Computational Microeconomics - Practice Midterm

Problem 1: True or False (30 points).

Label each of the following statements as true or false. You are not required to give any explanation.

1. If a problem can be (efficiently) modeled as a linear program, then it is solvable in polynomial time.
2. If a problem can be (efficiently) modeled as an integer program, then it is NP-hard.
3. The Dutch auction (in which the price goes down until somebody claims the item) and the first-price sealed-bid auction are strategically equivalent.
4. In a combinatorial auction, if we allow bids to be partially accepted, this can increase the optimal solution value.
5. Any valuation function can be expressed in the XOR language.
6. Arrow-Debreu securities (which pay out 1 in one state and 0 in all other states) can be used to construct any other security.
7. In a kidney exchange, if we increase k (the maximum number of people in a cycle), then the optimal solution value cannot get worse (at least the same number of people will get a kidney).
8. In a kidney exchange, if we increase k, it cannot make the problem harder computationally.
9. If most voters prefer A to B, and most voters prefer B to C, then most voters must prefer A to C.
10. If most voters prefer A to B, and most voters prefer B to C, then most voters must prefer C to A.

Problem 2: A combinatorial reverse auction with reserve prices (35 points).

Recall that in a combinatorial reverse auction, the auctioneer seeks to procure a set of items. She aims to obtain all the items, at the lowest possible total cost. If she gets multiple copies of an item, that is fine, but she needs to obtain every item at least once. This corresponds to the following integer program (where x indicates whether a bid is accepted, v the value of the bid, and c which items occur in which bid):

```
set ITEMS;
set BIDS;

var x{j in BIDS}, binary;

param v{j in BIDS};
param c{i in ITEMS, j in BIDS}, binary;

minimize cost: sum{j in BIDS} x[j]*v[j];

s.t. at_least_once{i in ITEMS}: sum{j in BIDS} x[j]*c[i,j] >= 1;
```

This implies that the auctioneer is willing to pay *any* price to obtain the items. For example, if there is only one bid for item A, at a value of \$1,000,000, the auctioneer is forced to accept it. This is generally not realistic. Instead, the auctioneer may have a *reserve price* $r[i]$ for each item i, indicating that she will not pay more than $r[i]$ for item i. Another way to interpret this is that she has an alternative way to obtain item i (outside the auction) which will cost $r[i]$, and she will try to minimize cost, potentially using this outside option on some items. Thus, the reserve prices can be thought of as additional bids on individual items.

For example, suppose that there are three items, A, B, and C. There are three bids, $(\{A, B\}, 10), (\{B, C\}, 3), (\{C\}, 2)$. Without reserve prices, the auctioneer will accept the first and third bids, at a cost of $10 + 2 = 12$. However, now suppose that there are reserve prices $r[A] = 8, r[B] = 3, r[C] = 3$. Then, the auctioneer is better off accepting the second bid, and obtaining A outside the auction (at the reserve price), at a total cost of $3 + 8 = 11$, which is less than 12.

a (10 points). Suppose that there are four items, A, B, C, and D. Find the optimal (lowest-cost) solution for the following bids:
$(\{A, B\}, 6), (\{B, C\}, 8), (\{B, D\}, 8), (\{B, C, D\}, 14)$,
and the following reserve prices:
$r[A] = 3, r[B] = 4, r[C] = 6, r[D] = 5$. Also state what the lowest cost actually is.

b (25 points). Modify the integer program above to take reserve prices into account. You should have `param r{i in ITEMS};` for the reserve prices. You are allowed to add new variables if you want.

Problem 3: Creating project groups of at most 3 students (35 points).

We have a course in which the students need to do a project. They can work in teams of at most 3 people. For each two students i, j, there is a parameter $v[i,j]$ which indicates how much i likes working with j (and conversely, $v[j,i]$ indicates how much j likes working with i). It can also be the case that $v[i,j] < 0$, which means that i dislikes working with j. We say that the value $v[i,j]$ is "realized" if i and j actually end up in the same team. The goal is to create teams to maximize the sum of the realized values.

For example, suppose the students are A, B, C, D, and we have $v[A,B] = 4, v[B,A] = 6, v[B,C] = 10, v[A,C] = -4, v[C,A] = 1, v[B,D] = 3$ (all other values are 0). Then, the optimal solution is to put A, B, C in one team, and D in a team by himself. This gives a total value of $4 + 6 + 10 - 4 + 1 = 17$. Note that A does not like being in a team with C, but it's worth it because of the high values of putting A and B in the same team, and B and C in the same team. We'd like to add D to the big team and get an additional 3, but cannot do so because there can be at most 3 students in a team.

a (10 points). Suppose the students are A, B, C, D, and we have $v[A,B] = 15, v[B,A] = -5, v[A,C] = 3, v[B,C] = 3, v[D,B] = 2, v[C,D] = 7, v[D,C] = 3$ (all other values are 0). What is the optimal solution (and what is its total realized value)?

b (25 points). Complete the following integer program for optimizing the teams. $x[i,j]$ is 1 if i and j are in the same team. Note that we must have $x[i,i] = 1$ for all i (a student is in always in her own team). Also, if $x[i,j] = 1$, then we also must have $x[j,i] = 1$. Finally, if $x[i,j] = 1$ and $x[j,k] = 1$, then we also must have $x[i,k] = 1$. You must add constraints to make sure that all of this holds (and that there are at most 3 students per team).

```
set STUDENTS;
var x{i in STUDENTS, j in STUDENTS}, binary;
param v{i in STUDENTS, j in STUDENTS};
```