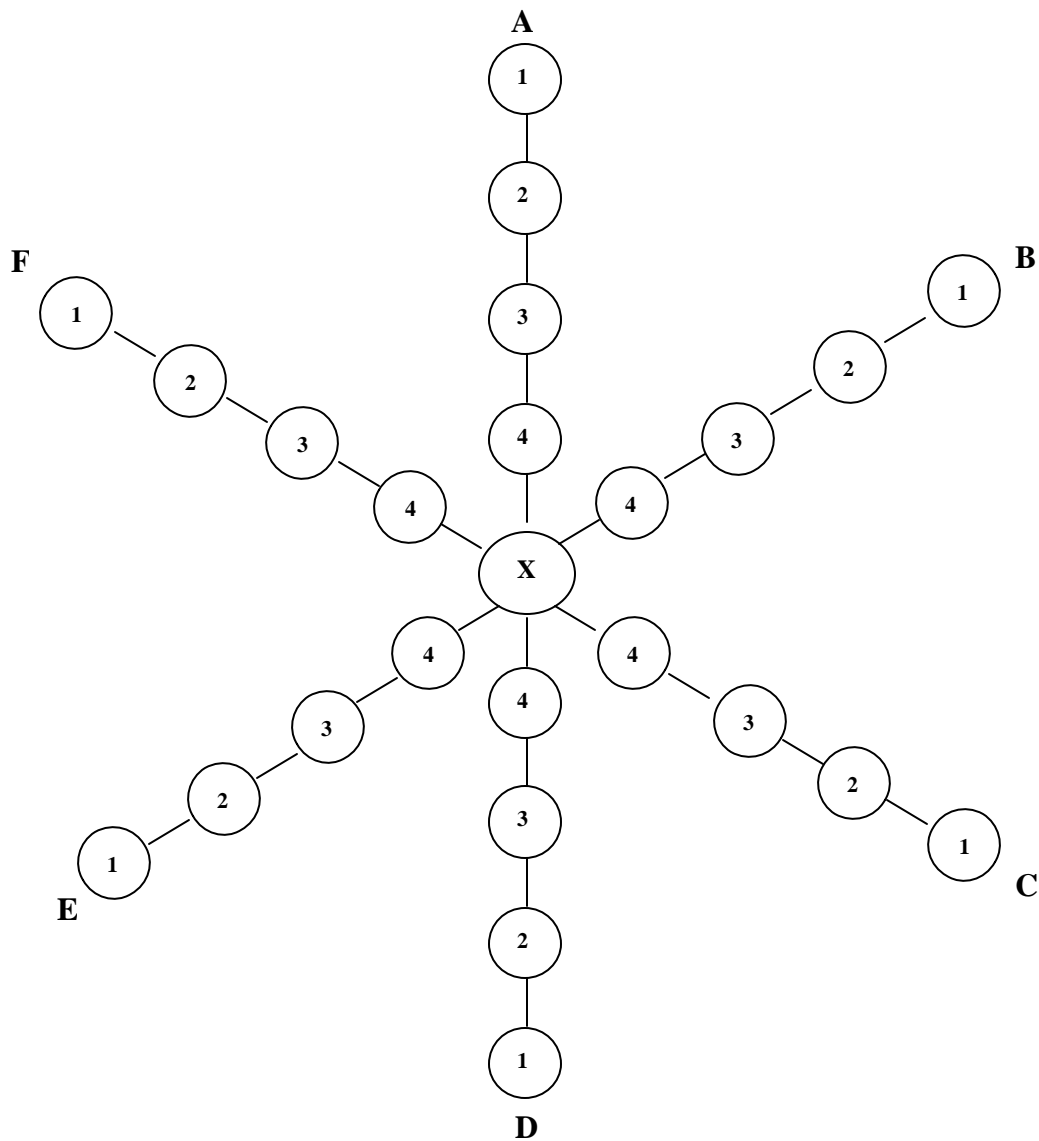


Assignment for Basics of C++ language 2008 – 2009

Bus schedules

1. Definition of the assignment

The route map of the bus routes in a town is the following:



The routes hence consist of six lines and 25 bus stops, four stops on each line. The stop X is a central station that is visited by every bus. The route types are the following:

Route type 1: Bus starts from the end of one line and continues via the central station to the end of another line and stops there.

Route type 2: Bus starts from the end of a line and stops at the central station.

Route type 3: Bus starts from the central station, drives to the end of a line and stops there.

It takes five minutes for a bus to drive from one stop to another; the bus leaves immediately after visiting a bus stop.

The task is to write a program that, based on the given timetable prints the fastest way to get from the given initial stop to the given destination bus stop at the given time. You also take into account the case that it is not possible to get from the initial bus stop to the destination. If bus must be switched at the central station, the first possible bus to the central station shall be taken.

Timetable is read from a file, whose name is given as a command line argument for the program. Also initial bus stop, destination bus stop and time are given as command line arguments.

2. Input for the program

The program reads the timetable from a file; each line in the file correspond the schedule for one bus. The form of the line is one of the following:

```
BLT Num Startline Destinationline Time1 Time2 Time3 ...
BLI Num Startline Destinationline Starttime Endtime Timeinterval
BXL Num Destinationline Starttime Endtime Timeinterval
BLX Num Startline Starttime Endtime Timeinterval
```

The three letter code in the beginning defines the type of the schedule.

All times are represented in the form h.m, where h = 0,1, ..., 23 and m = 00, 01,02, ... ,09, 10, 11, 12, ..., 58, 59.

2.1 Explanation of the lines in timetable

2.1.1 BLT type

BLT Num Startline Destinationline Time1 Time2 Time3 ...

Num = Bus number, can be any character string without whitespace characters.

Startline = The location line of the bus's initial stop, one of the letters A-F.

Destinationline = The location line of the bus's destination stop, one of the letters A-F.

Time1 Time2 ... list of the bus's starting times in time order.

Example:

BLT 18 A D 11.15 12.45 14.00 15.15

The bus number 18 starts from the end of A line (A1) at times 11.15, 12.45, 14.00 and 15.15. The bus starts from the central station (X) 20 minutes after starting from the initial bus stop and arrives at the D line's end stop (D1) 40 minutes after starting from the initial bus stop.

2.1.2 BLI type**BLI Num Startline Destinationline Starttime Endtime Timeinterval**

Num = Bus number, can be any character string without whitespace characters.

Startline = The location line of the bus's initial stop, one of the letters A-F.

Destinationline = The location line of the bus's destination stop, one of the letters A-F.

Starttime = The time at which the buses start to run.

Endtime = The time from which the buses do not run anymore.

Timeinterval = The interval between the departures of the buses.

Example:

BLI 21a F C 9.15 12.15 0.30

The bus number 21a starts from the end of the F line (F1) at times 9.15, 9.45, 10.15 ... 12.15. The bus starts from the central station (X) 20 minutes after starting from the initial bus stop and arrives at the C line's end stop (C1) 40 minutes after starting from the initial bus stop.

2.1.3 BXL type**BXL Num Destinationline Starttime Endtime Timeinterval**

Num = Bus number, can be any character string without whitespace characters.

Destinationline = The location line of the bus's destination stop, one of the letters A-F.

Starttime = The time at which the buses start to run.

Endtime = The time from which the buses do not run anymore.

Timeinterval = The interval between the departures of the buses.

Example:

BXL 10 E 7.30 21.30 1.00

The bus number 10 starts from the central station (X) at times 7.30, 8.30, 9.30 ... 21.30. The bus arrives at the E line's end stop (E1) 20 minutes after starting from the initial bus stop.

2.1.4 BLX type

BLX Num Startline Starttime Endtime Timeinterval

Num = Bus number, can be any character string without whitespace characters.

Startline = The location line of the bus's initial stop, one of the letters A-F.

Starttime = The time at which the buses start to run.

Endtime = The time from which the buses do not run anymore.

Timeinterval = The interval between the departures of the buses.

Example:

BLX 7 B 6.15 17.00 0.15

The bus number 7 starts from the end of the B line (B1) at times 6.15, 6.30, 6.45 ... 17.00.

The bus arrives at the central station (X) 20 minutes after starting from the initial bus stop.

2.2 Command line arguments

Six command line arguments are given to the program.

- The first argument is an option, either -show or -quiet. The first option (-show) means that the program prints both to console window and to a file. The latter option (-quiet) means that the program only prints to a file.
- The second argument is the initial bus stop. The bus stop is given in the form B3, where the letter represents the line and the number represents the number of the bus stop on the line; the central station is marked with the letter X.
- The third argument is the destination bus stop given in the same form as the initial bus stop.
- The fourth argument is the time starting from which the fastest way to get from the initial bus stop to the destination bus stop is computed. Time is represented in the form described previously.
- The fifth argument is the name of the timetable file.
- The sixth argument is the name of the output file.

All six arguments must be given, otherwise the program prints instructions how to use the program. The program must be able to handle also erroneous inputs (first option is incorrect; bus stop is given in a wrong form; time is given in a wrong form; input file does not exist or is incorrect).

3. Output of the program

3.1 Output to the console window

If command line arguments are lacking or are of the wrong form, the program prints instructions how to use the program. If the input file is erroneous or is lacking, the program prints an announcement. If there is no bus traveling between the input bus stops, the program prints an announcement. If a bus travels between the bus stops, the program prints the number of the bus that departs from the initial bus stop, possible bus switching time at the central station and the number of the bus leaving from the central station. Finally, the arrival time at the destination bus stop is printed. **The program prints in the console window only if the first command line argument is `-show` or if command line arguments are defective; otherwise the program prints only to a file.**

3.2 Output to a file

The program prints also to a text file one line containing the following:

1. If one can travel by bus between the input bus stops, the program prints the following items separated by space characters:
 - a. initial bus stop,
 - b. the number of the bus that is taken at the initial bus stop,
 - c. departure time at the initial bus stop,
 - d. **if there is a switch at the central station**, the letter X,
 - e. **if there is a switch at the central station**, the number of the bus that is taken at the central station,
 - f. **if there is a switch at the central station**, the departure time from the central station,
 - g. destination bus stop,
 - h. arrival time at the destination stop.
2. If it is not possible to travel by bus between the input bus stops, the program prints -1.
3. If the input file is erroneous or does not exist, the program prints -2.
4. If a runtime error occurs, the program prints -3.

If the output file exists, it is allowed to be written over without warning.

3.3 Examples of output to file

Let the timetable file be "table.txt" and let its contents be:

```
BLI 21a F C 9.15 12.15 0.30
BLT 18 A D 11.15 12.45 14.00 15.15
BXL 10 E 7.30 21.30 1.00
BLX 7 B 6.15 17.00 0.15
```

Input :

```
-quiet A2 D3 12.00 table.txt out.txt
```

Output to the file out.txt:

```
A2 18 12.50 D3 13.15
```

Explanation: The bus number 18 leaves from the end of the A line at 12.45 and arrives at the bus stop A2 at 12.50. The bus drives directly to the D line and arrives at the bus stop D3 at 13.15.

Input :

```
-quiet A2 E3 13.45 table.txt out.txt
```

Output to the file out.txt:

```
A2 18 14.05 X 10 14.30 E3 14.40
```

Explanation: The bus number 18 leaves from the end of the A line at 14.00 arrives at the bus stop A2 at 14.05. The bus number 10, leaving at 14.30, is taken on the central station. The bus arrives at the bus stop E3 at 14.40.

Input :

```
-quiet C4 C1 10.30 table.txt out.txt
```

Output to the file out.txt:

```
C4 21a 10.40 C1 10.55
```

Explanation: The bus number 21a leaves from the end of the F line at 10.15 and arrives at the central station at 10.35. This bus arrives at the bus stop C4 at 10.40 and at the bus stop C1 at 10.55.

Input :

```
-quiet B1 C1 16.40 table.txt out.txt
```

Output to the file out.txt:

```
B1 7 16.45 X 21a 9.35 C1 9.55
```

Explanation: The bus number 7 leaves from the end of the B line at 16.45 and arrives at the central station at 17.05. Then it would be necessary to stay the night on the central station waiting for the bus number 21a that leaves from the central station at 9.35 and arrives at the bus stop C1 at 9.55.

Input :

```
-quiet A1 B1 11.30 table.txt out.txt
```

Output to the file out.txt: -1

Explanation: No bus leaving from the central station drives along the B line.

4. Concerning implementation

Note that the fastest way to travel between two bus stops is obtained in a rather simple fashion. If one is traveling towards the central station, it is sufficient to check, which bus coming from the end of the line arrives next at the bus stop. Thereafter one checks (if necessary) after arriving at the central station, which bus leaves next along the destination line. If one is traveling from a bus stop on a line towards the end stop of the same line, one checks, which bus comes next from the central station.

The program must contain a class structure. **No procedural solutions are allowed.**

5. General requirements

The assignment is returned as a zipped file (that can be opened with pkzip) using Prolab's Puzzle system. The assignment must contain the following items:

1. readme.txt file with author information (name, department, faculty, student id and e-mail address) plus possibly some other important information for the inspector. At least the compiler that was used during development shall be mentioned.
2. Source code with appropriate documentation. The form of the documentation is described below.

5.1 Documentation

In the beginning of each file, a comment of the following type shall be written:

```
/*
 * File: the name of the file
 * Author: the name of the author
 * Created: creation date
 *
 * Short description of the contents of the file.
 */
```

Prior to each method (and function) definition in the code files (*.cpp), a comment of the following type shall be written:

```
/*
 * Method/function: the name of the method/function
 *
 * Short description of operation of the method/function.
 * Parameter list in the form:
 * @param type name meaning
 * @param type name meaning etc.
 *
 * @return return value type and meaning
 */
```

5.2 Concerning the inspection of the assignment

First, the inspectors compile the code, if possible, using the same compiler as the author. Naturally, the code must compile.

The inspectors execute the program with both adequate and erroneous inputs. The program should react sensibly to erroneous inputs: the program may not crash or get into an eternal loop. Erroneous inputs must be informed to the user. On the course web page, there will be an input file and some example inputs with correct outputs to test the assignment program.

The inspectors also check the program for memory leaks.

6. Conclusion

The list below contains the most important items that should be taken into account:

1. The program is run by typing in the command prompt

```
<program_name> <-show | -quiet> <initial bus stop> <destination bus stop > <time>  
<input_file_name> <output_file_name>
```

2. Program prints, when the option is `-show`, in the console window route information for the fastest route, if one can travel by bus between the input bus stops.
3. Program prints in a text file route information for the fastest route, if one can travel by bus between the input bus stops. Otherwise it prints in the text file an error code that was described in the section "Output of the program".
4. The assignment must contain `readme.txt` file and source code with previously described comments.
5. The code must compile.
6. The program is executed with correct and erroneous inputs.
7. The program is checked for memory leaks.