Data Structures (COMP 2000)
Assignment 3

Available Date: Tuesday, March 29, 2016
Due Date: 11.50 PM, Friday, April 15, 2016
Total Mark: 100 marks
Assessment
Coursework: 40%
Assignments (20%): A1 (7%), A2 (7%), A3 (6%)
Coursework exams (20%): CWE1 (10%), CWE2 (10%)
Final Examination: 60% (one two-hour writing exam)

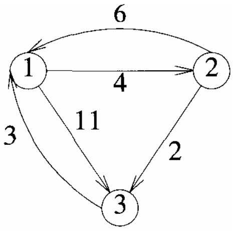
Assignment Requirements

Write the following complete C (or C++) programs.

1. `Floyd.c` to implement Floyd's algorithm to solve the all-pairs shortest-paths problem, where the directed weighted connected graph has no negative-length cycle. [40 marks]
2. `Dijkstra.c` to implement Dijkstra's algorithm to solve the single-source shortest-paths problem, where the directed weighted connected graph has no negative weight. [30 marks]
3. `BellmanFord.c` to implement Bellman and Ford's algorithm to solve the single-source shortest-paths problem, where the directed weighted connected graph has no negative-length cycle. [30 marks]

- A graph is represented using an adjacency matrix (also called weight, cost, or length matrix).

- For the `Floyd.c` program, you may use the following graph to test your program.



- The weight (cost, length) adjacency matrix of the graph is

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

Given weight adjacency matrix

	0	1	2
0	0	4	11
1	6	0	2
2	3	∞	0

Weight adjacency matrix stored in memory

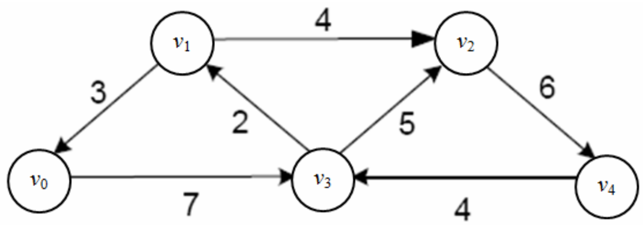
(I = 99999 indicates ∞)

A demonstrative output of the **`Floyd.c`** program is given below.

```
The weight matrix W is
  0  4 11
  6  0  2
  3  I  0
D(1) is
  0  4 11
  6  0  2
  3  7  0
D(2) is
  0  4  6
  6  0  2
  3  7  0
D(3) is
  0  4  6
  5  0  2
  3  7  0
The distance matrix D is
  0  4  6
  5  0  2
  3  7  0
The Path matrix is
 -1 -1  1
  2 -1 -1
 -1  0 -1

Path length =   4, Path from 1 to 2 is: 1 --> 2
Path length =   6, Path from 1 to 3 is: 1 --> 2 --> 3
Path length =   5, Path from 2 to 1 is: 2 --> 3 --> 1
Path length =   2, Path from 2 to 3 is: 2 --> 3
Path length =   3, Path from 3 to 1 is: 3 --> 1
Path length =   7, Path from 3 to 2 is: 3 --> 1 --> 2
```

- For the **`Dijkstra.c`** program, you may use the following graph to test your program.



- The cost (weight, length) adjacency matrix of the graph is

	v_0	v_1	v_2	v_3	v_4
v_0	0	∞	∞	7	∞
v_1	3	0	4	∞	∞
v_2	∞	∞	0	∞	6
v_3	∞	2	5	0	∞
v_4	∞	∞	∞	4	0

Given cost adjacency matrix

	0	1	2	3	4
0	0	I	I	7	I
1	3	0	4	I	I
2	I	I	0	I	6
3	I	2	5	0	I
4	I	I	I	4	0

Cost adjacency matrix stored in memory

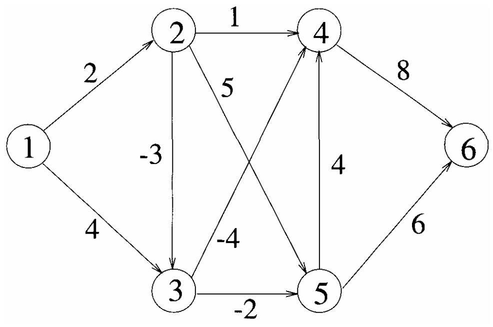
(I = 99999 indicates ∞)

A demonstrative output of the **`Dijkstra.c`** program is given below.

```
The weight matrix W is
     0     I     I     7     I
     3     0     4     I     I
     I     I     0     I     6
     I     2     5     0     I
     I     I     I     4     0
0. dist[] and parent[] are
     0     I     I     7     I
    -1    -1    -1     0    -1
1. dist[] and parent[] are
     0     9    12     7     I
    -1     3     3     0    -1
2. dist[] and parent[] are
     0     9    12     7     I
    -1     3     3     0    -1
3. dist[] and parent[] are
     0     9    12     7    18
    -1     3     3     0     2
The distance array dist[] is
     0     9    12     7    18
The parent array parent[] is
    -1     3     3     0     2

Path length =   9, Path from 0 to 1 is: 0 --> 3 --> 1
Path length =  12, Path from 0 to 2 is: 0 --> 3 --> 2
Path length =   7, Path from 0 to 3 is: 0 --> 3
Path length =  18, Path from 0 to 4 is: 0 --> 3 --> 2 --> 4
```

- For the **BellmanFord.c** program, you may use the following graph to test your program.



- The cost (weight, length) adjacency matrix of the graph is

	1	2	3	4	5	6
1	0	2	4	∞	∞	∞
2	∞	0	-3	1	5	∞
3	∞	∞	0	-4	-2	∞
4	∞	∞	∞	0	∞	8
5	∞	∞	∞	4	0	6
6	∞	∞	∞	∞	∞	0

Given cost adjacency matrix

	0	1	2	3	4	5
0	0	2	4	∞	∞	∞
1	∞	0	-3	1	5	∞
2	∞	∞	0	-4	-2	∞
3	∞	∞	∞	0	∞	8
4	∞	∞	∞	4	0	6
5	∞	∞	∞	∞	∞	0

Cost adjacency matrix stored in memory

(I = 99999 indicates ∞)

A demonstrative output of the **BellmanFord.c** program is given below.

```
The weight matrix W is
   0  2  4  I  I  I
   I  0 -3  1  5  I
   I  I  0 -4 -2  I
   I  I  I  0  I  8
   I  I  I  4  0  6
   I  I  I  I  I  0
dist(1) is
   0  2  4  I  I  I
  -1  1  1 -1 -1 -1
dist(2) is
   0  2 -1  0  2  I
  -1  1  2  3  3 -1
dist(3) is
   0  2 -1 -5 -3  8
  -1  1  2  3  3  4
dist(4) is
   0  2 -1 -5 -3  3
  -1  1  2  3  3  4
dist(5) is
```

```
   0  2 -1 -5 -3  3
  -1  1  2  3  3  4
The distance array dist is
   0  2 -1 -5 -3  3
The parent array is
  -1  1  2  3  3  4

Path length = 2, Path from 1 to 2 is: 1 --> 2
Path length = -1, Path from 1 to 3 is: 1 --> 2 --> 3
Path length = -5, Path from 1 to 4 is: 1 --> 2 --> 3 --> 4
Path length = -3, Path from 1 to 5 is: 1 --> 2 --> 3 --> 5
Path length = 3, Path from 1 to 6 is: 1 --> 2 --> 3 --> 4 --> 6
```

Submission: **carefully** submit your source program files to Mr. Sterling Ramroach via the email: sramroach@gmail.com.

• At the top of your program, you should include the following information.

```
/* Student Full Name:
   Student ID:
   E-mail:
   Course Code:
*/
```

End of Assignment 3