This solution is implemented all the way up to an A. It can take in a text file with any extension including .gv for graphViz integration.

1. **For the input text files**, they must contain an adjacency matrix with space-separated values and a -1 at the end of each row. The matrix must be square (not counting the -1's). The values that are in the matrix must be integers that represent the weight of that edge. If all weights are 1, the graph is unweighted. If the value is 0, there is no edge there. No values less than 0 can be in the matrix, and therefore this program does not support negative edges. If the matrix is symmetric, it is an undirected graph, otherwise it is assumed to be directed. No given size is needed. The text file can be inputted via the second command line argument (after ./Graphs).

   Examples of valid text files:

| | |
|---|---|
| 0 1 0 0 1 1 -1 | 0 4 0 9 0 0 -1 |
| 1 0 1 1 0 1 -1 | 0 0 2 0 0 2 -1 |
| 0 1 0 1 0 0 -1 | 0 0 0 0 6 7 -1 |
| 0 1 1 0 1 1 -1 | 0 0 0 0 0 0 -1 |
| 1 0 0 1 0 1 -1 | 0 0 0 2 0 1 -1 |
| 1 1 0 1 1 0 -1 | 0 0 0 1 0 0 -1 |
| **Unweighted Undirected** | **Weighted directed** |

2. **For the input graphViz files**, whether the graph will be directed or undirected is only determined by the header: "graph" or "digraph". The name of the graph does not matter. The connector ( "--" or "->") has no effect on the input. The opening curly brace can be on the same line as the header or on its own line. **Each edge must be on its own line** with a space on at least one side of the connector ( no "1->3"). Weight can be given as [weight = x] where x is the weight. If there is no weight given, it is assumed to be 1. There can be any amount of space between anything on a given line as long as the word "weight" is not broken up. The final curly brace has to be on its own line. Any characters that follow "//" on a line will be ignoredm, so comments are allowed. Overall, if graphViz can not run it, then this program will not run it. The graphViz file can be inputted via the second command line argument (after ./Graphs).

   Examples of valid graphViz files:

| | |
|---|---|
| graph G{ | digraph sdflkj{ |
| 0 -- 1 | 0 -> 1 [weight = 4] |
| 0 -- 4 | 0 -> 3 [weight=9] |
| 0 -- 5 | //this is comment |
| 1 -- 2 | 1 -> 2 [ weight = 2 ] |
| 1 -- 3 | 1 -> 5 [weight= 2] |
| 1 -- 5 | 2 -> 4 [weight = 6] //this is an inline comment |
| 2 -- 3 | 2 -> 5 [weight = 7] |
| 3 -- 4 | 4 -> 3 [weight = 2] |
| 3-- 5 | 4 -> 5 |
| 4 --5 | 5 -> 3 //this weight would be 1 |
| } | } |

3. **To run the program**, it needs to be run from within CLion's debugger to properly open files with the correct file path. If desired, the file paths can be modified in the myGraph file constructor and the makeDotFile member function. The program takes 3 or 4 arguments. The **first argument** will always be "./Graphs". The **second argument** is the name of the input file with any extension (.gv for graphViz file). The **third argument** is one of eight operations (listed below) that the user wants to perform on the given graph. The **fourth argument** is optional. It is the name of the graphViz file (no extension required) that the program will produce. If no fourth argument is provided a default name will be given as the name of the input file with the word "Out" concatinated to the end. The fourth argument will only be used for functions that produce a graphViz file. If any arguments are invalid, an error message will be displayed accordingly.

    Valid options are:

    1. findCycle
    2. Topological
    3. shortestPath
    4. primsMST
    5. Dijkstras
    6. maxFlow
    7. kruskalsMST
    8. eulerCircuit