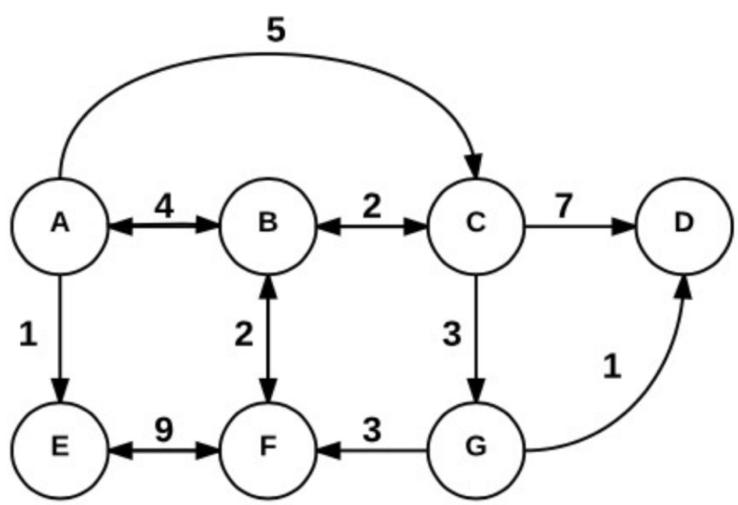
CS106B
Spring 2016 Cynthia Lee

Section 8 (Week 9) Handout

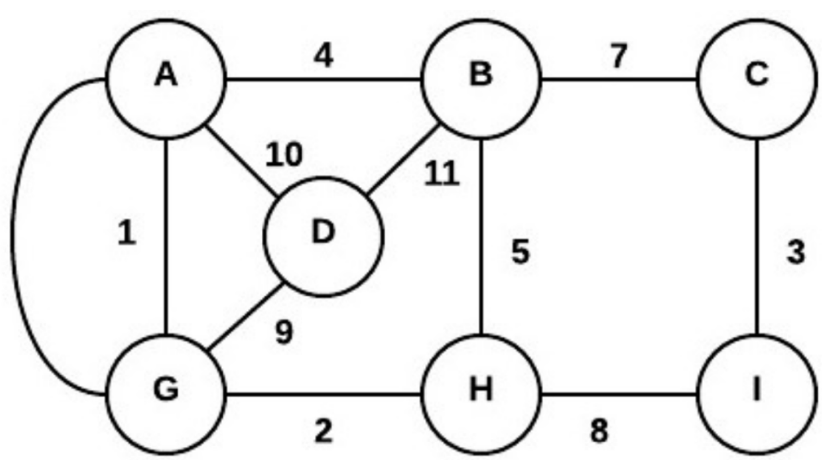
Problem and solution authors include Marty Stepp.

This week is about graph algorithms and inheritance, with a final exam topics sheet attached.

1. **Dijkstra and A*.** Trace through Dijkstra's algorithm on the following graph to find the shortest paths from node A to each other node in the graph. Then use A* to find the shortest path from A to G, using the heuristic where the distance between two nodes is the distance between those two letters in the alphabet. (For example, the distance between B and D is 2.)



2. **Kruskal**. List the edges that Kruskal's algorithm would select to be part of a minimum spanning tree (MST) for the graph above. List them in the same order that Kruskal's would add them to the MST. Then give the MST cost.



3. **isCyclic.** Write a function named **`isCyclic`** that accepts a reference to a BasicGraph and returns true if a path can be made from any vertex back to that same vertex (a cycle), or false if there are no cycles in the graph. To figure out whether a graph contains any cycles, use the following pseudo-code algorithm. The algorithm involves "marking" vertices as being in various states: unvisited, partially visited, or fully visited. It is up to you to decide how to implement such marking behavior.

```
at the start, all vertices and edges are UNVISITED.
for each vertex v in the graph:
    if visit(graph, v) returns true, then the graph contains a cycle.

function visit(graph, v):
    v is now PARTIALLY VISITED.
    for each neighbor vertex v2 of v where the edge e from v -> v2 is unvisited:
        mark that edge e as visited.
        if v2 is PARTIALLY VISITED, the graph contains a cycle.
        if v2 is UNVISITED and visit(graph, v2) returns true,
                                                 the graph contains a cycle.
    v is now FULLY VISITED.

bool isCyclic(BasicGraph& graph) { ...
```

4. **Inheritance and polymorphism**.

Consider the following classes; assume that each is defined in its own file.

```
class Hamburger : public Bacon {
public:
    virtual void m2() {
        cout << "H 2" << endl;
        Bacon::m2();
    }

    virtual void m4() {
        cout << "H 4" << endl;
    }
};

class Mayo : public Hamburger {
public:
    virtual void m3() {
        cout << "M 3" << endl;
        m1();
    }

    virtual void m4() {
        cout << "M 4" << endl;
    }
};

class Lettuce {
public:
    virtual void m1() {
        cout << "L 1" << endl;
        m2();
    }

    virtual void m2() {
        cout << "L 2" << endl;
    }
};

class Bacon : public Lettuce {
public:
    virtual void m1() {
        Lettuce::m1();
        cout << "B 1" << endl;
    }

    virtual void m3() {
        cout << "B 3" << endl;
    }
};
```

Now assume that the following variables are defined:

```
Lettuce* var1 = new Bacon();
Bacon* var2 = new Mayo();
Lettuce* var3 = new Hamburger();
Bacon* var4 = new Hamburger();
Lettuce* var5 = new Lettuce();
```

In the rows below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, **indicate the line breaks with slashes** as in "x / y / z" to indicate three lines of output with "x" followed by "y" followed by "z". If the statement does not compile, write "**compiler error**". If a statement would crash at runtime or cause unpredictable behavior, write "**crash**".

Statement	Output
a. `var1->m1();`	____________
b. `var1->m2();`	____________
c. `var1->m3();`	____________
d. `var2->m1();`	____________
e. `var2->m2();`	____________
f. `var2->m3();`	____________
g. `var2->m4();`	____________
h. `var3->m1();`	____________
i. `var3->m2();`	____________
j. `var4->m2();`	____________
k. `var4->m3();`	____________
l. `var4->m4();`	____________
m. `((Bacon*) var1)->m1();`	____________
n. `((Bacon*) var1)->m3();`	____________
o. `((Mayo*) var5)->m3();`	____________
p. `((Lettuce*) var4)->m3();`	____________
q. `((Hamburger*)var2)->m4();`	____________
r. `((Mayo*) var2)->m4();`	____________