CS 310 'Algorithms' **Assignment-1**
Due date: Monday, October 01, 2018 at 11:55 PM. [Total Marks = 90]

Graph Algorithms

IMPORTANT: Read the instructions at the end of this document and follow the naming conventions.
Marks for writing well structured and efficient code. **[5 marks]**

Q1. **[22 marks]**
You are given a set of 'n' chemical compounds which have to be packed in two boxes. Some of the chemicals may react with other chemicals and should not be packed in the same box.

1. Code an **efficient** algorithm in C/C++ that determines if it is possible to safely pack these chemicals in two boxes and if so, then print which chemicals should go in Box-1 and which in Box-2. [8]
2. If it is not possible to safely pack the chemicals in two boxes then print at least one sequence of chemical interactions that prevents such a division. [8]
3. Give a clear description of your algorithm and the data structures used to implement it in the comments at the beginning of your code. **What is the running time of your algorithm and how much space does it use?** Your should include clear comments that explain your algorithm's time and space complexity. [2+1]
4. You should create at least three different input files to test your algorithm for different scenarios (see format below). [3]

Your code will read input from a text file. The format of the file is as follows. The first line contains the number of chemicals. This will be followed by n lines (one for each of the n chemicals) and each line will contain the chemical number followed by a colon and a comma separated list of chemicals with which it reacts. Some examples are given below.

Input :
n 6
C0: C1, C2
C1: C0, C3, C5
C2: C0
C3: C4, C1
C4: C3
C5: C1

Output Format:
Yes
C0, C3, C5
C1, C2, C4

Input :
n 6
C0: C1, C2

C1: C0, C3, C5
C2: C0, C4
C3: C4, C1
C4: C3, C2
C5: C1

Output Format:
No
C0->C1->C3->C4->C2->C0

Q2a.

[22 marks]

A group of world leaders are attending a United Nations meeting. Some of the leaders hold a grudge against some of the others. You are given the set of world leaders and a set of **ordered** pairs (L_i, L_j) of leaders such that L_i holds a grudge against L_j. Note that the feeling is not mutual, i.e., L_j does not have a grudge against L_i. The UN organizers have to take them to the meeting hall and they will be walking in a line down a passage. The organizers are afraid that L_i might try to throw something at L_j's back while they are in the line and want that those that hold a grudge (L_i) are not **somewhere** behind those they grudge (L_j) in the line (note the phrase "somewhere behind in the line"). The organizers have asked you for help.

1. Code an **efficient** algorithm in C/C++ that prints a safe ordering of the leaders in the line or lets the organizers know that it is not possible. [8]
2. If a safe ordering is not possible then print at least one sequence of grudges to illustrate that. [8]
3. Give a clear description of your algorithm and the data structures used to implement it in the comments at the beginning of your code. **What is the running time of your algorithm and how much space does it use?** Your should include clear comments that explain your algorithm's time and space complexity. [2+1]
4. You should create at least three different input files to test your algorithm for different scenarios (see format below). [3]

Your code will read input from a text file. The format of the file is as follows. The first line contains the number of leaders. This will be followed by n lines (one for each of the n leaders) and each line will contain the leader number followed by a colon and a comma separated list of leaders they hold a grudge against. Some examples are given below.

Input :
n 5
L0: L1, L2, L3
L1: L3
L2: L1
L3:
L4: L0

Output Format:
Yes
L4, L0, L2, L1, L3

Input :
n 5
L0: L1, L2, L3
L1: L3, L4
L2: L1
L3:
L4: L0

Output Format:
No
L4->L0->L1->L4

Q2b. **[10 marks]**

This part is a continuation of Q2a. Instead of lining up the leaders, you have to seat the leaders in rows while they are on the stage. Let's call the front row of the stage as Row-1 and second row as Row-2 and so-on. If L_i holds a grudge against L_j then L_i **must be in a lower numbered row** than L_j. Note: lengths of different rows (i.e. number of people sitting in that row) need not be the same. The organizers have asked you for help.

1. Code an **efficient** algorithm in C/C++ that finds the **minimum** number of rows needed. You have to print the number of rows and the order in which the leaders are seated in rows. If it is not possible to safely place the leaders in rows then mention that in the output. [7+1]
2. Give a clear description of your algorithm and the data structures used to implement it in the comments at the beginning of your code. The running time should not exceed that of Q2a. Your should include clear comments that explain your algorithm's time and space complexity. [2]
3. You may reuse the input files you created in Q2a above to test your algorithm.

Input :
n 5
L0: L1, L2, L3
L1: L3
L2:
L3:
L4: L0

Output Format:
Yes
4
L4
L0
L1, L2
L3

Input :
n 5
L0: L1, L2, L3
L1: L3, L4
L2: L1
L3:
L4: L0

Output Format:
No

Q3. **[16 marks]**

Consider a big road network. You are travelling from your home city (say city C_A) to visit a friend in another city (say city C_B) and you have figured out the **shortest length path** from C_A to C_B. Before leaving your home, you learn there is a scenic spot at city C_x and if you take a detour you can visit that before going to your friend's place. Since you are driving and you don't want to increase the distance you have to travel too much, you decide to only visit C_x if the shortest distance from C_A to C_B via C_x increases by no more than 20% of the shortest distance from C_A to C_B.

1. Code an **efficient** algorithm in C/C++ that determines if the distance from C_A to C_B via C_x is at most 20% more than the shortest distance from C_A to C_B. Your code will ask the user to input the values of C_A, C_B, and C_x (the cities are numbered from 0 to n-1, see format below). [6]
2. As output of your algorithm, print the following paths: (i) shortest path from C_A to C_B and its length, (ii) and the shortest path from C_A to C_B via C_x and its length. Use arrows(->) to separate your nodes. [4]
3. Give a clear description of your algorithm and the data structures used to implement it in the comments at the beginning of your code. **What is the running time of your algorithm and how much space does it use?** Your should include clear comments that explain your algorithm's time and space complexity. [2+1]
4. You should create at least three different input files to test your algorithm for different scenarios (see format below). [3]

Your code will read input from a text file. The problem will be modeled as a directed graph, where the vertices represent cities and the edges represent the roads that connect the cities. The direction on the edges represent that they are one-way roads. Each of the roads has a

length represented by a positive edge weight. The graph input file format is similar to Q1's except for the addition of the edge weights. The graph nodes are numbered from C_0 to C_{n-1}. The file will have the value of n on the first line. This will be followed by n lines (one for each of the n nodes in the graph) and each line will contain a node number (say C_i) followed by a colon and a comma separated list of nodes to which node i has an edge. The edge weights will be appended to each neighbor of i using a semicolon. For example, if there is an edge between node C_0 and node C_1 of weight 5 and edge between node C_0 and node C_5 of weight 2, then that will be represented as C0: C1;5, C5;2
An example is given below. Suppose user enters C0 as home city, C2 as friend's city and C3 as the scenic spot.

n 6
C0: C1;5, C5;2, C3;3
C1: C4;1
C2:
C3: C2;5
C4: C2;1
C5: C2;10

Asymptotic Complexity

Q4. **[10 marks]**

Q4-i What is the asymptotic time complexity of the following program fragment. [2]
Show your working.

```
for (i=1; i<=n; i*=2) {
    j = i;
 }
```

Q4-iia What is the asymptotic time complexity of the following program fragment. [3]
Give both the upper bound and lower bound for this fragment. Show your working.

```
int x, y, n;
...
/* P is a 1-D array of size n integers and
   W is a 2-D array of size n x n integers */
for (x=0; x<n; x++) {
  for (y=x+1; y<n; y++) {
    W[x][y] = func(P, x, y);
  }
}
...
```

The function func() called from the main program (above) is defined as follows:

```
int func(int *array, int i, int j)
{
  int ii, val=0;
  for (ii=i; ii<=j; ii++) {
    val += array[ii];
  }
  return (val);
}
```

Q4-iib What is being stored in the 2-D W array in Q4-iia? [2]

Q4-iic The program fragment in Q4-iia is not very efficient. Rewrite it to improve its time complexity. [3]

Q5. **[5 marks]**

You are given the following two `nxn`-dimensional matrices. The first matrix called "Intern Preference Matrix" stores information about `n` interns that want to apply for jobs at `n` different companies. Each row corresponds to one intern and indicates his/her order of preference for the employers. For example: Intern-1's first preference is Employer-3, second preference is for Employer-1, third preference is Employer-5 and so-on.

The second matrix called "Employer Preference Matrix" stores preference information of each of the `n` employers for the intern applicants. For example: Employer-1's first preference is Intern-2, second preference is for Intern-4 and so-on.

Intern Preference Matrix

Intern 1 (I_1)	E_3	E_1	E_5	E_8	...	E_2
Intern 2 (I_2)	E_5	E_2	E_1	E_7	...	E_3
Intern 3 (I_3)	...	...	...	...	...	...
...	...	...	...	...	...	...
Intern n (I_n)	...	...	...	...	...	...

Employer Preference Matrix

Employer 1 (E_1)	I_2	I_4	I_5	I_1	...	I_3
Employer 2 (E_2)	I_6	I_2	I_3	I_4	...	I_1
Employer 3 (E_3)	...	...	...	...	...	...
...	...	...	...	...	...	...
Employer n (E_n)	...	...	...	...	...	...

We have a number of queries such as "Does E_x prefer I_a over I_b?" or "Does I_x prefer E_y over E_z?" and we want to answer **each** query in O(1) time.

How would you restructure/re-store the above information so that we can answer such queries in O(1) time? You are only allowed to use arrays. You cannot use other data structures such as hash tables, etc. **Clearly** explain your answer and the **space** required by your solution.

Instructions and policies

1. You must submit your ***own*** work. You may discuss the problems with other classmates but must not reveal the solution to others or copy someone's work. Remember to acknowledge other classmates if discussions with them has helped you.
2. You should name your code files using the following convention: Qx-rollno.c
3. Type your answer to the theoretical questions (Q4 & 5) and submit a separate pdf file for each using the naming convention above.
4. Upload all your files in Assignment-1 folder on LMS. **There will be a 20% deduction for assignments submitted up to one day late. Assignments submitted 24 hours after the deadline will not be marked.**
5. There will be vivas during grading of the assignment. The TAs will announce a schedule and ask you to sign up for viva time slots. Failure to show up for vivas will result in a **70% marks reduction** in the assignment.
6. For full credit, comment your code clearly and state any assumptions that you have made. As mentioned in the beginning of the assignment, there are marks for writing well-structured and efficient code.
7. Familiarize yourself with LUMS policy on plagiarism and the penalties associated with it. We will use a tool to check for plagiarism in the submissions.