using the numbers of the grammar rules rather than the rules themselves. Is the grammar LL(1)?

<table>
<thead>
<tr>
<th>ID</th>
<th>INTLIT</th>
<th>=</th>
<th>+</th>
<th>;</th>
<th>,</th>
<th>[</th>
<th>]</th>
<th>EOF</th>
</tr>
</thead>
<tbody>
<tr>
<td>stmtList</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>stmt</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>rowList</td>
<td>5</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>nonEmpty</td>
<td>7</td>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>moreRows</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>row</td>
<td>10</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>more</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>exp</td>
<td>13</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>tail</td>
<td>16</td>
<td>17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11</td>
</tr>
</tbody>
</table>

\[
15 \text{ LL}(1) \quad \text{NO PREDICTION CONFLICTS}
\]
5. Consider the following grammar

\[
\begin{align*}
\text{File} & \rightarrow \text{Record} \\
& \quad \mid \text{Record File} \\
\text{Record} & \rightarrow \name \ \idnum \ \text{OptGrades} \\
\text{OptGrades} & \rightarrow \text{Grades} \\
& \quad \mid \lambda \\
\text{Grades} & \rightarrow \text{OneGrade} \\
& \quad \mid \text{OneGrade comma Grades} \\
\text{OneGrade} & \rightarrow \text{intlit OptLate} \\
\text{OptLate} & \rightarrow \text{Stars} \\
& \quad \mid \lambda \\
\text{Stars} & \rightarrow \text{star} \\
& \quad \mid \text{Stars star}
\end{align*}
\]

where File is the start non-terminal, and symbols in **bold** are terminals.

(a) Apply the transformations learned in class to left factor the grammar above and write the results below. Give the entire grammar, not just the transformed rules.

\[
\begin{align*}
\text{FILE} & \rightarrow \text{RECORD FILE1} \\
\text{FILE1} & \rightarrow \text{FILE} \mid \lambda \\
\text{RECORD} & \rightarrow \text{NAME IDNUM OPTGRADES} \\
\text{OPTGRADES} & \rightarrow \text{GRADES} \mid \lambda \\
\text{GRADES} & \rightarrow \text{ONEGRADE G1} \\
\text{G1} & \rightarrow \text{COMMA GRADES} \mid \lambda \\
\text{ONEGRADE} & \rightarrow \text{INTLIT OPTLATE}
\end{align*}
\]

(b) If the grammar you wrote above has any immediate left recursion, apply the transformation learned in class to remove it and write the result below. You do not need to give the entire grammar; you can just give the transformed rules.

\[
\begin{align*}
\text{STARS} & \rightarrow \text{STAR S'} \\
S' & \rightarrow \text{STAR S'} \mid \lambda \\
\text{OPTLATE} & \rightarrow \text{STARS} \mid \lambda \\
\text{STARS} & \rightarrow \text{STAR} \\
\end{align*}
\]