# COMP 8851
# Instructor: Borna Noureddin
# Assignment #3

***<u>All work should be done individually.</u>***

Total marks: 130

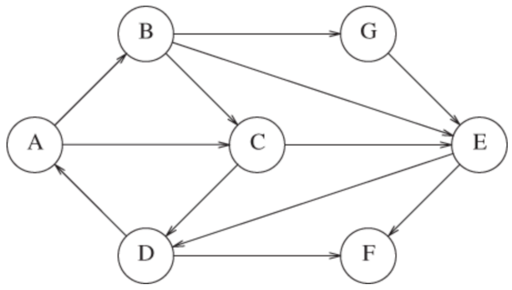
1. [5 marks] Rewrite the **`BinaryHeap insert`** routine by placing a copy of the inserted item in position 0.

2. [5 marks] What is the running time of heapsort for presorted input?

3. [10 marks] A large number of deletions in a separate chaining hash table can cause the table to be fairly empty, which wastes space. In this case, we can rehash to a table half as large. Assume that we rehash to a larger table when there are twice as many elements as the table size. How empty should the table be before we rehash to a smaller table?

4. [10 marks] Suppose we want to add the decreaseAllKeys($\Delta$) operation to the heap repertoire. The result of this operation is that all keys in the heap have their value decreased by an amount $\Delta$. For the heap implementation of your choice, explain the necessary modifications so that all other operations retain their running times and decreaseAllKeys runs in $O(1)$.

5. [10 marks] Find the strongly connected components in the graph below:



6. [25 marks] Implement the classic cuckoo hash table in which two separate tables are maintained. The simplest way to do this is to use a single array and modify the hash function to access either the top half or the bottom half.

7. [30 marks] You are a tournament director and need to arrange a round robin tournament among $N = 2^k$ players. In this tournament, everyone plays exactly one game each day; after $N - 1$ days, a match has occurred between every pair of players. Give a recursive algorithm to do this. Implement your algorithm in C++ and demonstrate that it works.

8. [35 marks] Create a 2D simulation with 100 white squares (they can be the same size or varying sizes). Use a background other than white. Give each square an initial position and velocity. Animate the squares so they are moving around the screen during the render loop. For the collision, implement a quadtree data structure of at least depth 4. The quadtree should NOT be recomputed every iteration of the loop, but rather updated to reflect new square positions. When the squares collide, change the color of each square involved in the collision to red. Once the squares have moved past each other and are no longer intersecting with another square, the colour should revert back to white. The quadtree implementation should be written in C++.