# Boğaziçi University, Dept. of Computer Engineering

# CMPE 250, DATA STRUCTURES AND ALGORITHMS

## Fall 2012, Midterm 2

Name: ______________________________

Student ID: ______________________________

Signature: ______________________________

- Please print your name and student ID number and write your signature to indicate that you accept the University honour code.
- During this examination, you may not use any notes or books.
- Read each question carefully and **WRITE CLEARLY**. Unreadable answers will not get any credit.
- For each question you do not know the answer and leave blank, you can get %10 of the points, if you write only "I don't know the answer but I promise to think about this question and learn its solution".
- There are **5** questions. Point values are given in parentheses.
- You have **100 minutes** to do all the problems.

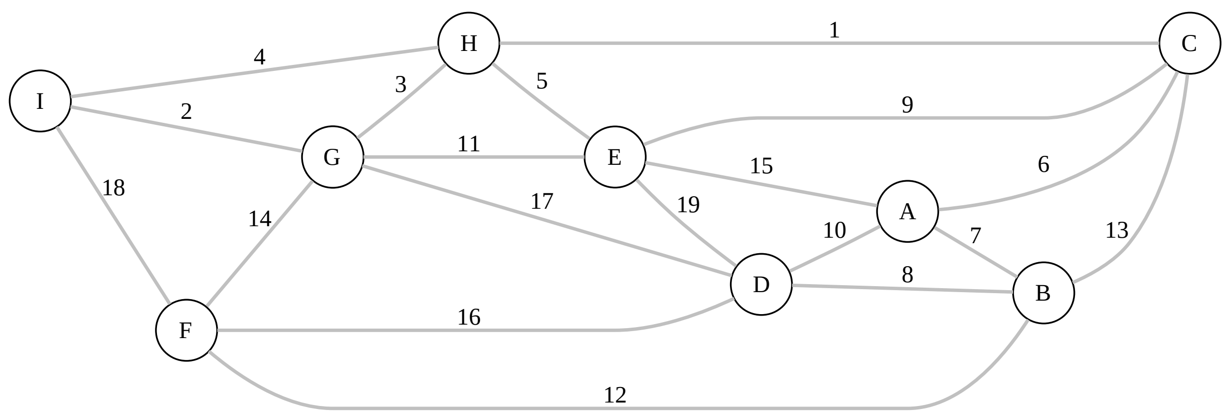
| Q | 1 | 2 | 3 | 4 | 5 | Total |
|---|---|---|---|---|---|---|
| Score | | | | | | |
| Max | 20 | 20 | 20 | 20 | 20 | 100 |

1. What is the output of the following C++ program? For each line numbered from 1-11, write the output. Every step must be explained. (Hint: Be careful with implicit calls to constructors and destructors).

```
#include <iostream>
using namespace std;

template <typename T>
struct obj{
  T i;
  obj(T j=0) : i(j) {cout<<'+';};
  obj(obj<T>& o2){this->i=o2.i; cout<<'<';};
  ~obj(){cout<<'-';};
  obj& operator=(obj<T>& o2){this->i=o2.i; cout<<'='; return o2;};
  T operator/(obj<T>& o2){cout<<'/'; return this->i/o2.i;};
  T operator/(int j){cout<<"i/"; return this->i/j;};
};

template <typename T>
void fun1(obj<T>& o){o.i=1; cout<<'1'; return;}

void fun2(obj<int> o){o.i=2; cout<<'2';};
template <typename T>
void fun2(obj<T> o){o.i=3; cout<<'3';};

int main(){
1       obj<int> o;                Output:

2       obj<double> p(2);          Output:

3       fun1(o); cout<<o.i;        Output:

4       fun2(o); cout<<o.i;        Output:

5       obj<int> o2=o;             Output:

6       obj<int> o3(o);            Output:

7       o2 = o;                    Output:

8       cout << o.i/p.i;           Output:

9       cout << o/o.i/2;           Output:

10      cout << p/p;               Output:

11      return 0;                  Output:
}
```

*(20 points)*

2. Minimum spanning tree. For parts (a), and (b) consider the following weighted graph with 9 vertices and 19 edges. Note that the edge weights are distinct integers between 1 and 19.




(a) Complete the sequence of edges in the MST in the order that Kruskal's algorithm includes them.

`1 ____ ____ ____ ____ ____ ____ ____`

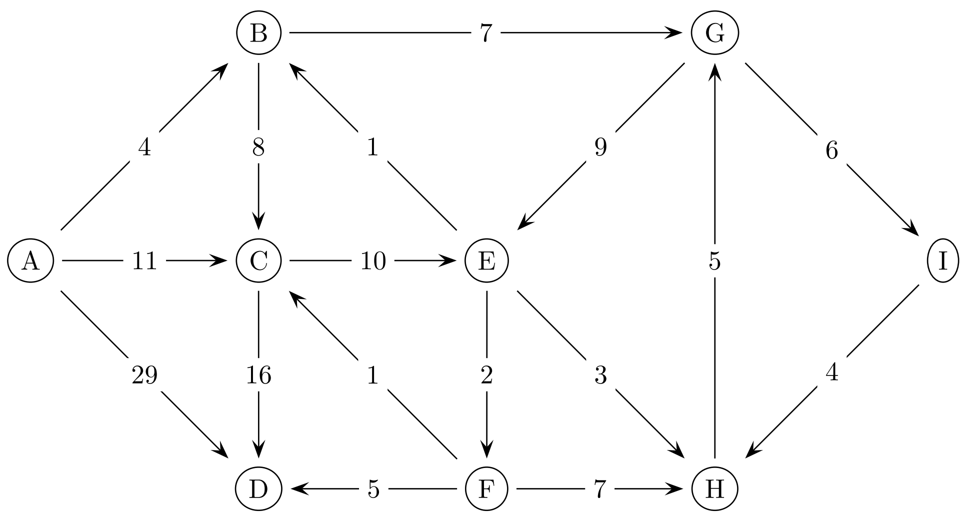
(b) Complete the sequence of edges in the MST in the order that Prim's algorithm includes them. Start Prim's algorithm from vertex A.

`6 ____ ____ ____ ____ ____ ____ ____`

(c) Given a minimum spanning tree $T$ of a weighted graph $G = (V, E)$, describe an $O(|V|)$ algorithm for determining whether or not $T$ remains a MST after an edge $x \leftrightarrow y$ of weight $w$ is added.

*(20 points)*

3. Run Dijkstra's algorithm on the weighted digraph below, starting at vertex A.




(a) List the vertices in the order in which the vertices are dequeued (for the first time) from the priority queue and give the length of the shortest path from A.

```
vertex:   A  ___ ___ ___ ___ ___ ___ ___ ___

distance: 0  ___ ___ ___ ___ ___ ___ ___ ___
```

(b) Draw the edges in the shortest path tree with thick lines in the figure above.

*(20 points)*

4. You are given a directed graph $G = (V, E)$, with a positive integer capacity $c_e$ on each edge $e$, a designated source $s \in V$, and a designated sink $t \in V$. Suppose you found the maximum $s - t$ flow in $G$, defined by a flow value $f_e$ on each edge $e$. Now suppose we pick a specific edge $e \in E$ and increase its capacity by one unit. Explain how to find a maximum flow in the resulting capacitated graph in time $O(m + n)$, where $m$ is the number of edges in $G$ and n is the number of nodes, that is without recomputing the maximum flow from scratch. *(20 points)*

5. (Perfect matching) Consider a bipartite graph $G = (R \cup C, E)$, a graph such that each edge has one endpoint in $R$ and one endpoint in $C$, and $R$ and $C$ have the same size.

   (a) Below is part of the adjacency matrix of $G$ showing links from $R$ to $C$

| | $C_1$ | $C_2$ | $C_3$ | $C_4$ |
|---|---|---|---|---|
| $R_1$ | 0 | 1 | 0 | 1 |
| $R_2$ | 1 | 0 | 0 | 0 |
| $R_3$ | 0 | 1 | 1 | 1 |
| $R_4$ | 0 | 1 | 0 | 1 |

   Draw this bipartite graph.

   (b) Describe an algorithm to find a set of edges $M \subset E$ such that $M$ is a perfect matching, that is edges in $M$ don't touch each other and each node in $R$ is associated with a single node in $C$. Using the algorithm find such a perfect matching and show it on the graph you drew. You must show all the intermediate steps

*(20 points)*