1) **Regular Expressions**

If the following descriptions define a regular language then write the corresponding regular expression. Otherwise indicate that the language is not regular. Note that only elegant and compact solutions will receive the full 3 points.

I All strings of 0’s and 1’s representing the binary numbers which are powers of 2.

II All Binary Coded Decimal (BCD) numbers. A BCD number is a binary representation of a decimal number where each decimal digit is encoded using a 4 bit representation of its binary value. For example the BCD of 2509 is 0010010100001001.

III All strings of 0’s and 1’s where at each 0, the number of consecutive 1’s following that 0 is higher than the number of consecutive 1’s preceding that 0.

IV All strings of 0’s and 1’s that do not have more than 3 consecutive 1’s in it.

V All strings of 0’s and 1’s with an even number of 0’s and an even number of 1’s.

\[
\frac{15}{15} + \frac{10}{10} + \frac{15}{15} = \frac{40}{40}
\]
2) Grammar for ìScheme:  

can have integer numbers, few keywords and variables which we will call primitives. The syntax of the language is very simple and as follows:

- A single primitive is a well formed string from the ìScheme language.
- A combination is a well formed string. A combination is defined as a list of combinations or primitives within a pair of matching parentheses.

Examples of few valid ìScheme programs are:

```
82
(+ 8 2 3)
(func () (+ 3 4) 5)
```

The tokens in the language are **number**, **keyword**, **left_paren** or “(“ and **right_paren** or “)”.

Write a grammar for ìScheme.
3) Parser Construction  

You are given the following grammar with the terminal symbols (, ) and term and non-terminals S, E and L.

\[
\begin{align*}
S & \rightarrow E \, \$ \\
E & \rightarrow \text{term} \\
E & \rightarrow ( \, L \, ) \\
L & \rightarrow \text{a} \\
L & \rightarrow E \, L
\end{align*}
\]

I If the terminal term accepts the character X, write 3 well formed strings in this grammar.

a) 

b) 

c) 

II What are the LR(0) items of the 3\textsuperscript{rd} production?

III On the next page an LR(0) state diagram and a parse table for the above grammar is given. However the information for the states 5 and 7 are missing.

a) Fill in the state diagram by adding items, and creating outgoing edges with labels.

b) Fill the appropriate entries in the parse table.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>term</th>
<th></th>
<th>E</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shift S4</td>
<td>error</td>
<td>Shift S3</td>
<td>error</td>
<td>Goto S2</td>
</tr>
<tr>
<td>2</td>
<td>error</td>
<td>error</td>
<td>error</td>
<td>Accept</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Reduce 2</td>
<td>Reduce 2</td>
<td>Reduce 2</td>
<td>Reduce 2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Shift S4</td>
<td>Reduce 4</td>
<td>Shift S3</td>
<td>Reduce 4</td>
<td>Goto S5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduce 4</td>
<td>Reduce 4</td>
<td>Reduce 4</td>
<td>Goto S6</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>error</td>
<td>Shift S7</td>
<td>error</td>
<td>error</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Reduce 5</td>
<td>Reduce 5</td>
<td>Reduce 5</td>
<td>Reduce 5</td>
<td></td>
</tr>
</tbody>
</table>