

You will receive 100% for turning this project in and passing the functionality test and style check before the deadline. After the deadline is 25% off.

This project is strictly individual.

Total points: 110

Before you start this project, you should have read the [project submission guidelines](#) and the [coding style rules](#).

After that, get started on the project *immediately*! You will submit your solution to the [new submission site](#).

Project Description

The goals of this project are to:

- Implement a graph data structure,
- Implement a heap data structure,
- Implement advanced algorithms for finding shortest paths,
- Practice the use of data abstraction,
- Practice coding an advanced networking algorithm,
- Test your program in the command line environment with input/output redirection and diff utilities, and
- Get you familiar with the [new submission site](#) for project uploads

Project description

You are in the planning stages of setting up the distribution network for a internet-based television system. There is an existing network of computers which are at different cities, which are the endpoints for your network. These computers have connections to each other with varying delay times.

For a network configuration, you need to also designate one computer as the primary source for the television content. This

computer will send the content over the network to all other computers. This source computer must be chosen in such a way that it minimizes the total transmission time to all the other computers in the network.

Network layout

We will represent the network as a set of computers connected by network links. Each computer will be named after the city it is in. Every name will be made up of only lowercase letters and underscores.

Each network link will connect two computers, in a particular direction, and will have an associated delay time. Note that just because there is a network link from computer A to computer B doesn't mean that there is one from B to A. Even if there is, they may have different delays.

Input

Your input will consist of a description of the existing network. You will get as input the number of network connections, followed by a list of the connections with the delay of each connection. Delay times will always be non-negative integers (representing milliseconds).

Not all computers may be the server. Only a computer that has the string "_server" at the end of its name may be a server.

Output

You should print out the name of the computer that has the shortest total delay time to all other computers in the network, as well as the total delay time. If there are multiple computers that have equivalent delay times, print them all, sorted in alphabetic order.

It may be that some networks do not have a server that can reach every other computer. If this is the case, print an appropriate error message.

Sample input and output

Here are several sample inputs and outputs.

- [input.1](#)
- [input.2](#)
- [input.3](#)
- [input.4](#)

As in previous projects, if you give a command-line argument to these executables, they will print extra information about how they are running.

Provided code

You must use the .h files provided here. I have provided a lot of code in this files. You are not allowed to change the existing code in this file. Instead, write your code in and submit the edited file.

- [arrayheap-prof-proj6.h](#)
- [arrayheap-student-proj6.h](#)
- [graph-proj6.h](#)

You will recycle most of the ArrayHeap code you wrote for project 5, with a modification that allows you to change the value on the heap. This is necessary to implement Dijkstra's algorithm.

Remember that when using templates, all of the code you write goes in the (student's) .h file. You will submit these files: arrayheap-student-proj6.h, graph-proj6.cpp, and driver-proj6.cpp.

In your driver, you may find the [STL map](#) useful for converting the strings given on the input to integers which are used in the Graph class as vertex indexes. We will talk about this in class.

Please feel free to use the following STL cheatsheets to learn more about some of the STL types used in this project: [cheatsheet 1](#), [cheatsheet 2](#).

Structuring the project

Since this is a large project, it helps to have a plan of attack. The following milestones will not be graded but you should try to do them to finish on time.

You will receive 100% for turning this project in and passing the functionality test and style check before the deadline. After the deadline is 25% off.

This project is strictly individual.

Total points: 110

Before you start this project, you should have read the [project submission guidelines](#) and the [coding style rules](#).

After that, get started on the project *immediately*! You will submit your solution to the [new submission site](#).

Project Description

The goals of this project are to:

- Implement a graph data structure,
- Implement a heap data structure,
- Implement advanced algorithms for finding shortest paths,
- Practice the use of data abstraction,
- Practice coding an advanced networking algorithm,
- Test your program in the command line environment with input/output redirection and diff utilities, and
- Get you familiar with the [new submission site](#) for project uploads

Project description

You are in the planning stages of setting up the distribution network for a internet-based television system. There is an existing network of computers which are at different cities, which are the endpoints for your network. These computers have connections to each other with varying delay times.

For a network configuration, you need to also designate one computer as the primary source for the television content. This computer will send the content over the network to all other computers. This source computer must be chosen in such a way that it minimizes the total transmission time to all the other computers in the network.

Network layout

We will represent the network as a set of computers connected by network links. Each computer will be named after the city it is in. Every name will be made up of only lowercase letters and underscores.

Each network link will connect two computers, in a particular direction, and will have an associated delay time. Note that just because there is a network link from computer A to computer B doesn't mean that there is one from B to A. Even if there is, they may have different delays.

Input

Your input will consist of a description of the existing network. You will get as input the number of network connections, followed by a list of the connections with the delay of each connection. Delay times will always be non-negative integers (representing milliseconds).

Not all computers may be the server. Only a computer that has the string "_server" at the end of its name may be a server.

Output

You should print out the name of the computer that has the shortest total delay time to all other computers in the network, as well as the total delay time. If there are multiple computers that have equivalent delay times, print them all, sorted in alphabetic order.

It may be that some networks do not have a server that can reach every other computer. If this is the case, print an appropriate error message.

Sample input and output

Here are several sample inputs and outputs.

- [input.1](#)
- [input.2](#)
- [input.3](#)
- [input.4](#)

As in previous projects, if you give a command-line argument to these executables, they will print extra information about how they are running.

Provided code

You must use the .h files provided here. I have provided a lot of code in this files. You are not allowed to change the existing code in this file. Instead, write your code in and submit the edited file.

- [arrayheap-prof-proj6.h](#)
- [arrayheap-student-proj6.h](#)
- [graph-proj6.h](#)

You will recycle most of the ArrayHeap code you wrote for project 5, with a modification that allows you to change the value on the heap. This is necessary to implement Dijkstra's algorithm.

Remember that when using templates, all of the code you write goes in the (student's) .h file. You will submit these files: arrayheap-student-proj6.h, graph-proj6.cpp, and driver-proj6.cpp.

In your driver, you may find the [STL map](#) useful for converting the strings given on the input to integers which are used in the Graph class as vertex indexes. We will talk about this in class.

Please feel free to use the following STL cheatsheets to learn more about some of the STL types used in this project: [cheatsheet 1](#), [cheatsheet 2](#).

Structuring the project

Since this is a large project, it helps to have a plan of attack. The following milestones will not be graded but you should try to do them to finish on time.