**Instructions** Follow instructions carefully. This homework consists of two parts; a set of analytical problems and a programming assignment. For the analytical part, you are to work individually, and all work should be your own. Clarity and legibility of presentation of your submission are as important as your answers to problems. You are very strongly advised to typeset your solution using some document processing system. If the grader cannot easily read your writings, you may not be awarded full points even if you claim your answers are correct. Be sure to show *sufficient work* to justify your answer(s). Numerical answers with no explanation will NOT be granted full points. If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. For the programming part, be sure to submit all your program files via the webhandin ([http://www.cse.unl.edu/~raik283h/handin](http://www.cse.unl.edu/~raik283h/handin)). Late homework policy specified in course syllabus will be strictly followed. The CSE academic dishonesty policy is in effect ([http://cse.unl.edu/undergrads/academic_integrity.php](http://cse.unl.edu/undergrads/academic_integrity.php)).

**Analytical Problems**

<table>
<thead>
<tr>
<th>Problem</th>
<th>Page</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1.5</td>
<td>291</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8.2.5</td>
<td>297</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

1. (10 points) Show the comparisons the Boyer-Moore algorithm makes for the pattern “10101” in the text “11101001101101010010”. 

---

RAIK 283  
Homework 4  
Fall 2012  
Due 1:30PM Tuesday, December 4
2. (10 points) Apply the algorithm discussed in class to determine a longest-common-subsequence (LCS) of “10010101” and “010110110”.
3. (10 points) Design a memorized version of LCS-LENGTH algorithm that runs in $O(mn)$ time. For a review of memory functions, refer to Levitin textbook page 294-296.
Programming Assignment

For this part of the homework, you are to work in a team of two or three students. One of your team members should submit all your program files via the webhandin on behalf of your team.

Programming of Linear Programming Solver

Your assignment is to program a general purpose linear programming solver

General Requirements
1. Your solver should implement the two-phase simplex algorithm. You can implement this in tableau form or the revised simplex method (matrix) form.
2. Be sure to test your solver for an assortment of linear programming models to ensure that it works correctly.
3. Your solver should be efficient in terms of execution time (imagine it having to solve 1,000 larger linear programming problems). Be sure to test your solver on several large linear programming models (e.g., models with over 50 decision variables). You can compare your answer to those generated by MS-Excel. Your algorithm will be tested in your spring software engineering class to determine its efficiency compared to others.

Software requirements
Your solver will need to be programmed in .NET

Input format
To ensure that your solver is interchangeable with others (and with the Microsoft solver), you will need to use a specific approach for specifying your linear program formulation. Consider this example:

\[
\begin{align*}
\text{MIN} & \quad Z = 40x + 36y \\
\text{S.T.} & \quad x \leq 8 \quad \text{(inspection 1)} \\
& \quad y \leq 10 \quad \text{(inspection 2)} \\
& \quad 200x + 120y \geq 1800 \quad \text{(production)} \\
& \quad \text{and } x, y \geq 0
\end{align*}
\]

This would be specified as:

```csharp
//Create two decision variables //
Decision x1 = new Decision(Domain.RealNonnegative, “Inspector 1”);
Decision x2 = new Decision(Domain.RealNonnegative, “Inspector 2”);
model.AddDecisions(x1, x2);

//Add Constraints //
model.AddConstraints(“inspector limits”,
1 * x1 <= 8,
1 * x2 <= 10);
model.AddConstraints(“production”,
200 * x1 + 120 * x2 >= 1800);

//Add Objective Function //
model.AddGoal(“cost to hire”, GoalKind.Minimize,
40 * x1 + 36 * x2);
```

Output Format
For a submitted linear program formulation, the output report generated by your solver should indicate one of the following:
- the optimal value and optimal solution for the linear program formulation. It should also state if alternate optimal solutions exist.
- a message saying the linear program formulation is infeasible
- a message saying the linear program formulation is unbounded and no optimal solution could be found

Requirements
- Service must be implemented as a Windows Communication Foundation (WCF) service.
- You may use any .NET language as long as it will compile into a .DLL (C# is recommended).
- Service MUST conform to the service and data contracts provided.
- Unit tests for the service must be written to validate functionality.
- Create a separate project that consumes the service to demo functionality (keep it simple).

Deliverables
- A WCF .DLL file that implements the contract provided and can be deployed anywhere.
- A Visual Studio 2010 solution that includes unit tests and demo project.