CISC-365*
Test #1 Sample Questions
Fall 2019

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 (15 marks)

Let A and B be two sets, each containing n integers in random order. Each of the sets is stored in an n-element array.

Create an algorithm to compute $A \cap B$ (that's "A intersect B"). Your algorithm should run in O($n * \log n$) time.

(A note on data structures: many people are tempted to solve problems like this using hash-tables which give O(1) ***expected case*** search time. Unfortunately the ***worst case*** search time for a hash-table is O(n).)

Express your algorithm in clear pseudo-code or a standard procedural language. You may assume that ***sort()*** is a built-in function that runs in O($n * \log n$) time.

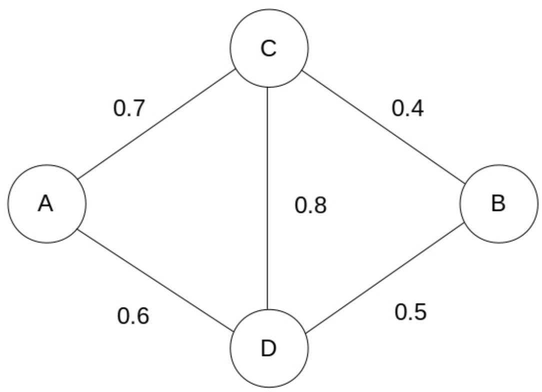
QUESTION (15 marks)

What is the computational complexity (ie the “big O” class) of this algorithm?

```
Mystery(n):
    if n <= 1:
        print 1
    else if n <= 100:
        print n
        Mystery(n-1)
    else:
        print n
        Mystery(n/2)
```

QUESTION (15 marks)

Consider the Path Product Problem: Given a graph G in which every edge is weighted with a number in the range [0 .. 1] , and given two identified vertices A and B, find a path from A to B that **maximizes** the **product** of the weights of the edges in the path.



For example in this graph the optimal path from A to B is A-D-B because 0.6 * 0.5 is greater than the product of the weights in any other path from A to B

Dijkstra's Algorithm be adapted to solve the Path Product Problem.

Dijkstra's Algorithm is stated on the next page, exactly as given in the course notes. This version finds the least-weight paths from A to all other vertices. You are **not** required to change it to terminate as soon as B is reached.

```
Dijkstra(W, A):
    Cost[A] = 0
    Reached[A] = True
    for each other vertex x:
        Reached[x] = False
    for each neighbour x of A:
        Estimate[x] = Weight(A,x)
        Candidate[x] = True
    for all other vertices z:
        Estimate[z] = infinity
        Candidate[z] = False
    while not finished:
        # find the best candidate
        best_candidate_estimate = infinity
        for each vertex x:
            if Candidate[x] == True and Estimate[x] < best_candidate_estimate:
                v = x
                best_candidate_estimate = Estimate[x]
        Cost[v] = Estimate[v]
        Reached[v] = True
        Candidate[v] = False
        for each vertex y:                    # update the neighbours of v
            if W[v][y] > 0  and  Reached[y] == False:
                if Cost[v] + W[v][y] < Estimate[y]:
                    Estimate[y] = Cost[v] + W[v][y]
                    Candidate[y] = True
                    Predecessor[y] = v
```

Explain how to modify this algorithm to solve the Path Product Problem. You don't need to copy the whole algorithm - just show the lines that need to change.

QUESTION (15 marks)

Let A be an array of n distinct integers ($n \geq 3$), arranged so that the integers start out increasing, and then decrease. For example A might look like this:

A = [2, 5, 7, 93, 86, 81, 77, 34, 22, 11, 9, 8, 6]

Create an algorithm that finds the largest value in A in $O(\log n)$ time. Your algorithm must solve all instances of the problem, not just the one given in the example.