CSC445 Programming Project. Fall 2016 (20pts)

`Student Name:` `Student Number:`
`Instructor George Tzanetakis`

Question	Value	Mark
1	5	
2	5	
3	5	
4	5	
Total	20	

Overview

The goal of this assignment is to familiarize you with numerical programming in the context of linear programming. Your implementation will be pedagogical rather than efficient but hopefully it will help you better understand how solvers are implemented and how to interface with matrices and linear algebra packages. Be aware that some of the questions require much more work than others. This is a programming assignment and ALL code submitted must be your original work. If you use code taken from the internet, an API, other students, previous model solutions or other sources and submit it as your own work you will not get credit for that code. Further if your source is not acknowledged, you are subject to disciplinary action according to the department policies for plagiarism. A random subset of students will also be selected to do walkthroughs of their submitted code with the instructor.

I have indicated what level I consider each question to be (basic, regular, advanced). Your deliverable should be a **SINGLE PDF - NO EXCEPTIONS** submitted through ConneX. Don't hesitate to contact the instructor via email or utilize the chat room of the ConneX course website for any questions/clarifications you might need.

IMPORTANT: YOU SHOULD ONLY SUBMIT A SINGLE PDF FILE WITH YOUR REPORT THROUGH CONNEX. ANY OTHER FORMAT WILL NOT BE ACCEPTED

Question 1 (5 points - Basic)

Your goal is to provide a pedagogical implementation of the revised simplex method for solving linear programs. I use the term pedagogical to refer to following characteristics for your code: it should be written to be readable with comments and descriptive variable names, each matrix used during the computation steps should being explicitly named, and the data should be copied as needed without worrying about efficiency. Also unlike actual revised simplex implementations, the calculations of the inverse matrix, and matrix multiplications should be done from scratch each iteration.

You CAN use code that you did not write for matrix multiplication and inverting matrices. Everything else should be your own code. If you are using *Python*, the numpy/scipy packages have everything you need and *MATLAB/Octave* is designed for that kind of stuff. For other programming languages such as *C*, *C++*, and *Java* it is easy to find libraries for basic linear algebra operations but if you are having trouble finding an implementation feel free to contact me. There are tons of revised Simplex implementations that one can find online, but most of them are designed to operate directly on the data by accessing the corresponding columns rather than copying and creating separate matrices and solve systems of linear equations rather than directly using the inverse (a very inefficient approach). The goal of this programming project is to understand the algorithm NOT to provide an efficient implementation. You are welcome to look at existing implementatiosn but in my opinion this will confuse you more than help you. You can and are expected to reuse parts of your code for assignment 2.

More specifically the input to your algorithm should in the form of .csv files similarly to assignment 2. The input will consist of the following matrices/vectors A, b, c specifying the problem as typically. During each iteration you should have explicit representations (that is separate matrices/vectors as needed) $\mathcal{B}$, $\mathcal{N}$, B, N, $x^*_{\mathcal{B}}$, $z^*_{\mathcal{N}}$, B^{-1}, e_j, e_i, $\Delta X_{\mathcal{B}}$, and $\Delta Z_{\mathcal{N}}$ as well as the scalars s, t. Your code should output the final optimal solution and associated value of the objective function.

For solving the problem you can assume that the solution provided is primal feasible. If it is not you should output a warning message and the code should not attempt to solve the problem. In the next question you will be asked to extend this to the more general cases when either the primal or the dual solution provided is not feasible and a two phase procedure is required.

The deliverabe for this question is a code listing, as well as example input/output files that show that it works.

Question 2 (5 points - Basic)

In this question you are asked to extend and thoroughly test the code you wrote in question 1. In terms of extensions you need to also implement the revised dual simplex method (similarly using matrices and directlycomputing the invesre) - as you probably expect it is essentially identical to the revised primal simplex other than which matrices are involved in each computation. Figure 6.1 of the 4th edition of the textbook is a good summary. When you have the dual phase implemented then do a two-phase method that handles all cases of problems as described in section 5 of Chapter 6. You can choose to do either a Phase 1 type of initialization using an auxiliary problem or use the approach described in section 7 of chapter 8 (A dual-based Phase I Algorithm).

Also using a unit testing framework of your choice add unit tests (at least 5) that test diffferent parts of your implementation. If you are working in MATLAB check:
`www.mathworks.com/help/matlab/matlab-unit-test-framework.html`. At least one unit test should use duality theory as a certificate of optimality to check your implementation.

Question 3 (8 points - Regular)

In this question the goal is to produce beautiful printed output of all the stages of the algorithm. For only this question you can impose some reasonable limits to the number of variables/constraints so that the output fits on the page. For the output you will need to use `https://en.wikipedia.org/wiki/LaTeX` which is a document preparation system that uses plain text with markup for formatting. It can produce nice looking math notation and is what I use for my slides and what is used in your textbook. All you need to produce the output is the ability to output plain text which makes it easy to create pretty printing in any programming language.

For example consider the following output:

$$\Delta X_{\mathcal{B}} = B^{-1}Ne_j = \begin{bmatrix} 1 & -1 \\ 2 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}$$

This was produced by the following code:

```
\documentclass [12pt] {article}
\usepackage{amsmath}
\usepackage{url}
\pagestyle{plain}
\begin{document}

\section{An example}
\[
\Delta X_{\mathcal B} = B^{-1} N e_j =
  \begin{bmatrix}
    1 & -1 \\
    2 & -1 \\
    0 & 1 \\
  \end{bmatrix}
  \begin{bmatrix}
    1 \\
    0
  \end{bmatrix}
  = \begin{bmatrix}
    1 \\
    2 \\
    0
    \end{bmatrix}
\]
\end{document}
```

You will need to learn how to produce pdf output from LaTeX source (there are different ways depending on what software you use) and then I want you to generate both text and matrices for all the steps of the algorithm. Basically with the appropriate input data your code should be able to produce output that looks like Section 3.1 or Section 3.2 of Chapter 6. The numbers that show up in the equations should not be fixed but based on the problem your implementation is solving. Your output for the example given in the textbook should be as close to the actual textbook as you can get it to be.

For deliverable provide a listing of the relevant code as well as some examples of output.

Question 4 (2 points - Advanced)

Your goal is to implement the criss-cross method for sovling linear problems. This is intentionally a somewhat more open ended question that requires some initiative to understand how to do it. It is more similar to what you might be faced with in the future when working in industry where it is unlikely that you will find something as well described as the Simplex method.

The criss-cross method is defined as follows. One starts with an initial dictionary that can be both primal and dual infeasible. One assume that all of the variables (both original and slack) have been ordered in some way. Then from all the infeasibilities, select the one corresponding to the variable that appears first in the ordering. If this infeasibility is a dual infeasibility (i.e the corresponding variable is nonbasic) then the associated nonbasic variable is selected as the entering variable in a primal pivot. The leaving variable is that basic variable for which the dictionary's coefficient on the entering variable is negative and whose place in the order is is minimal among is minimal among all other basic variables for which the coefficient is negative. If, on the other hand, the infeasibility is a primal infeasibility (i.e the corresponding variable is basic), then the associated basic variable is selected as the leaving variable in a dual pivot. The entering variable is that nonbasic variable for which the dictionary's coefficient on the leaving variable is positive and whoe place in the order is minimal among all other nonbasic variables for which the coefficient is positive. In your code you will have two arrays storing the mapping of variables (basic and non-basic) to row and columns of the dictionary. Since the original indices form an ordering, they can be used to implement the criss-cross method's pivot rule.

Your deliverable again should be a listing of your code as well as some examples of input/output to show that it works. You should also implement some unit tests including one that ensure that the optimal solution found using your Simplex implementation is the same as the one found using the Criss-cross method.