(Exercise 5) We saw in Chapter 1 how finding the dual of a linear program could be done from first principles. It is important that you know how to do this, in particular the delicate arguments required when considering maximizing non-positive terms, and positive and negative coefficients.

In this exercise you will write your first proper function, which will be a procedure for taking a standard LP and returning its dual automatically!

BE WARNED: the dual is very sensitive to things like whether you have augmented a problem, it is sensitive to whether constraints are $\geq$, $\leq$ and $=$ and to max v min. So do be careful if applying 'standard results' that you check the conditions really do apply.

Here's how a basic function in R looks. This function takes two variables as its input and returns their sum:

```
frog <- function(m, n) {
    a <- m + n
    return(a)
}
```

Now try frog(1, 8). For those of you with some programming experience an R function doesn't need you to specify why type of data the variables are! You can actually apply frog above to vectors too! The indenting is optional, but recommended to make things easier to read.

In R we use the return function to specify the *return variable*. Here's a basic function in R which takes two variables for its input and returns both their sum and their product:

```
rabbit <- function(m, n) {
    a <- m + n
    b <- m * n
    return(list(rabprod=b, rabsum=a))
}
```

Now things are more interesting, this function returns an R-list of variables.

Try rabbit(3, 4), rabbit(3, 4)**$**rabprod and rabbit(3, 4)**$**rabsum

You can use the **return**() function inside a function whenever you want it to exit immediately with a known answer. Note that by using the rabprod=b and rabsum=a syntax you can make the returned answer easier to access later, as you've explicitly labelled the outputs.

Some functions you might like: **t**(), **return**(), **if**. In particular, for later parts of the exercise you may need to compare strings to find out if the problem being given is a maximization or minimization. For example, you could guess what the following code does:

```
T <- "max"
if (T == "max") {
  print("Hello!")
  new.T <- "min"
}
```

Note the use of the == operator to ask R if something is equal to something else. Do NOT use = by itself as this is an assignment operator that works like <− for storing the RHS into the LHS.

Exercise 5 (Zhang Heng – Dual problems)**.** In the problems below you will be asked to create some functions: they will have variables/parameters/arguments which are *passed* to them and will need to return a collection/list of parameters/variables too. Read the advice above for hints on writing functions. As a further suggestion, when creating functions you should make sure to provide to the function itself (in its parameters/variables/arguments) all the information it needs to perform its operations (and no extra ones!). Have a think about the information you need to provide the lp function if you're looking for inspiration for what info a function concerning linear programs requires to know what problem is being solved.

(i) Explain briefly in words where the dual problem comes from. (One sentence should be enough)

(ii) State the dual of $\max c^T x$ s.t. $Ax \leq b$, $x \geq 0$.

(iii) Write a function in R which takes as its parameters vectors, matrices etc.. describing a problem like that in part (ii) and outputs an R-list containing similar R variables BUT describing the dual problem. The function 'finds the dual' of a given problem.

(iv) Set up some variables to describe a standard maximization LP example/application (from the notes): apply lp to find the optimal value; apply your function to find parameters for the dual problem; then directly use the output from your function inside a call to lp to verify the primal and dual have the same optimal value.

(v) Find the optimal basic feasible solutions to the primal and dual for the problem you studied, and verify the conditions of the complementary slackness theorem hold.

(vi) Your code currently will turn a max problem into a min. Extend your code 'to find the dual' to cope with someone passing variables describing a problem of the form $\min c^T y$ s.t. $Dy \geq b$, $y \geq 0$, for a constraint matrix D. Your code, should be the inverse of the procedure above. It would be nice to check it really does so!

In this module we have only given a general formula for the dual of maximization problems with entirely $\leq$ constraints, or minimizations with $\geq$ constraints: there is a good reason for this – there isn't as nice a formula in other cases!

Extension*: Now modify your code to cope with also being able to deal with $\geq$ constraints in a maximization problem. I shan't give advice on how to do this, it's for you to think about (or search online for, or look in a book for).

Extension*: Look carefully at our derivation of the dual from Chapter 1, applying it to the dual of a maximization problem, or search online to learn more about the dual, to speculate on how complicated it would be to write code to cope with the appearance of the full range of: max, min, $\leq$, $=$, and $\geq$. (see p295 in Winston)