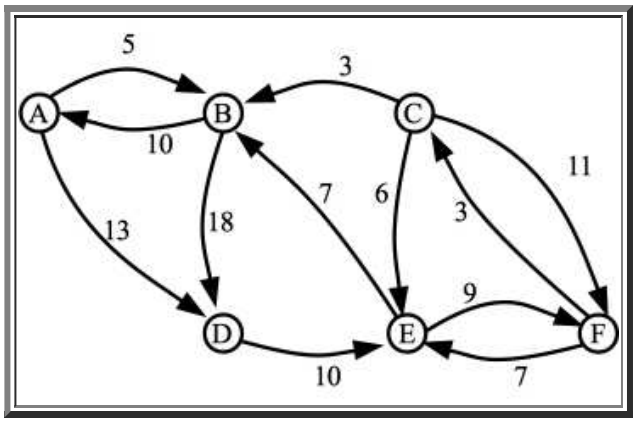
CS302 Final Exam, May 5, 2017 - James S. Plank

There are five questions to this exam. Please answer all five questions.
Hand in your work on the answer sheets provided.

My advocated timings on these questions are as follows. You may want to start with question 3, then do the others.

- Question 1: 15 minutes.
- Question 2: 20 minutes.
- Question 3: 10 minutes.
- Question 4: 30 minutes.
- Question 5: 30 minutes.

Question 1



Consider the graph to the right. Please answer the following questions about the network flow in this graph from node A to node F.

- **A**: If you are using the Edmonds-Karp algorithm, what is the next augmenting path through the graph?
- **B**: Please draw the residual graph after processing this path.
- **C**: Assume that we started with a directed acyclic graph, and that the picture above is the residual graph after we have processed two augmenting paths. What were those two paths and what were their flows?
- **D**: What is the maximum network flow of the original directed acyclic graph?
- **E**: What is the minimum cut of the original directed acyclic graph?

CS302 Final Exam, May 5, 2017 - James S. Plank

Question 2

In this question, do not bother drawing the graphs -- that will be a waste of your time.

Part A: To the right is the adjacency matrix for a directed, weighted acyclic graph. You are performing a topological sort of the graph to find the shortest path from node 0 to every other node. When you start the main loop of topological sort, you'll have three data structures with the following values:

```
deque <int> Q                = { 0 };
vector <int> Nincident       = { 0, 2, 2, 4, 2, 1, 2 };
vector <int> Sh_Path_Length = { 0, sent, sent, sent, sent, sent, sent };
```

In this specification, `sent` is a large sentinel value, like 1000000000.

On the answer sheet, show me what `Q`, `Nincident` and `Sh_Path_Length` equal after one iteration of the main loop.

	0	1	2	3	4	5	6
0			6	34	15	4	
1				1			
2		14			5		2
3							
4				33			
5		13	1				6
6				30			

Part B: To the right, I give you an adjacency list representation of a directed, weighted graph. Edges are represented as (node,weight). You are performing Dijkstra's algorithm to determine the shortest path from node 0 to every other node. After the first iteration of the main loop of Dijkstra's algorithm, you are maintaining the following data structures:

```
multimap <int int> Q = { [2,1], [4,5], [6,6], [7,3] };
vector <int> Sh_Path_Length = { 0, 2, sent, 7, sent, 4, 6 };
```

Tell me what the values of `Q` and `Sh_Path_Length` are afer the second iteration of the main loop.

```
0: { (5,4), (1,2), (6,6), (3,7) }
1: { (5,1), (6,2), (2,10), (3,6) }
2: { (3,1) }
3: { (2,10) }
4: { (2,3) }
5: { (1,1) }
6: { (3,4), (4,3), (2,9) }
```

Part C: To the right, I give you an adjacency matrix specification of an undirected, unweighted graph. The matrix is specified as a vector of strings, where `s[i][j] = '1'` if there is an edge from i to j and `'0'` otherwise. Self-edges are ok in this representation. You are performing Breadth-First Search to determine the shortest path from node 0 to every other node. After one iteration of the main loop, your two data structures, `Q` and `Sh_Path_Length` have the following values:

```
deque <int> Q = { 1, 2, 5, 6 };
vector <int> Sh_Path_Length = { 0, 1, 1, -1, -1, 1, 1, -1, -1, -1 };
```

Tell me what the values of `Q` and `Sh_Path_Length` are afer the second iteration of the main loop.

```
Adj[0]: "0110011000"
Adj[1]: "0111011100"
Adj[2]: "0010000001"
Adj[3]: "1000000010"
Adj[4]: "0010000100"
Adj[5]: "1010001110"
Adj[6]: "0010000100"
Adj[7]: "0100100001"
Adj[8]: "0111101000"
Adj[9]: "0100001000"
```

CS302 Final Exam, May 5, 2017 - James S. Plank

Question 3

Please tell me the big-O running times of the best algorithms for the following:

A: Determining whether there is a cycle in a directed graph with v nodes and e edges.
B: Finding the shortest path between two nodes in an weighted, undirected graph with v nodes and e edges.
C: Finding the minimum spanning tree of a graph with v nodes and e edges.
D: Determining the number of connected components in a graph with v nodes and e edges.
E: Finding the maximum flow path between two nodes in a directed, weighted graph with v nodes and e edges.
F: Finding the shortest path between two nodes in a weighted, directed graph with v nodes and e edges.
G: Finding the shortest path between two nodes in an unweighted, directed graph with v nodes and e edges.
H: Determining the shortest path between two nodes in a directed, weighted, acyclic graph.
I: Having finished determining the maximum flow of a graph with v nodes and e edges, determining the edges that compose its minimum cut.
J: In a complete, undirected graph with n nodes, where every node has an edge to every other node, printing all simple paths that contain every node.

Question 4

In the following problems, I would like you to tell me the following:

- Which graph algorithm from the class is best to solve the problem? If you are deciding between two algorithms, use the one with the best big-O running time.
- What kind of graph is it in terms of weighted/unweighted, directed/undirected?
- What are the nodes?
- How are the edges defined? Your answer here should be of the form, "There is an edge from node x to node y if ....." or something similar.
- If there are weights, how are they defined?
- How does the solution to the algorithm solve the problem?

Problem A: It's the year 2050, and scientists have finally figured out how to make people teleport. There is a network of teleport stations, where between every two stations X and Y, there may be up to two teleports -- one going from X to Y, and one going from Y to X. Each teleport takes a certain amount of time, and costs a certain amount of money. These are all specified, of course, on the web site `http://teleport-schedules.gov`. You want to write an app called *Taze,* which tells you the fastest way to get from station A to station B, where there is no waiting time when you transfer through a teleport station.

Problem B: Now, suppose that each teleport station forces you to wait 20 minutes, for security, before you are allowed to teleport from that station. Your goal is to derive a route from station A to station B which minimizes the amount of time that you spend waiting for security.

Problem C: Now, suppose that teleports are bi-directional. Between every pair of stations, there may be only one teleport, but it may be taken in either direction, for the same time and cost. Each teleport has a maintenance need, which is a number. Every night from midnight to 5 AM, the teleportation company wants to shut down as many teleports as possible for maintenance, while still enabling passengers to get from each station to each other station (potentially taking many teleports). How does the company determine the teleports on which to perform maintenance, in such a way that the maximum amount of maintenance is performed?

CS302 Final Exam, May 5, 2017 - James S. Plank

Question 5

Let us define two operations on a string:

1. Append an 'A' to the string.
2. Append a 'B' to the string and then reverse the string.

String *X* is "reachable" from string *Y* if you can perform a sequence of the operations above on string *Y* that results in string *X*. For example, "BAABBA" is reachable from "ABBA" with the following two operations:

- Append an 'A' to "ABBA" to yield "ABBAA".
- Append a 'B' to "ABBAA" and then reverse the string to yield "BAABBA".

With those definitions, take a look at the following class definition:

```
class Conversion {
  public:
    string Y;
    int Reachable(string X);
};
```

When `Reachable(X)` is called, you are to return 1 if string `X` is reachable from string `Y`, and 0 if it is not. Please do the following to answer this question:

Part A: Tell me in precise English how you will solve this problem with part 1 of Dynamic Programming. In other words, specify the recursion (including the base cases).

Part B: Implement `Reachable()` using Dynamic Programming with memoization. You may add data to the `Conversion` class definition for this.

In case you'd like a little more detail on how `Reachable()` is called, here's a program which takes *X* and *Y* on the command line, and prints "Yes" if *X* is reachable from *Y* and "No" otherwise.

```
int main(int argc, char **argv)
{
  Conversion c;
  int ans;
  string X;

  if (argc != 3) { fprintf(stderr, "usage: a.out X Y\n"); exit(1); }
  X = argv[1];
  c.Y = argv[2];
  ans = c.Reachable(X);
  printf("%s\n", (ans) ? "Yes" : "No");
  return 0;
}
```