Duration: **50 minutes**
Aids Allowed: **NONE** (in particular, no calculator)

Student Number: |_|_|_|_|_|_|_|_|_|

Last (Family) Name(s): ____________________

First (Given) Name(s): ____________________

---

*Do **not** turn this page until you have received the signal to start.*
In the meantime, please read the instructions below *carefully*.

This term test consists of 2 questions on 10 pages (including this one), printed on both sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete, fill in the identification section above, write your student number where indicated at the bottom of every odd-numbered page (except page 1), and write your name on the back of the last page.*

Answer each question directly on the test paper, in the space provided, and use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and *indicate clearly the part of your work that should be marked.*

In your answers, you may use without proof any result or theorem covered in lectures, tutorials, homework, tests, or the textbook, as long as you give a clear statement of the result(s)/theorem(s) you are using. You must justify all other facts required for your solutions.

Write up your solutions carefully! In particular, use notation and terminology correctly and explain what you are trying to do — part marks *will* be given for showing that you know the general structure of an answer, even if your solution is incomplete.

If you are unable to answer a question (or part), you will get 20% of the marks for that question (or part) if you write "I don't know" and nothing else — you will *not* get those marks if your answer is completely blank, or if it contains contradictory statements (such as "I don't know" followed or preceded by parts of a solution that have not been crossed off).

Marking Guide

# 1:	______/14
# 2:	______/19
TOTAL:	______/33

Use this page for rough work — clearly indicate any section(s) to be marked.

CONT'D...

Question 1. [14 MARKS]

You are consulting for a trucking company that does a large amount of business shipping packages from Toronto to Montréal. The volume is high enough that they have to send a number of trucks each day between the two locations. Trucks have a fixed integer limit $W > 0$ on the maximum amount of weight they are allowed to carry. Boxes arrive at the Toronto loading platform one by one, and each package i has a positive integer weight $w_i \leqslant W$. The loading platform is quite small, so at most one truck can be at the platform at any time. Company policy (and the limited space available at the platform) requires that boxes are shipped in the order they arrive—there would be complaints if boxes made it to Montréal out of order!

At the moment, the company is using a simple greedy algorithm for packing: they pack boxes in the order they arrive (as required), putting as many boxes as possible in each truck without exceeding the weight limit W, *i.e.*, whenever the current box would make the load exceed the weight limit W, they send the truck on its way and keep the box for the next truck. But they wonder if they might be using too many trucks, and they want your opinion on whether the situation can be improved: maybe one could decrease the number of trucks needed by sometimes sending off a truck that was less full, and in this way allow the next few trucks to be better packed? More formally, the current algorithm used by the company can be written down as follows.

```
TruckPacking($W, w_1, w_2, \ldots, w_n$):
    $k = 1$          # current truck number
    $T_k = [w_1]$    # list of packages in current truck
    $t = w_1$        # total weight of current truck
    $S = [T_k]$      # current partial solution
    for $i = 2, \ldots, n$:
        if $t + w_i \leqslant W$:   # package $i$ fits in current truck; put it in
            $T_k$.append($w_i$)
            $t$ += $w_i$
        else:                       # package $i$ does not fit in current truck; start next one
            $k$ += 1
            $T_k = [w_i]$
            $t = w_i$
            $S$.append($T_k$)
    return $S$    # $= [T_1, T_2, \ldots, T_k]$
```

Following the structure given in class, give a detailed proof that the current algorithm is guaranteed to always use the minimum number of trucks. In particular, give a clear, precise definition of what it means for a partial solution to be "promising" for this problem. (Your solution will be marked on its structure as well as its content.)

Define partial solution:
S_i — the partial solution, assignments of package $w_1, \ldots, w_i$

S_i is optimal if $\exists$ OPT solution s.t. the assignment of packages are the same as S_i on $w_1, \ldots, w_i$

[There's more space on the next page. . .]

B.C: $i=1$, $T_1 = [w_1]$
for OPT, it is same

I.H. assume for S_i, $\exists$ OPT S_i'

I.S. WTP S_{i+1}, $\exists$ OPT S_{i+1}'

For W_{i+1}

① $T_k + W_{i+1} \leq W$

So $w_{i+1} \rightarrow T_k$
# of Truck stays, which is OPT

② $T_k + W_{i+1} > W$
Put on truck T_{k+1}

$\Rightarrow$ optimal

$\Rightarrow S_1, \ldots, S_n$ — optimal

CONT'D...

$\Rightarrow$ 1 optimal solution

Question 1. (continued)

[Use this page for the rest of your solution.]

Use this page for rough work — clearly indicate any section(s) to be marked.

CONT'D...

Question 2. [19 MARKS]

An "ordered graph" $G = (V, E)$ is a **directed** graph whose nodes can be ordered $v_1, v_2, \ldots, v_n$ such that:

(i) every edge goes from a lower indexed node to a higher indexed node, *i.e.*, $i < j$ for all $(v_i, v_j) \in E$;
(ii) all nodes except v_n have at least one outgoing edge.

Given an ordered graph $G = (V, E)$, we want to find a longest path from v_1 to v_n, *i.e.*, a path with the maximum number of edges. For example, in the graph below, (v_1, v_2), (v_2, v_4), (v_4, v_5) is a longest path of length 3.

$$\begin{tikzcd}
v_1 \arrow[r] \arrow[rrr, bend left] & v_2 \arrow[rr, bend right] \arrow[rrr, bend right] & v_3 \arrow[r] & v_4 \arrow[r] & v_5
\end{tikzcd}$$

Part (a) [5 MARKS]

Give a simple greedy algorithm that attempts to solve this problem, on input $G = \big(V = \{v_1, v_2, \ldots, v_n\}, E\big)$ where the vertices have already been ordered to satisfy the conditions above. Then, show that your algorithm does not always return an optimal solution (give an explicit counter-example and desribe the solution returned by your algorithm, as well as why it is not optimal).

```
Greedy:
    R = {v1}
    pointer = v1
    While pointer ≠ vn :
        Find shortest vi for pointer:
            R += vi
            pointer = vi
    return sizeof(R) - 1
```

$$\begin{tikzcd}
v_1 \arrow[r] \arrow[rr, bend left] & v_2 \arrow[rrr, bend right] & v_3 \arrow[r] & v_4 \arrow[r] & v_5
\end{tikzcd}$$

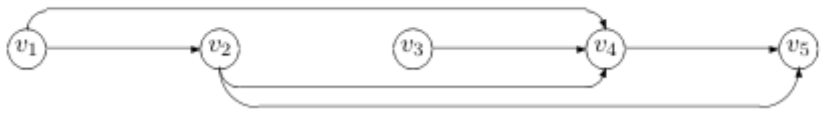
By Greedy, $\{v_1, v_2, v_5\}$ actual: $\{v_1, v_3, v_4, v_5\}$

Question 2. [19 MARKS]

An "ordered graph" $G = (V, E)$ is a **directed** graph whose nodes can be ordered $v_1, v_2, \ldots, v_n$ such that:

(i) every edge goes from a lower indexed node to a higher indexed node, *i.e.*, $i < j$ for all $(v_i, v_j) \in E$;
(ii) all nodes except v_n have at least one outgoing edge.

Given an ordered graph $G = (V, E)$, we want to find a longest path from v_1 to v_n, *i.e.*, a path with the maximum number of edges. For example, in the graph below, (v_1, v_2), (v_2, v_4), (v_4, v_5) is a longest path of length 3.



$L[i] =$ max edges from V_1 to V_i

$= \underset{\text{max}}{\{} 1 + L[j], \; e_{ji} \in E \}$

$L[i] = -\infty$ if no j that $e_{ji} \in E$

CONT'D...

Question 2. (CONTINUED)

Part (b) [14 MARKS]

Following the structure given in class, give a dynamic programming algorithm for this problem; state the running time of your algorithm. (Your solution will be marked on its structure as well as its content.)

On this page, please write nothing except your name.

Last (Family) Name(s): ______________________________

First (Given) Name(s): ______________________________