BITS Pilani, Pilani Campus
2nd Semester 2017-18
CS F211 Data Structures & Algorithms
Lab Test II - (15th April, 2018)

Maximum Marks: 20 + 40 = 60 **Duration: 160 + 10 = 170 min**

Instructions

- In this lab test you have to solve two problems given in two different files. Total time for solving the problems and uploading solutions is 160 minutes with at most 10 additional minutes for uploading the solution on the portal. Portal closing timer is shown at top-right corner of your page.
- You may make multiple submissions on the online portal, but only the latest submission shall be considered for evaluation. It is your responsibility to save and upload files at regular intervals.
- For each of the given two problems, upload all relevant source code (*.c) and header (*.h) files as a single submission. Do not upload any other file.
- If your program is having a "*compilation error*" or "*segmentation fault*" or other similar errors, we shall **evaluate your code for 50% of the marks only**. So, try to keep your code as clean as possible. Also note that **commented code** will **NOT** be considered for evaluation. Codes not compiling will have higher penalty as compared to codes with segmentation fault.
- Each problem is associated with a zip file which contains pre-filled code for solving the problem:
  - Write your code only in the designated areas in each of the file.
  - Do NOT rename any file, or create a new file.
  - Please note that each function may be tested separately for various test cases. So, don't make changes to the signatures and return types of the pre-defined functions.

Problem 2 of 2

[Expected Time: 115 minutes.] **[40 Marks]**

Consider a directed graph G = (V,E,w) - where w is an integer function on V - stored as an adjacency list. Write a ***best-first*** search procedure:

- start at a root
- pick the next vertex to be visited as one with the *highest weight* from *currentset* [*currentset* contains the unvisited children of all visited vertices]

Given a count of vertices, V, create an empty graph with adjacency list representation. The input graph will be supplied as an edge list, where each edge is represented by an ordered pair of vertices, where each vertex is an integer in the range 0..|V|-1. Weights are supplied as a separate input list.

Complete definition of following functions in `graph.c`. Write any additional functions and data structures in `extras.c` and `extras.h`. Consider `driver.c` as the driver program.

a) `Graph initGraph(int V)`
Return an empty V-vertices adjacency list with all relevant fields properly initialized.

b) `void printAdjacencyList(Graph g)`
Print data of each vertex in a line, with following format:
<vertex id> <tab> ==> <adjacent vertex id 1> <adjacent vertex id 2>

c) `void insertEdge(Graph g, unsigned int u, unsigned int v)`
Add an edge in the graph between u and v.

d) `void bestFirstSearch(Graph g, unsigned int root)`
Traverse and print the graph in best first order starting at root.

[Hint: You may implement a priority queue in extras.c and extras.h to identify the best vertex to pick for expansion during traversal. End of Hint]

Solution Upload Instruction:

`driver.c`	**Do not edit.**	Upload on the portal by browsing and selecting all of these together.
`graph.c`	Fill in function definitions for (a), (b), (c), and (d).	
`graph.h`	**Do not edit.**	
`extras.c`	For any additional implementation.	
`extras.h`	For any additional interfacing.	

Sample Test Case:

Input: `15042018 10`

Output:

```
8 1
6 5
3 9
8 9
6 1
8 5
2 4
2 5
8 7
5 0
7 9
9 5
8 2
9 8
6 0
3 7
8 4
2 7
5 9
0 2
Weights: 70      90    70    40    90    60    30    80    10    20
Adjacency List:
0     (70)  ==> 2
1     (90)  ==>
2     (70)  ==> 7       5     4
3     (40)  ==> 7       9
4     (90)  ==>
5     (60)  ==> 9       0
6     (30)  ==> 0       1     5
7     (80)  ==> 9
8     (10)  ==> 4       2     7     5     9     1
9     (20)  ==> 8       5
Best First Traversal: (root is 0)
0     2     1     3     7     9     4     5     6     8
```