IMSE 500
Model of Operations Research

ASSIGNMENT #2

Readings: *AMPL: A Modeling Language for Mathematical Programming,* chapters 2–3.

1 In the presentation of the McDonald's diet example, you saw that there is a tradeoff between the conflicting objectives of total calories and total cost. To explore this tradeoff further, we can replace the `minimize` statement in the diet model with the following objective function definition:

```
param wt >= 0, <= 1;

var Total_Cost = sum {j in FOOD} cost[j] * Buy[j];
var Total_Cals = sum {j in FOOD} amt["Cals",j] * Buy[j];

minimize Tradeoff: 1000*wt * Total_Cost + (1-wt) * Total_Cals;
```

The revised model and some representative data are posted with this assignment, in files `mcdietwt.mod` and `mcdietwt2.dat`.

a: Suppose that you set the parameter `wt` to zero and then apply a solver, such as CPLEX (or Gurobi or FortMP), that can deal with integer variables:

```
ampl: model mcdietwt.mod;
ampl: data mcdietwt2.dat;

ampl: option solver cplex;

ampl: let wt := 0.0;

ampl: solve;
CPLEX 12.2.0.0: optimal integer solution; objective 2500
6 MIP simplex iterations
1 branch-and-bound nodes

ampl: display Total_Cost, Total_Cals;
Total_Cost = 17.16
Total_Cals = 2500
```

Which is minimized in this case, total cost or total calories?

b: Now suppose that you set the parameter `wt` to one and then solve, like this:

```
ampl: let wt := 1.0;

ampl: solve;
CPLEX 12.2.0.0: optimal integer solution; objective 15200
196 MIP simplex iterations
156 branch-and-bound nodes

ampl: display Total_Cost, Total_Cals;
Total_Cost = 15.20
Total_Cals = 3950
```

Which is minimized in this case, total cost or total calories?

c: If you set `wt` to certain values between zero and one, you will get solutions different from the two shown above. By trying different values of `wt`, find three such solutions. Report them in a table like this:

wt	cost	calories
0.0	\$17.16	2500
—	—	—
—	—	—
—	—	—
1.0	\$15.20	3950

Replace the — entries with the values that you find.

d: Make a two-dimensional plot of all five solutions you have found, with cost along the horizontal axis and calories along the vertical axis.

e: You can see from your plot that whenever one of your solutions is better in cost, it is worse in calories; and whenever one is better in calories, it is worse in cost. A set of solutions having this property is said to be *efficient* (with respect to cost and calories).

Do you think that any collection of solutions found for some diet problem in this way must be efficient? Give a brief explanation of your reasoning.

2 A chain of fast-food restaurants operates 7 days a week, and has the following daily requirements for kitchen employees:

	Day	*Minimum number of employees required*
1	Monday	16
2	Tuesday	11
3	Wednesday	17
4	Thursday	13
5	Friday	15
6	Saturday	19
7	Sunday	14

Each employee is scheduled to work a stretch of *five* consecutive days, such as Monday through Friday, Tuesday through Saturday, or Friday through Tuesday. The management wants to know how many employees should be assigned each schedule, so that the requirements are satisfied on every day, and the total number of employees is as small as possible.

You have seen how a variation on the minimum-cost input model can be formulated for situations of this kind. First we define two underlying sets, one of days (corresponding to outputs) and one of work plans (corresponding to inputs):

```
set DAYS;
set PLANS;
```

Next we define the numbers that figure in the model. Certainly, there are the numbers of employees needed, one number for each day. In AMPL we represent these by declaring a parameter `need` indexed over the set `DAYS`:

```
param need {DAYS} > 0;
```

We also require some way to describe the work plans. For this purpose we declare a table of numbers `sched` indexed over all combinations of `DAYS` and `PLANS`:

```
param sched {DAYS,PLANS} binary;
```

For any combination of a member `d` of `DAYS` and a member `p` of `PLANS`, there exists an entry `sched[d,p]`; the keyword `binary` in the declaration specifies that each entry must be either a 0 or a 1. We interpret these entries as follows:

- ▷ If `sched[d,p]` is 1, then people following plan `p` must work on day `d`.
- ▷ If `sched[d,p]` is 0, then people following plan `p` don't work on day `d`.

Using these data definitions, the rest of the AMPL model defines the following variables, objective and constraints:

```
var Work {PLANS} >= 0;

minimize Employees: sum {p in PLANS} Work[p];

subj to MeetNeed {d in DAYS}:
   sum {p in PLANS} sched[d,p] * Work[p] >= need[d];
```

You may find the example given in the class to be helpful with this question, although the situation there is somewhat different.

a: The key to this formulation is the use of the table `sched` in describing the constraints. Give a brief explanation of why the constraints, as written in the AMPL model, guarantee that the solution will provide enough employees on each day. Your explanation should make a good comment to put in the model so that another analyst would understand it.

b: To solve a particular linear program using this AMPL model, it is necessary to supply particular data values. In the case of the requirements and work plans specified at the beginning of this problem, the appropriate data can be represented in AMPL as follows:

```
set DAYS := Mon Tue Wed Thu Fri Sat Sun ;
set PLANS := SaSu SuMo MoTu TuWe WeTh ThFr FrSa ;

param need :=  Mon 16   Tue 11   Wed 17   Thu 13
               Fri 15   Sat 19   Sun 14 ;

param sched: SaSu SuMo MoTu TuWe WeTh ThFr FrSa :=
        Mon    1    0    0    1    1    1    1
        Tue    1    1    0    0    1    1    1
        Wed    1    1    1    0    0    1    1
        Thu    1    1    1    1    0    0    1
        Fri    1    1    1    1    1    0    0
        Sat    0    1    1    1    1    1    0
        Sun    0    0    1    1    1    1    1  ;
```

The complete AMPL model and data are posted with this assignment in the files `sched1.mod` and `sched1.dat`. Run AMPL, using any linear programming solver, to get an optimal solution.

In your solution, how many employees follow each work plan?

c: Your solution in (**b**) unfortunately requires fractional numbers of people to work some days. To get an implementable solution that still meets the staffing constraints, you can round each fractional variable up to the next-highest whole number.

In the rounded-up solution, which work schedules should be used, and how many employees should follow each? How many employees are used in total?

d: To get the best possible whole-number solution, change your AMPL model so that all the variables are declared to be integer-valued:

```
var Work {PLANS} integer >= 0;
```

Save your modified model in `sched2.mod`. Now switch (if necessary) to a solver (such as CPLEX) that can handle integrality restrictions, and compute a new solution for the combination of `sched2.mod` and `sched1.dat`.

In the optimal whole-number solution, which work schedules should be used, and how many employees should follow each? How many employees are used in total?

State in a sentence or two how this optimal solution differs from the rounded solution in (**c**).

e: By typing *`display need, MeetNeed.body;`* you can get a table from AMPL that shows for each day the number of employees needed and the number actually working. On which days is the number working greater than the number needed, and by how much?

f: To rule out solutions that use "extra" employees, you might change the `>=` in the `MeetNeed` constraint to be `=`. Solve with this revision in the model and with the same data. Explain in a few sentences what results you get and what they tell you about the problem.

g: You can use AMPL to compute the following number:

```
ampl: display ceil(sum {d in DAYS} need[d] / 5);
```

(The function `ceil` rounds a number up to the next highest integer.) Explain in words why this number is a *lower bound* on the total number of employees required, assuming that every employee will work 5 days of each week.

What is the difference between the actual number of employees used in your solution in (**d**) and this lower bound?

h: By experimenting with various possibilities, find one or more additional five-day work plans that, when appended to the data in (**d**), give a solution that achieves the lower bound. *(The days off in the additional work plans will not be consecutive.)* How does the resulting solution differ from the one in (**d**)?

You can use `sched2.mod` for the model in this part, but you'll need to modify the data file. Store the new data file in `sched2.dat`.

Upload the files `sched2.mod` and `sched2.dat` along with the file containing your assignment writeup.