Problem 1 [5+5+5+10 points]
Give DFA for the following languages, over the alphabet \{0, 1\}

a) Set of all strings containing the substring 0110

b) Set of all strings that do not contain the substring 1010

c) Set of all strings that are exactly of length 5

d) Set of all strings that are at least of length 4 and contains even number of 1’s
Problem 2 [10+10+20 points]
Ex 2.2.5 Give DFA’s accepting following languages
a) Set of strings such that each block of five consecutive symbols contains at least 2 0’s
Solution (or something like it)
The complement of this language is *set of all strings in which some block of 5 consecutive strings contains at most one zero*. So just make an easy NFA for this, which has a start state with 0,1 loop, and then paths leading towards a final state with transition labels 11111, 01111,10111,11011,11101,11110. This will ensure that you reach the final state if *some block of 5 does not contain at least two 0’s*. Now to get the DFA we want, you just need to convert the NFA into a DFA (easy) and then take its complement by changing final states into non-final states (and the order of NFA-DFA conversion and reversing is important)

c) Set of strings that either begin or end (or both) with 01
A first path starting with 01 and then ending in sigma*(covers the case which both being and end), and then the strings that do not start with 01 must end with 01.

d) Set of all strings with number of 0’s in multiples of 5 and number of 1’s in multiple of 3
There are fifteen states, each remembers the no of 0’s and 1 seen. The final state is reached when we have seen 5 zeros(or multiple of 5 zeros, considering 0 to be a multiple of every number) AND multiple of 3 1’s.
Problem 3 [10+10 points]
Construct the min-NFA (an NFA with the minimum number of states) for the following languages, over the alphabet \{0,1\}. Prove that your solution is the best possible

a) \{1010^i \text{ s.t. } n = 0\} \cup \{1010^n \text{ s.t. } n = 0\}, using at most 6 states

There was a typo in the assignment. Instead of \(n = 0\) we should have \(n \geq 0\); (credit will be given in case of correct ans for either case. Here we give solution for non-typo version.

You can make an easy NFA with two paths, one for 1010^i and another for 1010^n; and see that you can easily combine the first two states on each path, to get a 6 state NFA(shown below). You cannot further reduce this, because, after having seen the 10 you need to make a distinction between the 1 that would lead you towards accepting only 0* or 01*. If you don’t then your NFA may end up accepting strings like 1010001, which do not belong to the language.

There have to be two distinct final states, one reached by 101 and then accepts all zeros and the other reachable by 1010 and then we accept all 1’s. So we need to distinguish between these two final states i.e. the one reached with 10101, and 10100. In order to distinguish between two strings of length 5 we need at least 6 states. Therefore the NFA is min.

b) set of all strings s.t there are two 1’s separated by a number of positions that is a multiple of 3

The diagram shows that five states are sufficient. The reason they are necessary is that from the start state, after seeing a 1, we can start counting in multiples of 3 for which we need 3 states, and then on seeing a 1 we can move to a final state. The min acceptable string is 1xxx1; which requires 5 states.
Problem 4 [5+5 points]
Convert the NFAs of problem 3 to corresponding DFAs using Subset Construction

a)  
\[ \rightarrow \{a\} \quad \emptyset \quad \{b\} \\
\{b\} \quad \{c\} \quad \emptyset \\
\{c\} \quad \emptyset \quad \{d,f\} \\
\{d,f\} \quad \{e,f\} \quad \emptyset \\
\{e\} \quad \emptyset \quad \{e\} \\
\{e,f\} \quad \{f\} \quad \emptyset \\
\{f\} \quad \{f\} \quad \emptyset \\
\]

b)  
Here it is interesting to note, as was pointed out by amirali in the discussion forum, that once we reach a subset with f in it, i.e. the final state, we will never go to a non-final state. So for all subsets that contain f, we can replace this by a single sink final state. Therefore there is no need to check 0,1 transitions leaving a set that contains f. All of these are final sink states.

\[ \rightarrow \{a\} \quad \{a\} \quad \{a,b\} \\
\{a,b\} \quad \{a,c\} \quad \{a,b,c\} \\
\{a,c\} \quad \{a,d\} \quad \{a,b,d\} \\
\{a,d\} \quad \{a,e\} \quad \{a,b,e\} \\
\{a,e\} \quad \{a,c\} \quad \{a,b,c,f\} \\
\{a,b,c\} \quad \{a,c,d\} \quad \{a,b,c,d\} \\
\{a,b,d\} \quad \{a,c,e\} \quad \{a,b,c,e\} \\
\{a,b,e\} \quad \{a,c\} \quad \{a,b,c,f\} \\
\{a,c,d\} \quad \{a,d,e\} \quad \{a,b,d,e\} \\
\{a,c,e\} \quad \{a,c,d\} \quad \{a,b,e,d,f\} \\
\{a,d,e\} \quad \{a,c,e\} \quad \{a,b,c,e,f\} \\
\{a,b,c,d\} \quad \{a,c,d,e\} \quad \{a,b,c,d,e\} \\
\{a,b,c,e\} \quad \{a,c,d\} \quad \{a,b,c,d,f\} \\
\{a,b,d,e\} \quad \{a,c,e\} \quad \{a,b,c,e,f\} \\
\{a,c,d,e\} \quad \{a,c,d,e\} \quad \{a,b,c,d,e,f\} \\
\{a,b,c,d,e\} \quad \{a,c,d,e\} \quad \{a,b,c,d,e,f\} \\
\]
Problem 5 [15 points]

Ex 2.3.3 : Convert the following NFA to a DFA and informally describe the language it accepts.

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>{p,q}</td>
</tr>
<tr>
<td>q</td>
<td>{r,s}</td>
</tr>
<tr>
<td>r</td>
<td>{p,r}</td>
</tr>
<tr>
<td>+s</td>
<td>Φ</td>
</tr>
<tr>
<td>+t</td>
<td>Φ</td>
</tr>
</tbody>
</table>

Solution:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>{p,q}</td>
</tr>
<tr>
<td>{p,q}</td>
<td>{p,q,r,s}</td>
</tr>
<tr>
<td>+{p,t}</td>
<td>{p,q}</td>
</tr>
<tr>
<td>+{p,q,r,s}</td>
<td>{p,q,r,s}</td>
</tr>
</tbody>
</table>

This gives a language of all strings that end in 000*, i.e. it cannot end with 1 or 10 at the end.

Problem 6 [15 points]

Prove or disprove that NFAs and DFAs can be used to “remember” in multiples. In other words, it is possible to use these models of computation to do modulo counting for a specific mod \( k \).

We have to show that given a parameter \( k \), we can do modulo counting. We only need a finite number of states to implement a modulo/remainder counter (we need exactly \( k \) states to count from 0 to \( k-1 \) times any symbol has appeared and on the \( k^{th} \) time we go back to count 0). For a given alphabet of size say \( n \), the number of \( k \) sized patterns can \( n^k \), which is a finite number for a given \( n \) and \( k \). Since DFA and NFA can have a finite number of states and thus can distinguish between any pattern.