THE UNIVERSITY OF SYDNEY

CONFIDENTIAL EXAM PAPER

This paper is not to be removed from the exam venue.

Computer Science

EXAMINATION

Semester 1- Practice, 2020

COMP2123/9123 Data Structures and Algorithms

EXAM WRITING TIME: 2 hours
READING TIME: 10 minutes

EXAM CONDITIONS:
This is a CLOSED book examination.
All submitted work must be **done individually** without consulting someone else's solutions, or extenal resources like the Internet or a textbook, in accordance with the University's "Academic Dishonesty and Plagiarism" policies.

MATERIALS PERMITTED IN THE EXAM VENUE:

One A4 sheet of handwritten and/or typed notes double sided

MATERIALS TO BE SUPPLIED TO STUDENTS:
Nothing

INSTRUCTIONS TO STUDENTS:
Type your answers in your text editor and submit via Canvas. Handwritten responses will **not** be accepted.

Start by typing you student ID at the top of the first page of your submission. Do **not** type your name.

Submit only your answers to the questions. Do **not** copy the questions.

For examiner use only:

Problem	1	2	3	4	Total
Marks					
Out of	10	10	20	20	60

Problem 1.

a) Analyze the time complexity of this algorithm. [3 marks]

```
def Compute(A)
    result ← 0
    for i = 0; i < n; i++ do
        if A[i] > i then
            result ← result + A[i]
    return result
```

b) Solve the following recursion: [3 marks]

$$T(n) = \begin{cases} T(n/2) + O(1) & \text{for } n > 1 \\ O(1) & \text{for } n = 1 \end{cases}$$

c) We are planning a board games event and we're using one of the shelves in my office to store the games. Unfortunately the shelf only has a certain amount of space S, so we need to carefully pick which games we want to bring. Every game takes some space s_i and has a fun factor f_i that indicates how much fun it is to play that game (for $1 \leq i \leq n$). [4 marks]

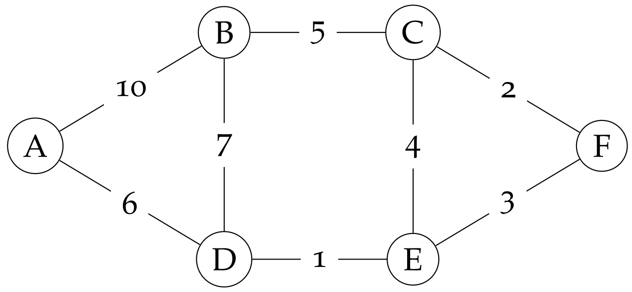
We want to maximize the amount of fun we'll have, so we want to maximize the sum of the fun factors of the games we pick (i.e., $\max \sum_{\text{picked game } i} f_i$), while making sure that the games fit on my shelf, so the sum of the space the games we pick take should be at most S (i.e., $\sum_{\text{picked game } i} s_i \leq S$). For simplicity, you can assume that all f_i, s_i, and S are all distinct positive integers.

The strategy of PickLargest is to always pick the game with the highest fun factor until my shelf is full: it sorts the games by their fun factor f_i in decreasing order and adds a game when its required space is less than the remaining space on the shelf.

```
def PickLargest(all f_i and s_i, S)
    currentSpace ← 0
    currentFun ← 0
    Sort games by f_i and renumber such that f_1 ≥ f_2 ≥ ... ≥ f_n
    for i ← 1; i ≤ n; i++ do
        if currentSpace + s_i ≤ S then
            currentSpace ← currentSpace + s_i        ▷ Pick the ith game
            currentFun ← currentFun + f_i
    return currentFun
```

Show that PickLargest doesn't always return the correct solution by giving a counterexample.

Problem 2. Consider the following edge weighted undirected graph G:



Your task is to:

a) Compute a minimum spanning tree T of G. List the edges in T. [7 marks]

b) Indicate the order in which the edges are added by Kruskal's algorithm. [3 marks]

(You do **not** have to explain your answer.)

Problem 3. Consider the *Dynamic Matrix* ADT for representing an matrix $A = \{a_{i,j}\}_{i,j=1}^{n}$ that supports the following operations:

- CREATE(): creates a 1×1 matrix where $a_{1,1} = 0$.
- SET/GET(i, j): set or get the value of the entry $a_{i,j}$.
- INCREASE-SIZE: If the current size of the matrix is $n \times n$, increase it to $n+1 \times n+1$ such that the new entries are set of 0. In other words, A becomes A' such that $a'_{i,j} = a_{i,j}$ if $1 \leq i, j \leq n$, and $a'_{i,j} = 0$ otherwise.

Your task is to come up with a data structure implementation for the Dynamic Matrix ADT that uses $O(n^2)$ space, where n is the size of the matrix, and CREATE, SET, GET take $O(1)$ and INCREASE-SIZE takes $O(n)$ time. Remember to:

a) Describe your data structure implementation in plain English. [8 marks]

b) Prove the correctness of your data structure. [7 marks]

c) Analyze the time and space complexity of your data structure. [5 marks]

Problem 4. Let G be a connected undirected graph on n vertices. We say that two distinct spanning trees T and S of G are *one swap away* from each other if $|T \cap S| = n - 2$; that is, T and S differ in only one edge.

For two distinct spanning trees T and S we say that $R_1, R_2, \ldots, R_k$ form a *swapping sequence* from T to S if:

1. $R_1 = T$,
2. $R_k = S$, and
3. for any $1 \leq i < k$, the trees R_i and R_{i+1} are one swap away from each other

Your task is to design a polynomial time algorithm that given G and two spanning trees T and S of G, constructs a minimum length swapping sequence. Remember to:

a) Describe your algorithm in plain English. [8 marks]

b) Prove the correctness of your algorithm. [7 marks]

c) Analyze the time complexity of your algorithm. [5 marks]