CISC-235*
Test #4
November 29, 2018

Student Number (Required) _____

Name (Optional) _____

This is a closed book test. You may not refer to any resources.

This is a 50 minute test.

Please write your answers in ink. Pencil answers will be marked, but will not be re-marked under any circumstances.

The test will be marked out of 50.

Question 1	/10
Question 2	/15
Question 3	/15
Question 4	/10
TOTAL	/50

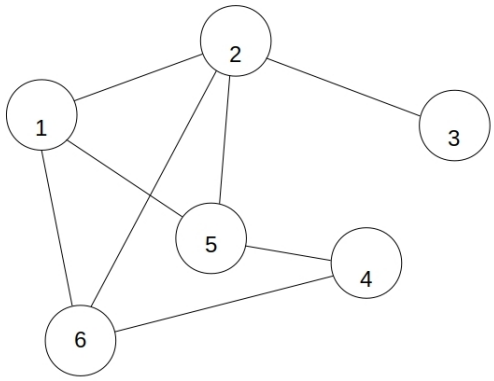
☐ By writing my initials in this box, I authorize Dr. Dawes to destroy this test paper if I have not picked it up by January 15, 2019.

Question 1: (10 Marks)

Let G be a graph with n vertices and m edges.

What data structure would you choose to represent the graph, so that it is possible to quickly determine the number of neighbours shared by two arbitrarily selected vertices? Explain your choice.

For example, in this graph Vertex 3 and Vertex 6 share exactly one neighbour (Vertex 2).

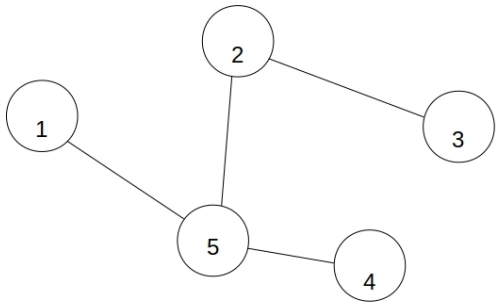


Question 2 (15 marks):

As we know, a tree is a graph that contains exactly one connecting path for each pair of vertices in the graph.

Suppose T is a tree on the vertex set $\{1, 2, \ldots, n\}$ with the edges stored in a list E, where each edge is in the form of a pair of vertices. The edges are randomly ordered in the list.

For example , if T looks like this



Then the list E might look like this: Head →(1,5) →(3,2) →(5,4) →(2,5) → Nil

Describe how, given vertices x and y, you would find the path in T that connects x and y. You do not have to write a full pseudo-code algorithm (unless you want to!) but do give enough detail to make it clear how your solution works.

Be sure to identify any data structures that you use in your solution.

For full marks, your solution should run in O(n) time.

Write your solution on the next page.

Write your solution to Question 2 on this page.

Question 3 (15 marks) :

Consider the following spanning tree algorithm, which is very similar to one of the algorithms we have studied. (The spanning tree it builds is not a minimum spanning tree, but that is not important for this question.) Examine the algorithm and then answer the questions stated below. **You can assume that in the graphs to which the algorithm will be applied,** $m \leq 4 * n$ (m is the number of edges, and n is the number of vertices). **You do not need to understand the purpose of the algorithm to answer this question**.

```
def test_alg(x):        # x is a vertex

        # initialization
        S = {}          # S is the set of edges that will be selected
        T = {x}         # The vertices we have connected to the tree
        R = V - {x}     # The rest of the vertices

        for each v in R:
             F[v] = 0                   # value of v
             N[v] = None                # best neighbour of v

        for each neighbour y of x:
             F[y] = weight(x,y)         # weight(x,y) is the weight
                                        #  of the edge from x to y
             N[y] = x

        # main loop
        while |T| < n-1:
             Let v be the vertex in R with the largest F value
             S = S + {(N[v],v)}
             T = T + {v}
             R = R - {v}
             for each neighbour z of v:
                if (z is in R)  and (min(F[v],weight(v,z)) > F[z]):
                     # update F[z] and N[z]
                     F[z] = min(F[v],weight(v,z))
                     N[z] = v

        return S
```

The question continues on the next page.

a) [10 marks] What data structure would you use to store the graph G? Why?

b) [5 marks] What data structure would you use to keep track of which vertices are in R? Why?

Question 4: (10 marks)

Imagine a social network in which friendships are always mutual (i.e. if A likes B then B likes A, and vice versa) and permanent (i.e. once a friendship is formed, it is never removed). New friendships are formed very frequently. What data structure would you use to make it easy to determine if two arbitrarily chosen people are connected by a chain of friends, and also easy to update the friendship information? Why?