Practical 5: Submit solutions in Matlab Grader before Fri 18th October at 5pm.

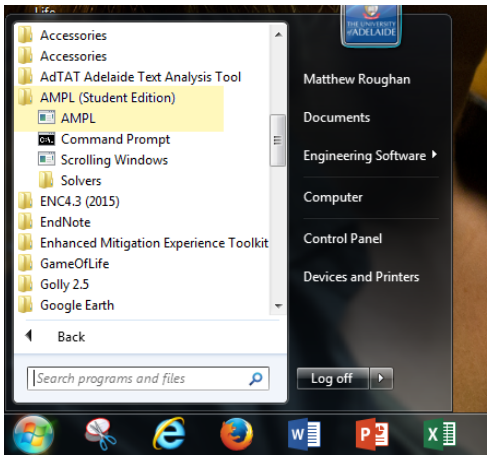
Bring your lecture notes with you for reference.

The aim of this practical is to learn AMPL.

Tasks

1. The first task is to input our first LP (from Lecture 1) in AMPL.

   (a) Start AMPL from the start menu (see below for a screen shot). Once you start AMPL you should see a window open, and display the prompt `ampl:`



   (b) Also start Notepad (or equivalent text editor) to edit the required input files.

   (c) Use Notepad to create a file called `first_problem.mod`. We will define our *model* in this file. In the model we specify the *variables*, *objective*, and *constraints*.

   Make a note of where you store the file. For the sake of our instructions, we will assume you store it in the folder/file called:

   `U:\myFiles\first_problem.mod;`

   However, you can store it where-ever you like, and in whatever filename you like, as long as you use the same filename below.

   (d) Enter the following into the file, and **save** it.

```
1  ## First problem from lecture 1
2  var x_desks >= 0 integer;
3  var x_chairs >= 0 integer;
4  var x_beds >= 0 integer;
5
6  maximize profit: 13*x_desks + 12*x_chairs + 17*x_beds;
7
8  subject to labour: 2*x_desks + 1*x_chairs + 2*x_beds <= 225;
9  subject to metal:  1*x_desks + 1*x_chairs + 1*x_beds <= 117;
10 subject to wood:   3*x_desks + 3*x_chairs + 4*x_beds <= 420;
```

   Note how we have created the variables as non-negative integers on lines 2-4.

   Common errors in typing this:

   - **Make sure to type the semi-colon ';' at the end of each line.**
   - **The objective "maximize" is spelled with a 'z'.**

(e) Now input the model into AMPL by typing at the AMPL prompt `ampl:` (you don't need to type the part in green). You can cut and paste your filename from Notepad.

```
ampl: model U:\myFiles\first_problem.mod;
```

Hints:

- **Make sure the filename matches the place *you* saved your file.**
- **Make sure to type the semi-colon ';' at the end of the line.**
- **If you need to change something, you might need to "reset;" before re-entering to clear the workspace.**

(f) Set the solver to be `LPsolve` using the AMPL command:

```
ampl: option solver lpsolve;
```

You are allowed to use alternative solvers, but this is the one installed in the labs and the exact form of output may vary.

(g) Then run the model and output the results with

```
ampl: solve;
ampl: display <variablename> ;
ampl: display <objective> ;
```

where you substitute your variables' names, and your objective's name. It should print the result.

(h) **Enter your solution into Matlab Grader to check it is correct.**

(i) Now that you have used AMPL to solve the LP with integer variables (as specified above), remove the integer requirement, and solve again and compare the results.

2. Use AMPL to rewrite the above problem in a more general form:

   - Your `model` file should have general definitions that could apply for any set of furniture types. An example is provided below, and you can download it from http://www.maths.adelaide.edu.au/matthew.roughan/notes/OORII/10ampl.html
   - Your `data` file should provide the actual values: the types of furniture, their related profit, and the relative resource requirements. An example is provided below, which you can download from the same web page.

Example model file.

```
## First problem from lecture 1, but generalised

# define various parameters, but we will read the values from the data file
set FURNITURE;

param available_labour;
param available_metal;
param available_wood;

param labour{FURNITURE};
param metal{FURNITURE};
param wood{FURNITURE};

param profit{FURNITURE};

# create a set of variables, one for each type of item defined in the data
var x{FURNITURE} >= 0 integer;

# create a a general model for the problem
maximize total_profit: sum{i in FURNITURE} profit[i] * x[i];

subject to total_labour: sum{i in FURNITURE} labour[i] * x[i] <= available_labour;
subject to total_metal: sum{i in FURNITURE} metal[i] * x[i] <= available_metal;
subject to total_wood: sum{i in FURNITURE} wood[i] * x[i] <= available_wood;
```

Example data file.

```
# data file for first problem, generalised

# types of available items
set FURNITURE := desk chair bed;

param available_labour := 225; # limit on labour
param available_metal := 117; # limit on metal
param available_wood := 420; # limit on wood

# resource cost of various items
param: labour metal wood :=
  desk   2    1    3
 chair   1    1    3
   bed   2    1    4;

# profits from various items
param profit
  desk   13
 chair   12
   bed   17;

# note how the "param" commands match those in the model
```

Most of the AMPL commands to run the new problem are just the same as before, but you will also need to use the `data` command. Assume you store your data file in

```
U:\myFiles\first_problem.dat;
```

The AMPL command to include this data in the problem is

```
ampl: data U:\myFiles\first_problem.dat;
```

Then you can solve it and display the results, but now you can display all of the variables with one command

```
ampl: display x;
```

(a) Create the data and model files, and use the appropriate commands to solve.

(b) Modify the problem to have a 4th type of furniture a `table`, which requires 1 unit of labour, 2 of metal, and 3 of wood, and which has a profit of 16. Rerun your optimisation, and **check the solution using Matlab Grader.**

(c) Now imagine that we don't want to flood the market with any one type of item. Hence introduce a maximum for production of 45 of any one type of article. Add this maximum to your model, then resolve the problem and **check the solution using Matlab Grader.**

Hint: you can add the constraint with one line of code in your model.