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.

<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question 1</td>
<td>/15</td>
</tr>
<tr>
<td>Question 2</td>
<td>/10</td>
</tr>
<tr>
<td>Question 3</td>
<td>/10</td>
</tr>
<tr>
<td>Question 4</td>
<td>/15</td>
</tr>
</tbody>
</table>

TOTAL /50
Question 1 (15 marks)

Not all graphs are connected. Suppose G is a graph that may or may not be connected.

Create a modified version of either BFS or DFS that will find the connected components of G (i.e. the sets of vertices that are connected).

For example, if G looks like this

then your algorithm should print something like this (precise wording and vertex order are not important):

1, 7, 6 are connected

2, 4 are connected

3, 5 are connected

Please use the next page to write your algorithm. Your algorithm can contain high-level operations such as “for each neighbour of x” and “mark x visited”
a) [10 marks] Your algorithm, in pseudo-code:

b) [5 marks] What data structure would you use to store G, and why?
Question 2 (10 marks)

Let G be a connected, edge-weighted graph.

(a) [5 marks] Suppose we wish to compute the total weight of all edges in G. What data structure would you choose to represent G that would allow this calculation to be made most efficiently? Why?
(b) [5 marks] Let T be a MST of G, found by applying Prim’s Algorithm. If we add a constant value k to the weight of every edge in G, will the edges of T form a MST of the modified graph? Either explain why they will, or give an example of a graph where this does not hold.

(Hint: What would Prim’s Algorithm do when applied to the modified graph?)
Question 3 (10 marks)

Imagine a large social network in which friendships are always mutual (i.e. if A likes B then B likes A, and vice versa). What data structure would you choose to represent the network, so that it is possible to quickly determine if two arbitrarily chosen people have a mutual friend? Explain your choice.
Question 4 (15 marks)

Here is a version of Dijkstra's Algorithm for computing the total weight of the least-weight path from vertex x to vertex y in a connected graph with non-negative weights on the edges. Note that since we are only interested in the total weight of the path, this algorithm does not keep track of the chosen edges.

```python
def Dijkstra(x, y):
    R = { all vertices except x }
    let d be a 1-dimensional array of length n (where n = number of vertices)
    d[x] = 0  # d[] will contain all computed distances from x
    for each z in R:
        if z is a neighbour of x:
            d[z] = w(x, z)  # w(x, z) is the weight of edge (x, z)
        else:
            d[z] = infinity
    current = x
    while current != y:  # keep going until we reach y
        let v be a vertex in R with the smallest d[] value
        R = R - {v}
        for each neighbour z of v:
            if z is in R:
                if d[v] + w(v, z) < d[z]:
                    d[z] = d[v] + w(v, z)
        current = v
    # now return the total weight of the xy-path
    return d[y]
```

What data structures would you use so that the complexity of this algorithm is in $O(n^2)$? (Remember, $n$ is the number of vertices.) You will need to choose a data structure for the graph, and also a data structure for R that lets you find the vertex in R with the smallest d[] value. Explain your choices.

Hint: recall that Dijkstra’s Algorithm is just a modified version of Prim’s MST Algorithm.

Use the next page to answer this question.
This page is blank-ish