(Exercise 4) For the Exercise below here are a few coding tips.

- The idea is to write some code which works for any valid A, b, c, m, n etc.... However, when testing whether what you've written works it will be useful to actually have some values stored into those variables. I would suggest you use the problem from Application 2.3, so rhs $<-$**c**(6, 4, 3), obj $<-$**c**(2, 3) etc ...I've used variable names rhs and obj here. Previously I have used names with dots in them, you can use whatever names you like, just choose names which are memorable and describe what they hold. I recommend you store values into these variables before attempting the exercise, and then you can check as you go along whether the new variables you're creating look like you want them to.

- You will probably find it beneficial to know what the following commands in R do: **rep**(−2, 8), **diag**(4), **cbind**(), **t**(), **c**(), %∗%, **which**, **which.min**(), **which.max**(), M[,2], M[, **c** (1,2,4)] .

- Here some sample code which takes a matrix A, finds its tranpose, adds a new column of ones, before finally finding the largest element in the first row.

```
B <- t(A)
B <- cbind(B, rep(1, nrow(B)))
toprow <- B[1, ]
max(toprow)
which.max(toprow)
```

- HW Exercise 2.10 also provides some useful computing practice.

- In HW Exercise 2.10, you will use lp again to solve problems we've been solving by hand recently. One suggestion is to ask the function to reveal a little bit more of its inner workings. The following command:

  lp(<the usual stuff>, compute.sens=TRUE)**$**duals

  asks lp to tell you the values of what it calls the *dual variables*. We will learn more about these later (you should already have a good idea what they are). In those exercises you are asked to look at the dual variable values while also looking at the final simplex tableau calculated by hand in the notes to see if you can spot anything of interest.

Exercise 4 (Zeno)**.** Coding up the simplex algorithm.

(a) Assuming the constants m, n, and vectors and matrices of a given problem: maximize $c^T x$ s.t. $Ax \leq b$, $x \geq 0$ are already stored as variables... write some lines of code which, do the following:

  (i) creates a new extended vector $\hat{c} \in \mathbb{R}^{m+n}$, and a new matrix $\hat{A} \in \mathcal{M}_{m\times(n+m)}(\mathbb{R})$, which represent the augmented problem: maximize $\hat{c}^T x$ s.t. $\hat{A}x = b$, $x \geq 0$, for $x \in \mathbb{R}^{m+n}$.

  (ii) identifies an initial basis and basic feasible solution, storing them both as vectors.

(b) Write some code which identifies a column (i.e. an x_i) to enter the basis.

  [If you get stuck on part (c) then skip down and try part (d) below, or at least comment on your difficulties]

(c) (Tricky) Write some code which finds the pivot row, for a given column. (Hint: first step is to use the bfs vector to create a new vector which holds the ratios)

(d) Extension*: Write some code to implement the pivoting procedure. Try it for a given problem, with three constraints. It's easiest to do this by identifying a 3x3 matrix E to pre-multiply A by (so that E represents the row operations in their totality) to get the desired answer. Hint: a 3x3 matrix is determinable from its action on three independent column vectors, what does it/pivoting do to the following three columns: the new basis variable and the two old basis variables? For this part you may like the function **solve**(A) which finds the inverse of matrix A.