Station Machination

This assignment contributes 10% to your final grade and is due on Sunday 21st of March at 11:59 PM AEDT.

Problem Description

You have been given the task to create a program that will find the next earliest train departure after your arrival at the station. This program will notify the user of the next train departure they should board using information from a timetable containing multiple entries of the starting station, destination station and the time the train is departing the starting station. The timetable will be in the form of a text file which will contain the source, destination and departure times of all trains on the rail network. The timetable information will be read by your program from standard input. The user will provide command line arguments, providing the source, destination and time of arrival at the source station.

Program Structure

Timetable Data

The train network timetable will be presented in a double colon separated format. The format will specify the source, destination and leaving time from the source station.
The source and destination component may contain spaces and will be sub-strings of the line entry. Each entry in the timetable is separated by a new line. Following the `<time>`, a new line character is expected. Each entry will not exceed the maximum number of characters (4096), however, in the event your submission encounters an entry greater than the maximum string length, your program must skip this entry.

The data is formatted in the following form.

```
<source>::<destination>::<time>
```

The following is an example schedule.

```
Paterson::Werris Creek::23:22:23
Paterson::Werris Creek::00:01:23
Paterson::Werris Creek::00:55:23
Paterson::Lochinvar::01:06:11
Paterson::Lochinvar::02:01:11
Paterson::Lochinvar::02:56:11
Condobolin::Wangaratta::11:56:26
Condobolin::Wangaratta::12:01:26
Condobolin::Wangaratta::12:58:26
Condobolin::Maitland::09:30:22
Condobolin::Maitland::10:01:22
Wangaratta::Maitland::10:58:22
```

Your submission must be able to handle erroneous data that is contained within the schedule. This may include entries that contain an inappropriate number of colons or breaks, missing source, destination and time information as well as invalid time formats.

To clarify, You do not need to handle intermediate stations

Standard Input

Your program will need to accept the train network schedule via standard input. The data will be pushed into standard input via redirection and you will be able to use any allowed function that can operate directly on standard input such as `fgets`,`fscanf`, `fread`.

```
./timetable < timetable.csv
```

Do note, your program cannot hold all the data from the file in main memory as per the restrictions of this assignment, you will need to process it, line by line. If no data is supplied, your program should output to standard error `No timetable to process`.

You should only read through the file once, your program should not attempt to re-read the data after it has been read. Please do not attempt to use `rewind` or `fseek` on standard input.

Command Line Arguments

The search criteria is passed to the program via command line arguments, your program will need to accept all three command line arguments or it should otherwise quickly exit and output how to use the program. Refer to to the **Not Enough Arguments** section.

The command line arguments are given in the format and order:

```
./timetable <source> <destination> <time>
```

The following is an example of program execution with command line arguments.

```
./choochoo Sydney Melbourne "11:59:59" < timetable.csv
The next train to Melbourne from Sydney departs at 12:05:40
```

The time format must be supplied in 24 hour format and in the following format `<hh>:<mm>:<ss>`, where `hh` represents the hour, `mm` represents the minutes and `ss`, the number of seconds.

Examples

Each example will contain the timetable contents, specified as timetable.list, program execution

Single Station 1

timetable.list

```
Wirragulla::Cardiff::09:38:23
Wirragulla::Cardiff::10:01:23
Wirragulla::Cardiff::10:30:23
Wirragulla::Cardiff::10:59:23
Wirragulla::Cardiff::11:01:23
```

```
./timetable Wirragulla Cardiff "10:20:05" < timetable.csv
The next train to Cardiff from Wirragulla departs at 10:30:23
```

Single Station 2

timetable.list

```
Coffs Harbour::Morisset::19:09:14
Coffs Harbour::Morisset::19:54:14
Coffs Harbour::Morisset::20:01:14
Coffs Harbour::Morisset::20:46:14
Coffs Harbour::Koolewong::19:43:40
Coffs Harbour::Koolewong::20:01:40
```

```
./timetable "Coffs Harbour" Koolewong "19:45:15" < timetable.list
The next train to Koolewong from Coffs Harbour departs at 20:01:40
```

Multiple Stations

timetable.list

```
Sydney::Melbourne::08:05:00
Sydney::Melbourne::09:32:45
Sydney::Melbourne::11:45:32
Sydney::Melbourne::13:12:19
Sydney::Melbourne::19:59:15
Sydney::Melbourne::22:32:21
Melbourne::Sydney::14:13:12
Melbourne::Sydney::16:55:12
Melbourne::Sydney::18:33:12
Melbourne::Sydney::20:25:12
Melbourne::Sydney::22:22:12
```

```
./timetable Sydney Melbourne "19:40:59" < timetable.list
The next train to Melbourne from Sydney departs at 19:59:15
```

Missing Station

timetable.list

```
Sydney::Melbourne::08:05:00
Sydney::Melbourne::09:32:45
Sydney::Melbourne::11:45:32
Sydney::Melbourne::13:12:19
Sydney::Melbourne::19:59:15
Sydney::Melbourne::22:32:21
Melbourne::Sydney::14:13:12
Melbourne::Sydney::16:55:12
Melbourne::Sydney::18:33:12
Melbourne::Sydney::20:25:12
Melbourne::Sydney::22:22:12
```

```
./timetable Sydney Brisbane "19:40:59" < timetable.list
No suitable trains can be found
```

Not enough arguments

```
./timetable Sydney Brisbane < timetable.list
Please provide <source> <destination> <time> as command line arguments
```

Restrictions

Below are a list of restrictions on your submission. An aim of this assignment is to efficiently use memory and construction an application within these fixed boundaries.

- You are not allowed to use any dynamic memory of any kind, this includes any alloc family functions, brk, sbrk and mmap. You are not allowed to use any temporary files or or variable length arrays. Static and global variables need to be rationalised when using them.
- You are not allowed to use any function that belongs in string.h directly.
- You are allowed to reimplement any function that you want to use in string.h such as finding the length of a string, tokenising a string or copying a string from one location to another.
- You can safely assume the largest line is 4096 characters.
- You cannot use static or global memory, utilise the preprocessor #define mechanism for your constants.
- Your submission must successfully compile with the submission system's compilation command.
- Your program will be compiled using the provided Makefile. You can make modifications to the file, but cannot remove flags from `CFLAGS`, or change the compiler being used.

If your submission violates any of these restrictions, your submission will receive 0 marks.

Error Handling and Test Cases

The public test cases will only include valid data and you will need to ensure that your program can handle invalid test data. You are tasked with developing your own set of test cases that will need to test both valid input and invalid input. You must document your test cases in a file named `testinfo.txt`, this file must contain a brief description of each test case you have created. Please keep `testinfo.txt` in the root of your assessment repository. The `make test` command must successfully execute your test cases and provide human readable output to the user.

Consider the following scenarios when testing your code.

- Error cases and return codes
- Timetable file that can exceed stack memory limit
- Cross-day times (when the closest time is the next day, candidate train may be available on the next day)
- Time format
- Lines longer than the maximum limit
- Unsorted data

Code Submission

Your code submission must be made via git. You are able to retrieve the git repository link from the code submission section. The following rules apply to your code submission:

- Only the last submission will be graded.
- Submission after the due date will incur a late penalty as described in the unit of study outline.
- Your submission must be able to compile with the given compilation command (using `make`)
- **libasan** is used in the compilation of your submission. This will check your program for memory related errors that occur at runtime.

Marking Criteria

The following is the marking breakdown, each point contributes a portion to the total 10% of the assignment. You will receive a result of zero if your program fails to compile.

- 3% Test Cases - Your program must pass public, private and hidden test cases to achieve the maximum number of points awarded for this section.
- 6% Solution Discussion - You will need to answer questions from a COMP2017 teaching staff member regarding your implementation. You will be required to attend a zoom session with COMP2017 teaching staff member after the code submission deadline. A reasonable attempt will need to be made, otherwise you will receive zero for the assessment.

  In this session, you will be asked to explain:

  - How your code identifies invalid entries of `timetable.list` input.
  - How your code determines the answer to a query for multiple stations and why it is optimal and correct.
  - Answer further questions.
  - Your code will also be assessed on C coding style conventions (see Ed resources). Clean code will attract the best grade.
- 1% Your test cases - Your solution must include a suite of test cases that should be automated and executable with the given make script. Please make sure your test suite outputs the result of each test case in a human readable format.

Academic Declaration

By submitting this assignment, you declare the following:

I declare that I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure, and except where specifically acknowledged, the work contained in this assignment/project is my own work, and has not been copied from other sources or been previously submitted for award or assessment.

I understand that failure to comply with the Student Plagiarism: Coursework Policy and Procedure can lead to severe penalties as outlined under Chapter 8 of the University of Sydney By-Law 1999 (as amended). These penalties may be imposed in cases where any significant portion of my submitted work has been copied without proper acknowledgement from other sources, including published works, the Internet, existing programs, the work of other students, or work previously submitted for other awards or assessments.

I realise that I may be asked to identify those portions of the work contributed by me and required to demonstrate my knowledge of the relevant material by answering oral questions or by undertaking supplementary work, either written or in the laboratory, in order to arrive at the final assessment mark.