Name: ______________________________

Roll Number: _______________

1. A robber is trying to escape a cop on an undirected graph G. In the beginning, the cop is at a vertex s and the robber is at a vertex t. (You may assume that s and t are distinct.) They take turns making moves, and each knows the location of the other at all times. A move (by either of them) consists of either staying at the current vertex or moving to a neighbouring one.

   The cop is boastful, so he announces his moves before making them. Specifically:

   a. **before** anyone makes a move, the cop's first move is announced - so the robber knows where the cop is headed.

   b. Then, the robber makes an actual move.

   c. After this, each time the cop moves, he must respect the previous announcement (i.e, move to the previously announced vertex), and then decide his next move and announce it.

   d. The robber hears the announcements, so she always knows the cop's next move before making her own. She makes her move.

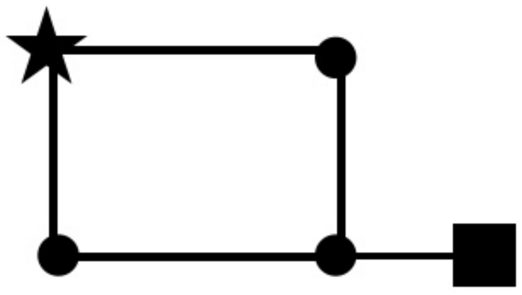
   If the cop and the robber are at the same vertex after either of them moves, then the robber is caught. Otherwise, the chase is on!

   The robber chooses her moves optimally to escape. If she cannot escape, she chooses her moves to maximize the total number of moves until she is caught. The cop chooses his moves optimally to try to catch the robber in as few total moves as possible.

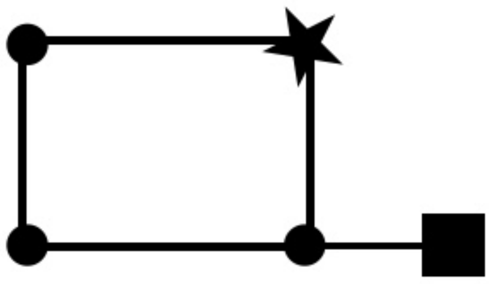
   Given the graph's layout and the initial locations of both the cop and the robber, find out whether robber will be caught by the cop and, if so, in how many moves. We say that the game is won by the robber if she's never caught, and by the cop otherwise.

   In the figures below, the square vertex depicts the initial location of the robber, and the star depicts the initial location of the cop. Indicate what happens under optimal play. If you choose that the cop wins, indicate how many moves the game lasts assuming optimal play. Each move made by each player counts as a distinct move.

   1A. **[2 marks]** Who wins? The Robber



1B. **[2 marks]** Who wins? The Cop



1C. **[2 marks]** If the robber starts on a vertex that is a part of a cycle, then which of the following statements is true?

- ○ The robber wins this game.
- ○ The cop wins this game and the number of moves is equal to the length of the cycle.
- ○ The cop wins this game and the number of moves is twice the length of the cycle.
- ○ The cop wins this game and the number of moves depends on the initial distance between the cop and the robber.
- ○ The outcome depends on where the cop starts.

> The robber can always move in the opposite direction of the cop.

1D. **[3 marks]** Suppose the game is being played on a path (i.e, a graph with vertices $u_1, \dots, u_n$ and edges $(u_1, u_2), (u_2, u_3), \cdots, (u_{n-1}, u_n)$. Suppose the cop starts at u_1 and the robber starts at u_n. Which of the following statements is true?

- ○ The robber wins this game.
- ○ The cop wins this game and the number of moves is n.
- ○ The cop wins this game and the number of moves is $2n$.
- ○ The cop wins this game and the number of moves is $2n - 1$.
- ○ The cop wins this game and the number of moves is $2(n - 1)$.

> The cop wins this game. By moving towards u_n, the cop can decrease its distance to the robber by at least one in each move, the best the robber can do is to not move at all. The number of moves is $2(n-1)$. For example, if $n = 3$, the moves are the following:
>
> - the cop announces a move to u_2.
> - the robber does nothing [1 move].
> - the cop moves to u_2 [1 move].
> - the cop announces a move to u_1.
> - the robber does nothing [1 move].

> - the cop moves to u_1 [1 move].
> - GAME OVER

1E. **[3 marks]** Suppose the graph G has a cycle on the vertices $v_1, v_2, \ldots, v_k$ and these are the *only* vertices that belong to any cycle in G. The robber is initially on a vertex u and the closest vertex on the cycle is v_1, via the path $(u, p), (p, q), (q, v_1)$. The cop is initially on a vertex w and the closest vertex on the cycle is v_n, via the path $(w, r), (r, v_n)$. Which of the following statements is true? Assume there are no other vertices in the graph G.

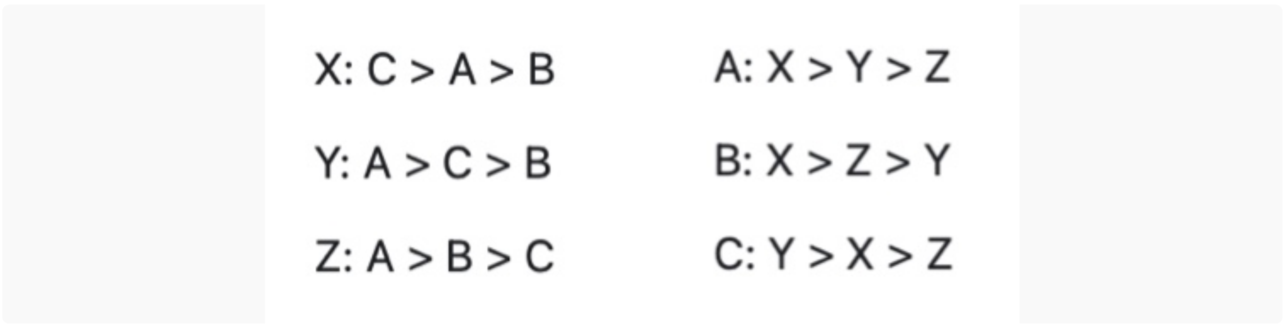
- The robber wins this game.
- [✓ The cop wins this game.]

Explain your answer: if you think the robber wins the game, explain how the robber will evade the cop forever, and if you think the cop wins this game, explain what is the sequence of moves in an optimal game. (You can use the space on the next page.)

> Sequence of moves in optimal play:
>
> Move 1: Robber stays at u
>
> Move 2: Cop moves from w to r
>
> Move 3: Robber stays at u
>
> Move 4: Cop moves from r to v_n
>
> Move 5: Robber stays at u
>
> Move 6: Cop moves from v_n to v_1
>
> Move 7: Robber stays at u
>
> Move 8: Cop moves from v_1 to q
>
> Move 9: Robber stays at u
>
> Move 10: Cop moves from q to p
>
> Move 11: Robber stays at u
>
> Move 12: Cops moves from p to u

---

2. Consider a stable marriage instance with A,B,C being the men and X,Y,Z being the women. The input is the following:



X: C > A > B

Y: A > C > B

Z: A > B > C

A: X > Y > Z

B: X > Z > Y

C: Y > X > Z

2A. **[2 marks]** What is the output of the stable matching algorithm for this instance? Assume that the men are proposing.

Round 1. A and B propose to X, C proposes to Y. C and Y are engaged, X and A are engaged.

Round 2. B proposes to Z, Z accepts, we are done.

C ⇔ Y; A ⇔ X; B ⇔ Z

2B. **[3 marks]** Consider again the algorithm where men are proposing. One of the women can misreport her preferences to get a better outcome from this algorithm. Identify the woman and explain what preference she can submit instead of her true preference to improve the output from her perspective.

> Suppose X changes her preference to C > B > A
>
> Then when A and B propose to X, instead of rejecting B, X will reject A.
>
> Now X and B are engaged, as are C and Y (as before).
>
> → X ⇔ B, Y ⇔ C
>
> A proposes to Y, who breaks her engagement with C to be engaged to A.
>
> → X ⇔ B, Y ⇔ A
>
> C now proposes to X, and X breaks off her engagement with B to accept C.
>
> → X ⇔ C, Y ⇔ A
>
> Finally, B proposes to Z and is accepted.
>
> → X ⇔ C, Y ⇔ A, Z ⇔ B
>
> Note that X is better off with C than with A now.

---

3. **[5 marks]** When an array is to be sorted, it may happen that some data values start out being in the same position where they should end up. For example, in the array which is originally:

$$45, -4, 32, 0$$

the 32 is right where it will be in the final sorted output:

$$-4, 0, 32, 45$$

But as a particular sorting algorithm operates, it might (depending on the algorithm) move such an element out of the position where it belongs and move it back eventually.

Let's say that a sorting algorithm **respects fixedpoints** if it never moves an element that is in its proper position, on any input.

Consider the following methods of sorting:

Selection sort. The algorithm divides the input list into two parts: a sorted sublist of items which is built up from left to right at the front (left) of the list and a sublist of the remaining unsorted items that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

Eg: **4** 3 2 **1** → 1 **3** **2** 4 → 1 2 3 4

Insertion sort. Insertion sort iterates over the array, consuming one input element each repetition, and grows a sorted output list. At each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain.

Eg: 4 **3** 2 1 → 3 4 **2** 1 → 2 3 4 **1** → 1 2 3 4

Which of the following statements are true?

- [✓ Insertion sort does not respect fixedpoints but selection sort does.]
- ○ Selection sort does not respect fixedpoints but insertion sort does.
- ○ Neither insertion sort nor selection sort respects fixed points.
- ○ Both insertion sort and selection sort respect fixed points.

Justify your answer. If you claim that a particular sorting method does not respect fixed points, then give an example. If you claim that an algorithm does respect fixed points, argue why.

> **Insertion sort does not respect fixed points.**
>
> Consider: 3 2 1 → 2 3 1 → 1 2 3
>
> **Selection sort does respect fixed points.**
> Consider an element x at it's correct position, say index t. Then x is the t-th smallest number. Observe that for all iterations before and after the t-th iteration, by definition x is not involved in any swaps. In the t-th iteration, the smallest element in the unsorted sublist and the leftmost unsorted element coincide at x, so the element x does not move from its position in this round either.

---

4. You have distributed M objects among N children. The set of objects given to a child is called his or her **bundle**.

   Each child has a specific value for their bundle: let us say child k has value v_k for their bundle.

   Each child also has a value for all the other bundles: so let us say that child k has value $v_{k,\ell}$ for the bundle that was given to child ℓ.

   We say that child a **is jealous** of child b if $v_{a,b} > v_a$, i.e, s/he values the bundle given to b more than the bundle that s/he has.

   Consider the following directed graph G. Introduce one vertex for every child, and add an edge from a to b if a is jealous of b.

   4A. **[2 marks]** Suppose G has a directed cycle $u_1 \to u_2 \to \cdots \to u_q \to u_1$. Describe a way to reassign the bundles (without changing them) so that with the new assignment, all the edges in the cycle disappear (i.e, there is no jealousy between u_1 and u_2, between u_2 and u_3, and so on, with respect to the *new* assignment). Explain your answer on the next page.

> Swap all bundles counterclockwise along the cycle: let u_1 get its bundle from u_2, u_2 get its bundle from u_3, and so on. Suppose u_1 is still jealous of u_2. Then, repeat this process. Go on till u_1 is not jealous of u_2. Notice that u_1 will also not be jealous of u_2 by definition of when we stop; and u_1 will not be jealous of u_q because u_q has u_1's old bundle, that u_1 valued less than its current bundle. This will lead to a situation where you would have deleted two edges from the given directed cycle. Notice that you will not get stuck (i.e, you will not have to repeat the process forever): why?
>
> However, it may not be possible to completely eliminate all the edges between the agents of this cycle. It is a fun puzzle to figure out an input for which an original cycle edge persists in *any reassignments of bundles*. So this question was unintentionally too demanding! The intended task was to just make the cycle disappear (by eliminating at least one edge from it), but the clarifying description mistakenly asked for more. Full marks will be awarded to all students, and bonus marks to those who made any reasonable attempt at reducing envy.

4B. **[1 marks]** Suppose G has no directed cycles. Is it true that there is a child who is not jealous of anyone?

○ [✓ Yes] ○ No ○ Impossible to conclude from the given information

> Start anywhere in G and keep taking an out-edge to walk around in the graph. Notice that you will never repeat a vertex in your walk (since G has no directed cycles). When you get stuck in your walk (which must happen, at some point you will run out of vertices since you cannot repeat them), you must be at a vertex that has no out-edge (indeed, this is why you got stuck!). The agent that this vertex represents is not jealous of anyone.

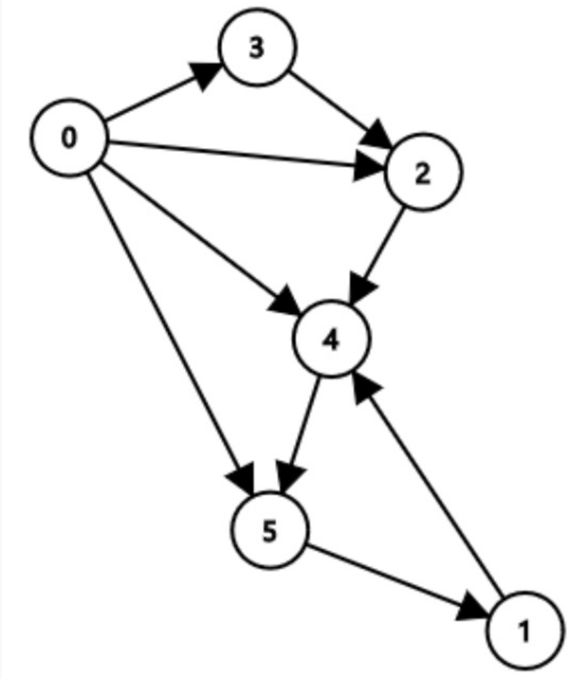
4C. **[1 marks]** Suppose G has no directed cycles. Is it true that there is a child who is nobody is jealous of?

○ [✓ Yes] ○ No ○ Impossible to conclude from the given information

> Same as the previous answer, except you walk backwards (taking in-edges in each step).

---

5. **[2 marks]** In the graph below, what is the smallest number of edges you need to add to make the graph *strongly* connected? Recall that a strongly connected graph is one where there is a path from u to v for any pair of vertices u and v.



> Adding a single edge is enough. For example, suppose you add the edge from 1 to 0. Then you get the following cycle: 1 → 0 → 3 → 2 → 4 → 5 → 1. Any valid answer with one edge gets full credit.
>
> - Any valid answer with more than one edge (for example, a suggestion to add two edges that do make the graph strongly connected) gets partial credit.
> - Those who have answered "1" with no further explanation will also get full credit.
> - Those who have answered with a number greater than one with no further explanation get no credit.
> - Those who have answered with a number greater than one with an incorrect explanation (for example, a suggestion to add two edges that does not work) get no credit.

---

6. **[2 marks]** The following is true for n guests at a party:
   - In any group of three guests, there are two guests who do not know each other, and
   - In any groups of seven guests, there are two guests who do know each other.

   At the end of the party, everyone gives a present to all the guests he or she knows.

   Prove that the total number of gifts given is at most $6n$.

   Hint: what can you say about the maximum degree of this graph?

   > [guests ⇔ vertices and knowing each other ⇔ edges]
   >
   > Claim: the maximum degree of this graph is 6.
   >
   > If a vertex v has degree 7 or more, then there are two neighbors of v that have an edge between them; say u and w. But then u v w form a triangle, which is not allowed.
   > Now #of gifts given = 2*#edges = sum of degrees $\leq$ 6n.