CPS222 - DATA STRUCTURES AND ALGORITHMS

Project #5

Purpose: To give you experience representing and working with graphs

- It is recommended, but not required, to do this project in teams of 2.
- Put all your code and test data in a repo under https://github.com/gordon-cs/ named
  cps222p5-2018-last-last
  (where "last" and "last" are replaced by the last names of the partners). Be sure the repo is **private**, accessible *only* to the partners working on it.

For this project, you will be working with data representing the road network of a hypothetical small country. Input to your program will come from standard input (probably redirected from a disk file), and will consist of a series of data sets - each describing the road network in one province. Each data set will occupy a series of lines in the file, laid out as follows:

Line 1:	# of towns in the province (n) and # of roads (e)
Lines 2..n+1:	names of individual towns (one per line)
Lines n+2..n+e+1:	data on individual roads (one per line)

- The data on the first line will be two integers, separated by a space.
- The town name will begin in column 1, and will contain no embedded spaces.
- The first town listed (line 2 of the data set) will be the provincial capital.
- The road data will consist of two town names, a bridge flag character (B for bridge, N for none), and a real number (representing the distance in miles between the towns) - with a single space separating each item from the next one. (Note: a bridge flag value of B indicates that the road passes over a bridge that could be washed out in a bad storm.)

All data can be successfully read by using the C++ `istream` extractor (operator >>) with operands of type `string`, `char`, or `float` (as appropriate) - the extractor will automatically skip over the intervening spaces.

The data file may contain any number of individual data sets. The first line for each data set will always immediately follow the last line of the preceding data set. Example:

```
5 6
SALEM
BEVERLY
DANVERS
LYNN
WENHAM
BEVERLY DANVERS N 2.9
BEVERLY SALEM B 2.4
BEVERLY WENHAM N 5.2
DANVERS WENHAM N 4.2
DANVERS SALEM B 3.7
LYNN SALEM N 4.9
4 7 <- start of second data set - remainder omitted
....
```

You may assume the following about the input data file that will be used to test your project:

1. All input will be properly formatted as specified in this handout.
2. Every province will contain at least one town (the capital).
3. Every province will be a connected graph.

Requirements

Write a program to read input from a file formatted as described above, performing some or all of the following analyses on each data set. For full credit on the project, you must do the read and print (1) and any three of the four remaining items (2-5). For bonus credit (above the normal 100 points), you may do all five. Note that the requirements for Lab 12 will be to fulfill requirement 1 and the final lab hour will be made available for you to work on the remaining requirements with the professor's help. For this reason, credit for requirement 1 is reflected through your grade for Lab 12, and the remaining requirements will be reflected in your project grade - though you **must**, of course, fix any problems with your requirement 1 code identified during the grading of Lab 12.

Test your program using data of your own invention; the official test will be administered using data of the professor's choice. (Note: the sample data in this handout is a good initial test, but is not a sufficiently complete test!) Turn in the test data you used. (Just the input file.)

NOTE: You should include appropriate headings in your output to label each analysis, and some kind of separator (e.g. a line of dashes) to visually separate the various data sets for the different provinces. Reasonably neat appearance of the output will be a factor in grading.

1. Read and then echo print the data. The output format is to be a town, followed by a list of roads going from that town to other towns. The printout should start with the provincial capital (the first town in the list), and must follow a *breadth-first* order in listing the remaining towns. (Each town will be listed once, and each road twice, once under each town it connects.)

   Example output - based on input data given above. (Your order of listing towns and roads may differ, so long as towns are in some BFS order, and SALEM, the capital, is first):

```
The input data is:
       SALEM
              BEVERLY 2.4 mi via bridge
              DANVERS 3.7 mi via bridge
              LYNN 4.9 mi
       BEVERLY
              DANVERS 2.9 mi
              SALEM 2.4 mi via bridge
`             WENHAM 5.2 mi
       DANVERS
              BEVERLY 2.9 mi
              SALEM 3.7 mi via bridge
              WENHAM 4.2 mi
       LYNN
              SALEM 4.9 mi
       WENHAM
```

```
BEVERLY 5.2 mi
DANVERS 4.2 mi
```

2. When provincial officials visit the various towns in the province, they don't want to do any more driving than necessary. Determine the shortest route from the provincial capital to each of the other towns, and print out both the route and the distance.

   Example output:

```
The shortest routes from SALEM are:
      The shortest route from SALEM to BEVERLY is 2.4 mi:
            SALEM
            BEVERLY
      The shortest route from SALEM to DANVERS is 3.7 mi:
            SALEM
            DANVERS
      The shortest route from SALEM to LYNN is 4.9 mi:
            SALEM
            LYNN
      The shortest route from SALEM to WENHAM is 7.6 mi:
            SALEM
            BEVERLY
            WENHAM
```

3. The national government is committed to a road upgrading project, but has limited funds. They would like to achieve the goal of having a high quality route connecting every pair of towns in the province (but not necessarily directly), while minimizing the miles of road they rebuild. Print out the roads that should be upgraded to achieve this goal.

   Example output:

```
The road upgrading goal can be achieved at minimal cost by upgrading:
     BEVERLY to SALEM
     BEVERLY to DANVERS
     DANVERS to WENHAM
     LYNN to SALEM
```

4. The government is worried that the roads with bridges on them might fail in the event of a major storm. In that case, it is possible that the province might be partitioned into regions that have no link between them. Perform a worst-case analysis of how the province would be partitioned, given that ALL the bridges fail in the same storm. (I.e. perform a connected components analysis, ignoring the bridge edges.)

   Example output:

```
Connected components in event of a major storm are:
  If all bridges fail, the following towns would form an isolated group:
     SALEM
     LYNN
  If all bridges fail, the following towns would form an isolated group:
```

```
BEVERLY
WENHAM
DANVERS
```

5. The country this task is being performed for has an insecure relationship with one of its neighbors, and the government is worried that its neighbor is developing nuclear weapons. The government is concerned that if a city is obliterated, the remainder of the province might be partitioned. (Of course, there are other concerns here as well ...) Analyze the data to determine which town(s) in each province - if any - would result in the province becoming disconnected if they were obliterated.

   Example output:

   ```
   Obliterating any of the following would result in the province becoming
   disconnected:
                SALEM
   ```

   (Note: if the graph is biconnected, you should say something like "None" as the output for this option - don't simply output a blank space)

Grading:

1. Requirements 2-5 above are worth 25 points each.
2. **Collaboration**: Use git and GitHub to store and collaborate on your code. Each partner must submit some of the code. Work in a branch and use a pull request so your partner can comment. If you do all the work together, pair-programming style, switch accounts regularly so you each drive and each navigate, and so commits come roughly evenly from both accounts. Each partner must comment on code submitted by the other. If all the work is pair-programming, then the commit messages and pull requests can simply say that. But if one of you writes some code alone, the other should have some substantive comments on what is good, what is clear or unclear, and what could be improved. Balanced contribution and high quality code reviews are worth 5 points.
3. **Design**: Choose and define classes carefully. Give them good methods with well-chosen arguments. Define your classes in .h files, and implement them in matching .cc files. Class structure and design, and code quality, are worth 5 points.
4. **Comments**: Use class and method comments, in the .h files, to very clearly explain what they do. Your comments in the .h files should be sufficient that a classmate could use your code without seeing your .cc files. Similarly, another classmate should be able to re-implement your .cc files correctly with only your .h files as a guide (and no copy of this assignment). If there are complicated pieces of code in your .cc files that need explaining, write good comments there, too. Clear and complete comments are worth 5 points.
5. **Tests**: Write thorough tests. They can be unit tests (using googletest) and/or input files with matching expected output files. Good and complete tests are worth 5 points (plus all the points for correctness they help your code earn).
6. **Makefile & output:** Write a Makefile to compile your code and run your tests. Format your output so it is clear and neat. Together, these are worth 5 points.
7. **Status**: Write a Status.md file just before the project is due, explaining what you attempted and its status. Clearly state what works, what tests pass and fail, and any known bugs or problems. (Required. 0 points for clear, complete, and accurate; up to -5 if wrong, incomplete, or missing)

Note that up to 125 of 100 points are possible.

There will not be a project quiz for this project.

Implementation Notes:

1. There is no starter code to copy for this project. You're on your own to create everything!
2. This is one project where careful thought about data structures in advance of coding can make a significant difference in terms of effort and ultimate correctness. For example, you might want to store some additional information in your data structures beyond that included in the handout of examples given out in class. I would be happy to discuss ideas with you before you implement them - any time prior to one week before the project due date.
3. You should make use of C++ `std` library containers to avoid having to create auxiliary data structures from scratch. At the very least, the map and queue containers will probably be useful, and maybe others as well. Note that the `std::list` container has a `sort()` method which can be used to sort the elements stored in a list, provided that an appropriate operator < is defined for the list elements. This might prove useful when doing the minimum cost spanning tree.
4. If you create objects to represent the vertices which contain fields that your methods modify, and want to pass these objects as parameters to a method and/or you want to store them in a `std` container, use a pointer to the object, not the object itself. (Otherwise you end up passing/storing a copy of the object, which will mess you up if you try to modify the object in the method or after removing it from the container - you will modify the copy, not the real object. You could solve this for a method by passing the parameter by reference, but `std` containers cannot store references - so you have to use pointers in containers.)
5. You can use the map container to help you with the fact that edges are specified by giving town names in the input file, as in the example constructor code handed out in class.
6. Since your program will process several different graph objects, declare your graph variable inside the block of your main loop (so it will be constructed afresh each time through), and be sure you have defined an appropriate destructor - see the examples in the handout.

Important Notes to Facilitate Official Testing

1. **Be sure that your program can read and process any number of data sets, not just one. This means that your program will need to check for end of file on standard input to know when there is no more data to process.** This can be a bit tricky, because the C++ `eof()` method does not become true until after you have tried (and failed) to read an item due to end-of file. The following code will do the job - note that it tries to read a character and, if successful, puts the character back into the input stream where it can be read "officially" when needed - else it exits.

```
while (true)
{
    // Check for eof and exit if so
```

```
    char c;
    std::cin >> c;
    if (std::cin.eof())
        break;
    else
        std::cin.unget();

    // Read a data set and construct its internal form
    HighwayNetwork theNetwork(std::cin);
    ...
```

But of course, it would be better to write a function, so the loop looks more like this.

```
while (!eof(std::cin))
{
   // Read a data set and construct its internal form
   HighwayNetwork theNetwork(std::cin);
   ...
```

2. Be sure to label output and include some sort of visual separator between distinct data sets.

3. Do not naively assume that a program that handles the sample data shown above correctly is correct! The sample data I have given you is meant to show you what the format of the output should look like - it is by no means a thorough test of your program! As noted above, test data you submit showing that you have exercised your program thoroughly will be one criterion used in evaluating your project.