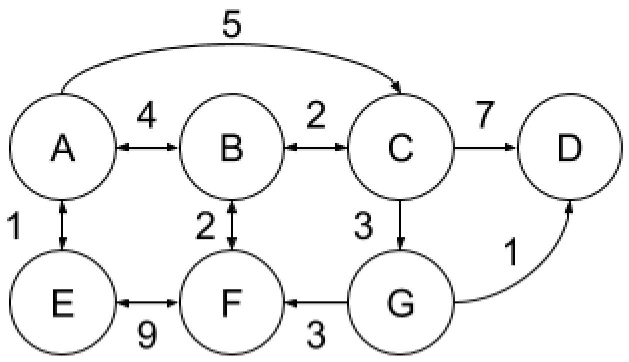
Week 9 Section Handout

This week's section handout has practice with graphs and inheritance. Follow along at CodeStepByStep: https://codestepbystep.com/problemset/view?id=72

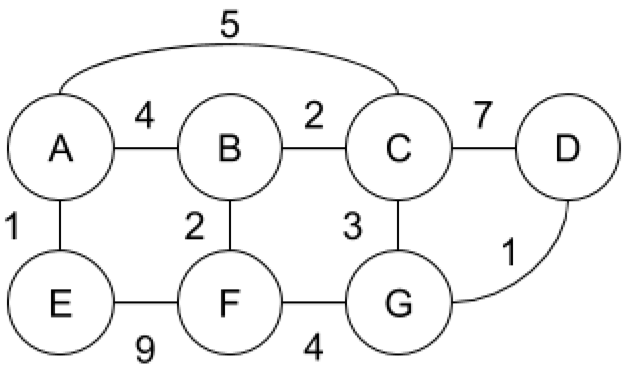
1. Dijkstra and A*

Trace through Dijkstra's algorithm on the following graph to find the least cost paths from node A to each other node in the graph. Then use A* to find the least cost path from A to G, where the heuristic is defined such that the distance between two nodes is the distance between those two letters in the alphabet. For example, the distance between B and D according to this heuristic is 2.



2. Minimum Spanning Tree

For the graph below, use Kruskal's algorithm to find the minimum spanning tree.



3. Polymorphism (not on CSBS)

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

```
class Lettuce {
public:
   void m1() { cout << "L 1" << endl; m2(); }
   void m2() { cout << "L 2" << endl; }
};
class Bacon : public Lettuce {
public:
   void m1() { Lettuce::m1(); cout << "B 1" << endl; }
   void m3() { cout << "B 3" << endl; }
};
```

Thanks to Marty Stepp, Chris Gregg, and other CS106B and X instructors and TAs for contributing problems on this handout.

```
class Hamburger : public Bacon {
public:
   void m2() { cout << "H 2" << endl; Bacon::m2(); }
   void m4() { cout << "H 4" << endl; }
};

class Mayo : public Hamburger {
public:
   void m3() { cout << "M 3" << endl; m1();}
   void m4() { cout << "M 4" << 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();
```

For each statement below, indicate the output produced. If the statement does not compile, write "COMPILER ERROR". If a statement would crash at runtime or cause unpredictable behavior, write "CRASH".

var1->m1();
var1->m2();
var1->m3();
var2->m1();
var2->m2();
var2->m3();
var2->m4();
var3->m1();
var3->m2();
var4->m2();
var4->m3();
var4->m4();

---

4. Polymorphism Mystery 8

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

```
class Hamilton {
public:
  virtual void m1() { cout << "H1  "; m2(); }
  virtual void m2() { cout << "H2  "; }
};
class Burr : public Hamilton {
public:
  virtual void m1() { Hamilton::m1(); cout << "B1  "; }
  virtual void m3() { cout << "B3  "; }
};
class Eliza : public Burr {
public:
  virtual void m2() { cout << "E2  "; Hamilton::m2(); }
  virtual void m3() { Burr::m3(); cout << "E3  "; }
};
class George : public Eliza {
public:
  virtual void m1() { cout << "G1  "; Burr::m1(); }
  virtual void m4() { cout << "G4  "; m2(); }
};
```

Now assume that the following variables are defined:

```
Hamilton* var1 = new Burr();
Hamilton* var2 = new Eliza();
Burr* var3 = new Eliza();
Eliza* var4 = new George();
```

For each statement below, indicate the output produced. If the statement does not compile, write "COMPILER ERROR". If a statement would crash at runtime or cause unpredictable behavior, write "CRASH".

```
var1->m1();
var1->m2();
var2->m1();
var2->m2();
var2->m2();
var3->m1();
var3->m2();
var3->m3();
var4->m1();
var4->m4();
((Burr*) var1)->m3();
((Eliza*) var2)->m4();
((George*) var4)->m4();
((George*) var4)->m3();
((George*) var2)->m4();
```