Computational Microeconomics
Practice Midterm 2

Problem 1: True or False (24 points).

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

1. The Kemeny voting rule (where we try to minimize the total weight of inverted edges in the pairwise majority graph) is Condorcet-consistent (if there's a Condorcet winner, the Kemeny voting rule will rank the Condorcet winner first).
2. When preferences are single-peaked, the winner under the Kemeny voting rule (i.e., the alternative ranked first by Kemeny) can be found in polynomial time.
3. The Borda rule ($m-1$ points for being ranked first, $m-2$ for being ranked second, $\ldots$) satisfies the majority criterion (an alternative ranked first by more than half the votes always wins).
4. The Copeland rule (2 points for a pairwise election victory, 1 point for a tie) satisfies the majority criterion (an alternative ranked first by more than half the votes always wins).
5. Suppose preferences are single-peaked and we use the median-voter rule discussed in class. This satisfies the majority criterion (an alternative ranked first by more than half the votes always wins).
6. We know how to formulate the kidney exchange problem (with some maximum cycle length k, for example $k = 3$) as a polynomial-sized linear program.
7. In kidney exchange, if we decrease k, it cannot make the problem harder computationally.
8. In kidney exchange, the following scenario can happen (i.e., there exists an instance for which this happens): with maximum cycle length $k = 2$ a given vertex v is part of the optimal solution, but with $k = 3$ that same vertex v is no longer part of the optimal solution.
9. An agent with a utility function over money who maximizes expected utility can always be classified as one of the following: risk-averse, risk-neutral, or risk-seeking.
10. We know how to compute a welfare-maximizing Nash equilibrium of a general-sum 2-player normal-form game in polynomial time.

Problem 2: Voting with single-peaked preferences (16 points).

Recall single-peaked preferences: the candidates are ordered on a line, every voter has a most preferred candidate and prefers candidates closer to that most preferred candidate. We have seen that there are good reasons to choose the median voter's most preferred candidate. The Copeland rule (for example) will always choose this candidate too. But some other rules will not.

For example, suppose the alternatives are a, b, and c, and appear on the line in that order. We have seven voters. Three voters rank the alternatives $a \succ b \succ c$, three voters rank them $c \succ b \succ a$, and one voter ranks them $b \succ a \succ c$. The last voter is the median voter, so b wins under the median voter rule. Also, b wins both pairwise elections, so it wins under the Copeland rule.

Name two voting rules that do not choose b for the votes above (and so do not always choose the same outcome as the median voter rule). For each of these two rules, **state** which candidate (or candidates, if there is a tie) the rule chooses instead, and very briefly **explain** why. If you don't remember the name of a rule that you want to use, just describe it. But it should be one of the rules we covered in class (and not "silly" rules like the dictator rule).

Problem 3: A simple approval voting game (30 points).

Two proposals, A and B, are being considered. At most one of them can be implemented (but it is also possible to implement *neither* one). There are two voters, 1 and 2, who must *approve* or *disapprove* each of the proposals. They make all these decisions at the same time.

A proposal can only be implemented if *both* voters approve it. If *both* proposals are approved by *both* voters, then we toss a fair coin to determine which one is implemented. If exactly one proposal is approved by both voters, then that one is implemented. Otherwise, neither proposal is implemented.

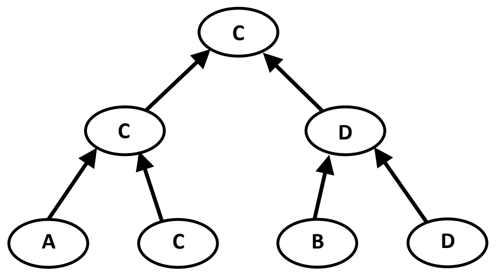
Voter 1 would receive utility 2 for A being implemented, and utility 1 for B being implemented. For voter 2, it is the other way around: voter 2 would receive utility 1 for A being implemented, and utility 2 for B being implemented. If neither proposal is implemented, both voters get utility 0. (All of this is common knowledge between the voters.)

a (8 points). Prove that strategies where voter 1 does not approve A are weakly dominated. (Similarly, strategies where voter 2 does not approve B are weakly dominated, but you do not need to prove this too.)

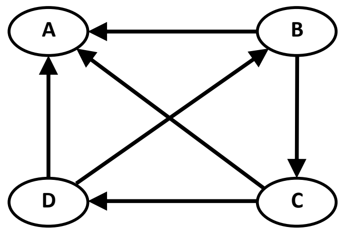
The above leaves only two strategies per voter in the game (do you, or do you not, approve your *less* preferred alternative as well?).

b (8 points). Write this out as a normal-form game with the (expected) utilities for each voter. For each voter, call one strategy "one" (for approving only that voter's preferred proposal) and the other "both" (for approving both).

c (14 points). Find/compute all the Nash equilibria of this game. (Hint: of which game that we studied in class does this game remind you?)



(a) The structure of the Cup (a balanced binary tree), with an initial assignment of candidates to leaves and the resulting outcome.



(b) The pairwise election outcomes among (A)loofa, (B)righta, (C)orrupta, and (D)iligenta. (The votes from which these pairwise election outcomes result are not given, and not needed for this problem.)

Figure 1: A Cup election in Opaquia as organized by SNEAC.

Problem 4: Investigating the scheduler's control over Cup voting rule outcomes (30 points).

In the country Opaquia, elections are run using the Cup voting rule. (Recall that under the Cup voting rule, we create a single-round elimination tournament, such as we often see in sports tournaments: pairs of candidates face each other, and the winner of their pairwise election – the one preferred by more voters – proceeds to the next round. In the end, only one candidate remains, and that one wins.) In Opaquia, the structure of the Cup is clear (say, a balanced binary tree as in Figure 1a), and the results of the pairwise elections can be trusted. However, only the people on the Superior National Elections Advisory Committee (SNEAC) know how it is decided which candidate is assigned to which position in the Cup (i.e., the initial assignment of candidates to leaves, i.e., to bottom nodes of the tree).

Intrepida is an Opaquian reporter. She is suspicious of this arrangement, and of the power of SNEAC in general. She starts to investigate which other candidates could have won, with different assignments of candidates to leaves.

For example, consider the pairwise elections graph in Figure 1b, where if x defeats y in their pairwise election, there is an edge from x to y. (There are no weights on the edges, because they do not matter for the Cup rule.) SNEAC has decided that the candidates are assigned to leaves as follows from left to right: Aloofa faces Corrupta in the first round, and Brighta faces Diligenta in the first round. Corrupta defeats Aloofa and Diligenta defeats Brighta, so Corrupta and Diligenta go on to the next round (they arrive at the next vertices of the tree). Then, in the "final" round, Corrupta defeats Diligenta, so that Corrupta arrives at the root of the tree, and wins the election – as is displayed in Figure 1a.

a (10 points). Which of the other candidates (A, B, D) could have won with a different assignment of candidates to leaves? For each of these candidates, **show** an assignment of candidates to the leaves, and how the Cup proceeds, so that that candidate wins. For every other candidate, **give** a very brief argument why this candidate cannot win for any assignment.

b (20 points). Give an integer program for determining whether a given candidate can win (i.e., wins for some assignment of the candidates to leaves). Actually, the key here is to choose an assignment of candidates to *all* the vertices of the tree in a way that makes for a sensible election result; you must find constraints that make sure it all makes sense. For this question you only need to find those constraints.
You should have a binary variable $x(v, i)$ for every pair of a vertex v and a candidate i, indicating whether that candidate is at that vertex.
You should have a binary parameter $l(v)$ that indicates whether v is a leaf vertex (is at the bottom of the tree).
You should have a binary parameter $c(v_1, v_2)$ that indicates whether v_1 is a child of v_2 (i.e., if the candidate at v_1 wins that round then that candidate would arrive at v_2 next). (Note we use i for candidates to prevent confusion with the child relation.)
You should have a binary parameter $d(i_1, i_2)$ that indicates whether candidate i_1 defeats candidate i_2 in their pairwise election. (You may assume that there will be no pairwise ties; also, $d(i, i) = 0$ for all i, i.e., candidates don't defeat themselves.)

You should have a constraint that every vertex has exactly one candidate assigned to it.
You should have a constraint that every candidate is assigned to exactly one leaf vertex.
You should also have a constraint that says that for every vertex v and every candidate i, at least one of the three following things is true: (1) i is not assigned to v, (2) v is a leaf vertex, or (3) the following two things are both true: (a) i is assigned to a child of v, and (b) there is a candidate that i defeats and that is assigned to a child of v. (This is the hardest part. In writing this constraint, it makes sense to start with writing a constraint for (3), and then add things to it to create exceptions for cases (1) and (2).)

(At this point it would be easy to add for example an objective that a given candidate should be assigned to as many vertices as possible, but you don't have to do this, so that we don't have to introduce even more notation.)

```
set VERTICES;
set CANDIDATES;
var x{v in VERTICES, i in CANDIDATES}, binary;  % is i assigned to v
param l{v in VERTICES};  % is v a leaf vertex, i.e., a vertex at the bottom (binary)
param c{v1 in VERTICES, v2 in VERTICES};  % is v1 a child of v2 (binary)
param d{i1 in CANDIDATES, i2 in CANDIDATES};  % does i1 defeat i2 in their
pairwise election (binary)
```