PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

DECEMBER 2014 EXAMINATIONS

CSC 373 H1 F

Duration — 3 hours

Examination Aids: One double-sided handwritten 8.5"×11" aid sheet.

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 final examination consists of 8 questions on 12 pages (including this one), printed on one side of the paper. *When you receive the signal to start, please make sure that your copy is complete and fill in the identification section above.*

Answer each question directly on the examination paper, in the space provided, and use the back side of the pages for rough work. If you need more space for one of your solutions, use the back 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 theorem or result covered in lectures, tutorials, problem sets, assignments, 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 of a question), you will get 10% of the marks for any solution that you leave *entirely blank* (or where you cross off everything you wrote to make it clear that it should not be marked).

Remember that, in order to pass the course, you must achieve a grade of at least 40% on this final examination.

MARKING GUIDE

Nº 1:	______/ 8
Nº 2:	______/ 7
Nº 3:	______/10
Nº 4:	______/10
Nº 5:	______/ 7
Nº 6:	______/ 5
Nº 7:	______/10
Nº 8:	______/ 8
BONUS:	______/ 5
TOTAL:	______/65

Good Luck!

Question 1. [8 MARKS]

Part (a) [2 MARKS]

Why is it customary to use an array when solving a problem with dynamic programming? Explain what the array is used for and what advantage is gained by using it.

Part (b) [2 MARKS]

Let A and B be decision problems. Show that if $A \leqslant_p B$ then $\overline{A} \leqslant_p \overline{B}$ (where $\overline{A}$ is the complement of A).

Part (c) [4 MARKS]

Show that if A is NP-complete and A is in $coNP$, then $NP = coNP$.

Question 2. [7 MARKS]

Write an algorithm that takes as inputs:

- a connected, undirected graph $G = (V, E)$ with positive integer edge weights $w(e)$, for all $e \in E$,
- a minimum spanning tree $T \subseteq E$ for G,
- a single edge $e_0 \in T$ such that $G_0 = (V, E - \{e_0\})$ is still connected,

and that constructs a minimum spanning tree T_0 for the graph $G_0 = (V, E - \{e_0\})$. In other words, we remove edge e_0 from graph G and we want to update T so the result is still a minimum spanning tree.

Your algorithm must run in worst-case time $\mathcal{O}(m + n)$ (where $n = |V|$, $m = |E|$). Write your algorithm in high-level pseudocode, then briefly explain why it is correct and runs within the required time bound.

Question 3. [10 MARKS]

Write a *dynamic programming* algorithm to solve the following SUBSETSUMSEARCH problem:

- **Input:** A set of positive integers $S = \{x_1, \ldots, x_n\}$ and a positive integer target t.
- **Output:** A subset $S' \subseteq S$ with sum exactly t $\left(\sum_{x \in S'} x = t\right)$, or NIL if there is no such subset.

Follow closely the outline given in class. In particular, justify your recurrence relation by describing the recursive structure of solutions, and argue that your algorithm runs in time polynomial in n and t.

Question 4. [10 MARKS]

Consider the SETPACKING decision problem ("SPD" for short) defined as follows.

- **Input:** A positive integer m, a collection of non-empty subsets $s_1, s_2, \ldots, s_n \subseteq \{1, 2, \ldots, m\}$, and a positive integer k.
- **Question:** Is there a set of indices $I \subseteq \{1, \ldots, n\}$ such that $|I| = k$ and no two subsets whose indices are in I have any element in common ($s_i \cap s_j = \varnothing$ for all $i, j \in I$ with $i \neq j$)?

Write a detailed proof that SETPACKING is NP-complete. (HINT: Consider the INDEPENDENTSET problem—think edges!)

Question 5. [7 MARKS]

Consider the MaxSetPacking optimization problem ("MSP" for short) defined as follows.

- **Input:** A positive integer m and a collection of non-empty subsets $s_1, s_2, \ldots, s_n \subseteq \{1, 2, \ldots, m\}$.
- **Output:** A set of indices $I \subseteq \{1, \ldots, n\}$ such that no two subsets whose indices are in I have any element in common ($s_i \cap s_j = \varnothing$ for all $i, j \in I$ with $i \neq j$), and $|I|$ is maximum.

Show that MaxSetPacking is polytime self-reducible (that is, MaxSetPacking $\xrightarrow{p}$ SetPacking).

Question 6. [5 MARKS]

Give an *integer program* to solve the MAXSETPACKING optimization problem defined in Question 5. State clearly the variables in your integer program, the objective function, and the constraints. Then show that solutions to your integer program correspond exactly to solutions to the MAXSETPACKING problem.

CONT'D...

Question 7. [10 MARKS]

Given a directed graph $G = (V, E)$ with two vertices $s, t \in V$, two paths from s to t are "edge-disjoint" if they share no common edge (though they may share one or more common vertex).

Use network flow techniques to write a polynomial-time algorithm that computes the *maximum* number of edge-disjoint paths from s to t in G. Explain carefully why your solution is correct and runs in polytime.

Question 8. [8 MARKS]

Recall the MINPACKETS optimization problem from Assignment 2 (below) and consider the algorithm on the right.

- **Input:** A positive integer size limit L and positive integer reply sizes $s_1, \ldots, s_n$.
- **Output:** A partition of $\{1, \ldots, n\}$ into packets $P_1, \ldots, P_k$ such that:
  - every reply belongs to exactly one packet;
  - each packet has total size at most L $\left(\sum_{j \in P_i} s_j \leqslant L\right)$;
  - the number of packets used (k) is minimum.

```
NEXTPACKET($s_1, \ldots, s_n, L$):
    $k \leftarrow 0$   # number of packets
    $\ell \leftarrow L$   # size of last packet
    for $i \leftarrow 1, \ldots, n$:
        if $\ell + s_i > L$:
            $k \leftarrow k + 1$
            $P_k \leftarrow \varnothing$
            $\ell \leftarrow 0$
        $P_k \leftarrow P_k \cup \{i\}$
        $\ell \leftarrow \ell + s_i$
    return $P_1, \ldots, P_k$
```

Part (a) [2 MARKS]

Prove that for any two consecutive packets P_i, P_{i+1} generated by NEXTPACKET, the sum of the replies in both packets is greater than L $\left(\sum_{j \in P_i \cup P_{i+1}} s_j > L\right)$.

Part (b) [2 MARKS]

Give a specific example where algorithm NEXTPACKET uses twice as many packets as necessary. State clearly the output produced by the algorithm, and the optimum solution.

Question 8. (CONTINUED)

Part (c) [4 MARKS]

Use the result from Part (a)—even if you did not solve it—to prove that algorithm NEXTPACKET has approximation ratio at most 2.

Bonus. [5 MARKS]

WARNING! This question is difficult and will be marked harshly: credit will be given only for making *significant* progress toward a correct answer. Please attempt this only *after* you have completed the rest of the final examination.

Can you find an algorithm to approximate the MINPACKETS problem (from Question 8) with a better approximation ratio than the NEXTPACKET algorithm (also from Question 8)? Justify your answer.

Use the space on this "blank" page for scratch work, or for any solution that did not fit elsewhere.
Clearly label each such solution with the appropriate question and part number.

Total Marks = 65