Midterm 1

Replace by cover page.

Replace by conduct during examinations page.

WRITE UGRAD IDs HERE (**-1 mark** if any missing or incorrect; use only the boxes you need)

UGRAD ID #1: ☐ **UGRAD ID #3:** ☐

UGRAD ID #5: ☐

UGRAD ID #2: ☐ **UGRAD ID #4:** ☐

NOTE: Throughout the exam a *simple* graph is an undirected, unweighted graph with no multiple edges (i.e., no exact repeats of the same edge) and no self-loops (i.e., no edges from a vertex to itself). Graphs are simple unless stated otherwise, and even where we explicitly contradict one of these, the rest remain true. So, for example, a "directed graph" with no other information specified would be unweighted with no multiple edges and no self-loops.

1 O'd to a Pair of Runtimes [6 marks]

The pairs of functions below represent algorithm runtimes on a pair of lists of length m and n, with $1 < m < n$. For each pair, fill in the circle next to the best choice of:

LEFT: the left function is big-O of the right, i.e., left $\in$ O(right)

RIGHT: the right function is big-O of the left, i.e., right $\in$ O(left)

SAME: the two functions are Θ of each other, i.e., left $\in$ Θ(right)

INCOMPARABLE: none of the previous relationships holds for all allowed values of n and m.

Do not choose **LEFT** or **RIGHT** if **SAME** is true. The first one is filled in for you.
[2 marks per answer]

Left Function	Right Function	Answer
n	n^2	**LEFT**
$n \lg m$	$m \lg n$	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$\frac{n^2+m}{m}$	$\frac{m}{\lg m}$	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE
$n + m \log m$	m^2	◯ LEFT ◯ RIGHT ◯ SAME ◯ INCOMPARABLE

2 eXtreme True And/Or False [10 marks]

Each of the following problems presents a scenario and a statement about that scenario. For each one, fill the circle next to the correct one among the three choices:

- The statement is **ALWAYS** true, i.e., true in *every* instance matching the scenario.
- The statement is **SOMETIMES** true, i.e., true in some instance matching the scenario but false in another instance matching the scenario.
- The statement is **NEVER** true, i.e., true in *none* of the instances matching the scenario.

Then, **briefly justify** your answer as follows:

- Justify an **ALWAYS** answer by giving a small instance that fits the scenario for which the statement is **true**, and then briefly sketching the key points in a proof that the statement is **true** for all instances that fit the scenario.
- Justify a **NEVER** answer by giving a small instance that fits the scenario for which the statement is **false**, and then briefly sketching the key points in a proof that the statement is **false** for all instances that fit the scenario.
- Justify a **SOMETIMES** answer by giving **two** small instances that fit the scenario: one for which the statement is **true** and one for which the statement is **false**. (Indicate which one is which!)

Here are the problems: **[5 marks per problem]**

1. **Scenario:** A directed, weakly-connected graph with $n \geq 2$ vertices, m edges and at least two vertices in the same strongly connected component.

   Statement: $m \geq n$.

   ◯ **ALWAYS**
   ◯ **SOMETIMES**
   ◯ **NEVER**

   True instance (always/sometimes) *or* **proof that statement is false in all instances** (never):

   False instance (sometimes/never) *or* **proof that statement is true in all instances** (always):

2. **Scenario:** An STP (SMP but allowing ties in preference lists) instance in which the preference list of w_1 may contain ties but **no** other preference list contains ties. **Statement:** Gale-Shapley run with **men** proposing on an SMP instance produced by breaking w_1's ties arbitrarily will produce the same result no matter how the ties are broken.

   ◯ **ALWAYS**
   ◯ **SOMETIMES**
   ◯ **NEVER**

   True instance (always/sometimes) *or* **proof that statement is false in all instances** (never):

   False instance (sometimes/never) *or* **proof that statement is true in all instances** (always):

   The rest of the page is intentionally blank.

   If you write answers below, CLEARLY indicate here what question they belong with AND on that problem's page that you have answers here.

3 Oh Oh Oh Oh Oh, Try Everything! [9 marks]

In this problem, we'll investigate what happens when the preconditions of some algorithms are violated.

1. Consider the following `colorize` algorithm from Assignment #2:

```
// G = (V, E) is an undirected graph, represented as an adjacency list
function colorize(V, E)
    n = |V|
    m = |E|
    let Colors be a list of at least n distinct colors
    for i = 0 to n - 1:
        V[i].setvisited(False)

    for i = 0 to n - 1:
        DFS(V[i], (function(v): v.setcolor(Colors[i])))
```

where DFS is defined as:

```
function DFS(v, visitfunction)
    if not v.getvisited():
        visitfunction(v)
        v.setvisited(True)
        for w in neighbours(v):
            DFS(w, visitfunction)
```

When called on an undirected graph, `colorize` colours all vertices in each connected component in the same colour unique to that component. Which of these best completes the following statement about what `colorize` does when called on a **directed** graph? Fill in the circle next to the **best** answer. **[2 marks]**

When called on a **directed** graph, `colorize` colours all vertices in each...

◯ ... connected component in the same colour unique to that component.
◯ ... strongly connected component in the same colour unique to that component.
◯ ... strongly connected component in the same color but not necessarily unique to that component.
◯ None of these.

2. Now, consider the following `intersection2` algorithm from Assignment #2:

```
function intersection2(X, Y)
    Z = { }
    i = j = 0
    m = length(X)
    n = length(Y)
    while i < m and j < n:
        if X[i] < Y[j]:
            i = i + 1
        elsif X[i] > Y[j]:
            j = j + 1
        else:
            add X[i] to Z
            i = i + 1
            j = j + 1
```

With the precondition that `X` and `Y` are both sorted in increasing order, this algorithm computes the intersection of `X` and `Y` interpreted as sets.

In this problem, we will instead consider the case where $n > 0$, $m = n$, `X` is sorted, `Y` is **not necessarily sorted**, and `X` and `Y` **contain the same elements** (ignoring order).

(a) Give exact lower- and upper-bounds on the number of elements in the set `intersection2` returns. Fill in the circle next to **best** answer for each bound. **[2 marks]**

Lower-bound:
- ◯ 0
- ◯ 1
- ◯ $\lceil \frac{n}{2} \rceil$
- ◯ n

Upper-bound:
- ◯ 0
- ◯ 1
- ◯ $\lceil \frac{n}{2} \rceil$
- ◯ n

(b) Now, give an example with $n = 3$ using your choice of the values 1, 2, 3, and 4 (but no other values) that achieves your **lower-bound**: **[3 marks]**

X = ☐ Y = ☐

3. Give a good big-O bound on the runtime of `intersection2` called on a sorted list `X` and a possibly unsorted list `Y` of length m and n, respectively. (We have removed the restrictions that $n = m$ and that `X` and `Y` contain the same elements.) **[2 marks]**

Big-O bound: ☐

4 Date-A Centre [10 marks]

A cloud service matches processors with jobs to run on them. Each processor is assigned at most one job, each job is assigned to at most one processor, there are n processors $p_1, \ldots, p_n$ and n jobs $j_1, \ldots, j_n$, but not all processors can run all jobs. $m(i,k)$ is true if and only if processor p_i can run job j_k.[1]

The Cloud Matching Problem (CMP) is to produce a list M such that either $M[k] = X$ (meaning job j_k is not assigned to any processor), or $M[k] = i$ (meaning job j_k is assigned to processor p_i) and $m(i,k)$ is true. Further, in a good solution, each processor is assigned at most one job, and no unused processor can run any unassigned job.

1. Complete the following reduction from CMP to STP (the Stable Marriage Problem with ties). **[6 marks]**

   Starting with a CMP instance, construct an STP instance as follows:

   - Woman w_i is processor p_i.
   - Man m_i is [ ]
   - The preference list for w_i lists [ ]

     before [ ]
   - The preference list for m_i lists [ ]

     before [ ]

   Then, solve the STP instance to produce a solution S with no strong instabilities. Build a CMP solution by, for each pair (w_i, m_k) in S, setting $M[k]$ to:

   [ ]

[1] Perhaps because processor p_i has enough memory, cores, drive space, etc. for job j_k.

2. **Complete the men's and women's preferences** in the following instance to show that the reduction above combined with an STP solver that produces a perfect matching with no strong instabilities (i.e., no instabilities in which m_i and w_j *strictly* prefer each other to their assigned partners) is not enough to **guarantee** that the *largest possible number* of jobs is matched to processors that can run them. **[4 marks]**

$m(i, k)$ is defined by this table:

	p_1	p_2
j_1	$true$	$true$
j_2	$true$	$false$

$$m_1: \quad w_1 = w_2 \qquad w_1:$$
$$m_2: \qquad\qquad\qquad w_2:$$

Now **explain the counterexample**:

Without any strong instability in the STP solution, we can assign job j_1 to ◯ p_1 ◯ p_2 ◯ no processor and job j_2 to ◯ p_1 ◯ p_2 ◯ no processor which means that ◯ 0 ◯ 1 ◯ 2 job(s) can be completed. However a better solution would be to assign job j_1 to ◯ p_1 ◯ p_2 ◯ no processor and job j_2 to ◯ p_1 ◯ p_2 ◯ no processor so that ◯ 0 ◯ 1 ◯ 2 jobs can be completed.

5 Not Just for Practice Anymore [5 marks]

Solve these variations on ungraded problems from Assignment #1:

1. Recall that STP is the Stable Marriage problem except that ties are allowed in preference lists. Also recall that a *strong instability* in a perfect matching consists of a woman w and a man m such that w and m both (strictly) prefer each other to their current partners.

   Describe a way to create an STP instance of an arbitrary size n that has exactly $(n-1)!$ solutions without strong instabilities, **briefly** justifying why it has the right number of solutions. **[3 marks]**

2. Imagine that we have a set of n employees. Each one is required to have exactly one mentor chosen from within the same set, but employees can have any number of mentees (zero or more). We can consider any such arrangement to be a directed graph, with each employee as a vertex and an edge from each employee to their mentor.

   Which of these is true about any such mentorship graph that follows these rules? Fill in the box next to **every** true statement. **[2 marks]**

   ☐ The graph has at most one cycle.
   ☐ The graph has at least one cycle.
   ☐ Every vertex in the graph is part of a simple cycle.
   ☐ The graph is strongly connected.

6 BONUS: From the Cutting Room Floor [1 BONUS marks]

Bonus marks add to your exam and course bonus mark total but are **not** required. **WARNING**: These questions are too hard for their point values. We are free to mark these questions harshly. Finish the rest of the exam before attempting these questions. Do not **taunt** these questions.

1. **RECALL** from Assignment #2: For its Blue-and-Gold fundraising campaign, UBC has found *anchors*—major donors, each with a *limit* (maximum amount they will donate)—and *causes* to which the anchors will donate. Each cause has a *goal*, the maximum money that will go to the cause. **An anchor donates to at most one cause**, but **one cause may receive donations from many anchors**, and the total amount of the donations of anchors to a cause cannot exceed that cause's goal. The Blue-and-Gold Problem, or BGP, consists in determining how much money each anchor will give to each cause, subject to the constraints stated above. A BGP instance's solution is the maximum dollar amount that can be raised. (Assume limits are distinct, as are goals.)

   Consider the following greedy algorithm for this problem:

   (a) Sort the limits in decreasing order.

   (b) For each limit L starting with the largest:

      Find the cause with the least money remaining toward its goal (where the amount currently remaining is G) such that $L \leq G$.
      If such a cause exists, match L with that cause, and decrease that cause's remaining goal by the minimum of L and G.
      Otherwise, find the cause with the largest remaining goal (which may be 0), match L with that cause, and set that cause's remaining goal to 0.

   Is this algorithm optimal for this problem?

   ◯ Yes
   ◯ No

   If you answered yes, give a clear, concise, and **complete** proof of your answer (i.e., not just a proof sketch). If you answered no, give a **complete** (but small) counterexample to its correctness (i.e., an instance and also the optimal solution, greedy solution, and clear explanations of each).

This page intentionally left (almost) blank.

If you write answers here, you must CLEARLY indicate on this page what question they belong with AND on the problem's page that you have answers here.