Programming and Algorithms (COMP1038) Coursework 2 Specification

Programming and Algorithms Teaching Team

Nov 23, 2023

CONTENTS

CHAPTER

ONE

INTRODUCTION

This is the second Programming and Algorithms (COMP1038) Coursework. It is worth **40% of the module mark**. It requires you to write a program which will calculate the costs of train journeys between cities. The deadline for this exercise is **16:00 on Thursday 14th of December 2023**.

Read the entire document before beginning the exercise.

If you have any questions about this exercise, please ask in the Q&A forum on Moodle, after a lecture, in a lab, or during the advertised office hours. Do not post your program or parts of your program to Moodle as you are not allowed to share your coursework programs with other students. If any questions require this exercise to be clarified then this document will be updated and everyone will be notified via Moodle.

CHAPTER

TWO

VERSION HISTORY

Warning: The content of this file may change in the future, please always refer to the latest version on Moodle.

- Version 1.0 - 2023-11-23 - Original version.

CHAPTER

THREE

SUBMISSION

You must submit a single C source code file containing all your code for this exercise. This file must be called `train.c` and must not require any other files outside of the standard C headers which are always available. The first line of the file should be a comment that contains your student ID number, username, and full name, of the form:

```
// 6512345 zy12345 Joe Blogs
```

The file must compile without warnings or errors using the command

```
gcc -std=c99 -lm -Wall -fsanitize=leak train.c -o train
```

This command will be run on our Linux server CSLinux. If it does not compile, for any reason, then you will lose all the marks for testing (common reasons in the past have been submitting a file with the wrong filename, or developing your solution on your personal computer without having tested it on our Linux server). If the file compiles but has warnings then you will lose some marks for not correcting the warnings.

The completed source code file should be uploaded to the Coursework 2 Submission link on the COMP1038 Moodle page. You may submit as many times as you wish and the last submission will be used for marking. However, if you submit after the deadline, your last submission time will be considered for the Late Submission penalty. Do not wait until the last moment to submit the coursework. Equipment occasionally breaks down, is full, inaccessible, or unreliable. You need to plan ahead to allow time for foreseeable things outside of your control to go wrong.

Late submissions: COMP1038 late submission policy is different from the standard university policy. Late submissions will lose 2 percentage points **per hour**, rounded up to the next whole hour. This is to better represent the large benefit a small amount of extra time can give at the end of a programming exercise. No late submissions will be accepted more than 50 hours after the exercise deadline. If you have extenuating circumstances you should file them before the deadline.

CHAPTER

FOUR

PLAGIARISM

You should complete this coursework on your own. Anyone suspected of plagiarism will be investigated and punished in accordance with the university policy on plagiarism (see your student handbook and the University Quality Manual). This may include a mark of zero for this coursework.

You should write the source code required for this assignment yourself. If you use code from other sources (books, web pages, etc), you should use comments to acknowledge this (and marks will be heavily adjusted down accordingly). *The only exception to this is the example programs given in lectures and tutorials; you may use them, with or without modification, without penalty.*

You must not copy or share source code with other students. You must not work together on your solution. You can informally talk about higher-level ideas but not to a level of detail that would allow you all to create the same source code.

A strong warning to the students against using ChatGPT or other AI tools. Firstly, because it is an academic offense and, secondly, in any case, ChatGPT is not reliable and does not always give the correct answer. Copying code from other students, from previous students, from any other source (including ChatGPT or other AI tools), or soliciting code from online sources and submitting it as your own is plagiarism and will be penalized as such. This can potentially result in failure of coursework, module, or degree.

Remember, **it is quite easy for experienced lecturers to spot plagiarism in source code**. If you are having problems you should ask questions rather than plagiarize. If you are not able to complete the exercise then you should still submit your incomplete program as that will still get you some of the marks for the parts you have done (but make sure your incomplete solution compiles and partially runs!).

CHAPTER

FIVE

MARKING

The marking scheme will be as follows:

- **Testing (90%)**: Your program should correctly implement the task requirements. A number of tests will be run against your program with different input data designed to test if this is the case for each individual requirement. The tests themselves are secret but general examples of the tests might be:
  - Does the program work with the example I/O in the question?
  - Does the program work with typical valid input?
  - Does the program correctly deal with input around boundary values?
  - Does the program correctly deal with invalid values?
  - Does the program handle errors with resources not being available (eg, malloc failing or a filename being wrong)?
  - Does the program output match the required format?
  - Your program is required to use a graph representation of the train data. Is that implementation designed correctly? Is the choice of data structure and algorithm appropriate, correct, and efficient? This is assessed separately from the tests and general language features sections to focus specifically on your understanding and implementation of the relevant data structures and algorithms.

As noted in the submission section, if your program does not compile then you will lose all testing marks (90% marks). Also if you submit a different type of file apart from a single C source code file containing all your code for this exercise and the file name is different from `train.c` then you will also lose all testing marks (90% marks). We usually use an automatic marking system to test/mark your coursework submissions. So you should strictly follow the output format specified in this task description while implementing your program. Otherwise, your program will fail to test and you will lose marks.

- **Source code formatting (10%)**: Your program should be correctly formatted and easy to understand by a competent C programmer. This includes but is not limited to, indentation, bracketing, variable/function naming, and use of comments. See the lecture notes and the example programs for examples of correctly formatting programs.

Late Submissions: see the submission section above.

CHAPTER

SIX

QUESTION

You are required to write a program that will produce the optimal route and cost C of train tickets between stations.

To complete this task, you will be given a source station S, a destination station D and a distance matrix (in kilometers) for N stations.

The cost C is defined as:

$$C = \lceil 1.2 * d + 25 * n \rceil$$

where d is the total distance of the route, n is the number of intermediate stations (excluding source and destination stations), and $\lceil x \rceil$ means x should be rounded up to the next nearest integer.

You should choose an appropriate and efficient algorithm to obtain the shortest route from station S to D. If there are more than one shortest route, the route with minimal cost C should be used.

Note: Connections do not have to be symmetric, ie, there may be pairs of stations where you can travel from A to B but not B to A.

6.1 Input

The first line of the input contains N strings of station names, separated by comma `,`.

The following N lines is the distance matrix, and each line contains N cells that are separated by comma `,`.

The last line of the input contains two strings of source and destination station names S, and D, separated by comma `,`.

The following rules for lines and cells are held:

- There may or may not be a value for each cell.
- Cells without values have nothing between commas or between the start/end of the line and the comma `,`.
- Each value can contain any number of any printable ASCII characters, excluding the comma `,` and newline `\n` characters.
- The cell is considered valid if the value contains only a positive integer (e.g. `1`, `22` and `333`) or nothing.
- Valid value in j-th cell of i-th line (excluding the first line) represents the distance from station i to j, or there is no direct connection between those two stations the value is empty.

6.2 Output

The output should have two lines:

- The first line should contain the total distance and the price of the ticket, separated by comma `,`;
- The second line should contain the sequence of the optimal route, also separated by comma `,`.

The following case-sensitive strings (with character `.`) should be the output when satisfy the corresponding condition:

Invalid distance matrix.
Condition: Failed to parse the distance matrix from raw input, or any cell in the distance matrix violates the rules described in *Section Input*.

Priority: 5

Invalid source station.
Condition: Failed to parse S from raw input, or the source station does not exist.

Priority: 4

Invalid destination station.
Condition: Failed to parse D from raw input, or the destination station does not exist.

Priority: 3

No journey, same source and destination station.
Condition: If the source and destination stations are the same.

Priority: 2

No possible journey.
Condition: If there is no possible journey between the stations.

Priority: 1

Note: If multiple invalid conditions are satisfied, please only output the one with the highest priority.

For example, in Sample 7, *Invalid source station.*, *Invalid destination station.* and *No journey, same source and destination station.* are satisfied, you should only output *Invalid source station.*

6.3 Sample Inputs and Outputs

Input 1	output 1
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Ningbo,Suzhou	425,560 Ningbo,Hangzhou,Shanghai,Suzhou

Input 2	output 2
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Ningbo,Ningbo	No journey, same source and destination␣ ↪station.

Input 3	output 3
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Glasgow,341ed admom1 q!!!!	Invalid source station.

Input 4	output 4
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Nanjing,Glasgow	Invalid destination station.

Input 5	output 5
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Wenzhou,Fuzhou	325,390 Wenzhou,Fuzhou

Input 6	output 6
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155a11!,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Wenzhou,Fuzhou	Invalid distance matrix.

Input 7	output 7
Ningbo,Hangzhou,Suzhou,Changzhou, ↪Shanghai,Taizhou,Wenzhou,Jinhua, ↪Fuzhou,Nanjing ,155,,,,380,,,, 155,,,210,180,,,180,,280 ,,,95,90,,,,, ,210,95,,,,,,,130 ,180,90,,,,,,, 380,,,,,,610,,, ,,,,,610,,235,325, ,180,,,,,235,,, ,,,,,,325,,, ,280,,130,,,,,, Glasgow,Glasgow	Invalid source station.

Input 8	output 8
A,B,C,D,E ,1,2,, 1,,,, 2,,,, ,,,,3 ,,,3, A,E	No possible journey.

6.4 Sample Files and Command-line Usages

To compile the C code:

```
gcc -std=c99 -lm -Wall -fsanitize=leak train.c -o train
```

To test the program with the sample files:

```
./train < 1.in
```

Then check the output is exactly the same as the content in the corresponding `.out` files.

To detect memory leak:

```
valgrind --leak-check=full ./train < 1.in
```

Note:

- < is used to redirect the input from your keyboard to a given file.
- `1.in` is a file containing the input in the same folder as the program, and it may be replaced by other file names in testing and marking. You can replace it with other `.in` files such as `2.in` as long as it exists.

CHAPTER

SEVEN

HINTS

- Remember to free any memory which you no longer need. Your program should not have any memory leaks (dynamically allocated areas of memory that are no longer reachable). You will need to consider how the responsibility for allocated data transfers as your program runs.

CHAPTER

EIGHT

SOURCE CODE SUBMISSION GUIDE

8.1 Keywords

The keywords **MUST**, **MUST NOT**, **REQUIRED**, **SHALL**, **SHALL NOT**, **SHOULD**, **SHOULD NOT**, **RECOMMENDED**, **MAY**, and **OPTIONAL** in this guide are to be interpreted as follow:

MUST
: This word, or the terms **REQUIRED**, **SHOULD** or **SHALL**, mean that the instruction is an absolute requirement of this guide.

MUST NOT
: This phrase, or the phrases **SHOULD NOT** or **SHALL NOT**, mean that the instruction is an absolute prohibition of this guide.

MAY
: This word, or the adjective **OPTIONAL**, mean that the instruction is truly optional.

8.2 Definitions

`standard input`
: This mark, or the mark `System.in` or `stdin`, mean that the stream from which input to the program is taken. Typically this is the keyboard, but it can be specified that input is to come from a serial port or a disk file, for example.

`standard output`
: This mark, or the mark `System.out` or `stdout`, mean that the stream to which output from the program is sent. Typically this is a display, but it can be redirected to a serial port or a file.

`standard error`
: This mark, or the mark `System.err` or `stderr`, mean that the stream to which error output from the program is sent. Typically this is a display, but it can be redirected to a serial port or a file.

`<empty line>`
: This mark, or the marks `\n`, `\r\n`, `<LF>` or `<CR><LF>`, mean the character `\n` after the last non-whitespace character in the previous content.

  For example, the following representations are all equivalent:

<empty line>	\n
first second third last <empty line>	first\nsecond\nthird\nlast\n

8.3 Technical Notes

This part contains important technical information and it is important that you read and understand all the information below.

- You program **MAY** have multiple classes if you wish, but **only** in one C file for each question.
- Your program **MUST** read its input from `standard input`.
- Your program **SHOULD** send its output to `standard output`. Your program may also send output to `standard error`, but **only** output sent to `standard output` will be considered during judging.
- If your program exits with a non-zero exit code, it will be judged as a run-error.
- Program submitted will be run inside a `sandbox`.
  - The sandbox will allocate **2GB** of memory for your program. Your entire program, including its runtime environment, must execute within this memory limit.

8.3.1 Command-line Usage for C

- Compile:

```
gcc -std=c99 -lm -Wall -fsanitize=leak train.c -o train
```

- Execute (for testing):

```
train < 1.in
```

8.4 Submissions Judging Process

The judging system is fully automated. Judging is done in the following way:

8.4.1 Submitting solutions

You should submit all related files to Moodle according to the `Coursework Issue Sheet`.

8.4.2 Compilation

Your program will be compiled on **CSLinux**. All submitted source files will be passed to the compiler which generates a single program to run.

8.4.3 Testing

After your program has compiled successfully it will be executed and its output is compared to the output of the judges. Before comparing the output, the exit status of your program is checked: if your program exits with a non-zero exit code, the result will be a run-error even if the output of the program is correct! There are some restrictions during execution. If your program violates these it will also be aborted with a run-error, see *restrictions*.

When comparing program output, it has to exactly match to output of the judges. So take care that you follow the output specifications. In case of problem statements which do not have unique output (e.g. with floating point answers), the system may use a modified comparison function. This will be documented in the problem description.

8.4.4 Possible Results

A submission can have the following results (not all of these may be available depending on configuration of the system):

CORRECT
: The submission passed all tests: you solved this problem!

COMPILER-ERROR
: There was an error when compiling your program. On the submission details page you can inspect the exact error (this option might be disabled). Note that when compilation takes more than 30 seconds, it is aborted and this counts as a compilation error.

TIMELIMIT
: Your program took longer than the maximum allowed time for this problem. Therefore it has been aborted. This might indicate that your program hangs in a loop or that your solution is not efficient enough.

RUN-ERROR
: There was an error during the execution of your program. This can have a lot of different causes like division by zero, incorrectly addressing memory (e.g. by indexing arrays out of bounds), trying to use more memory than the limit, reading or writing to files, etc. Also check that your program exits with exit code 0!

NO-OUTPUT
: Your program did not generate any output. Check that you write to standard out.

OUTPUT-LIMIT
: Your program generated more output than the allowed limit. The solution is considered incorrect.

WRONG-ANSWER
: The output of your program was incorrect. This can happen simply because your solution is not correct, but remember that your output must comply exactly with the specifications of the judges. See *testing* below for more details.

The judges may have prepared multiple test files for each problem.

8.4.5 Restrictions

Submissions are run in a sandbox to prevent abuse, keep the system stable and give everyone clear and equal environments. There are some restrictions to which all submissions are subjected:

compile time
Compilation of your program may take no longer than 30 seconds. After that, compilation will be aborted and the result will be a compile error.

source size
The total amount of source code in a single submission may not exceed 256 kilobytes, otherwise your submission will be rejected.

memory
The judges will specify how much memory you have available during execution of your program. This may vary per problem. It is the total amount of memory (including program code, statically and dynamically defined variables, stack)! If your program tries to use more memory, it will most likely abort, resulting in a run error.

creating new files
Do not create new files. The sandbox will not allow this and the file open function will return a failure. Using the file without handling this error can result in a runtime error.

number of processes
You are not supposed to explicitly create multiple processes (threads). This is to no avail anyway, because your program has exactly 1 processor core fully at its disposal.

People who have never programmed with multiple processes (or have never heard of threads) do not have to worry: a normal program runs in one process.

internet access
Your programs are not allowed to access the Internet. Any attempts to access the Internet from your programs will result in a run-error.

8.5 Code Example

Below is the example on how to read input and write output for a problem.

The example is solution for the following problem:

> The first line of the input contains the number of testcases. Then each testcase consists of a line containing a name (a single word) of at most 99 characters. For each testcase output the string `Hello <name>!` on a separate line.

Sample input and output for this problem:

Input	output
3 world Jan SantaClaus	Hello word! Hello Jan! Hello SantaClaus!

Note: The number 3 on the first line indicates that 3 testcases follow.

Listing 1: A Solution in C

```c
#include <stdio.h>

int main()
{
    int i, ntests;
    char name[100];

    scanf("%d\n", &ntests);

    for (i = 0; i < ntests; i++)
    {
        scanf("%s\n", name);
        printf("Hello %s!\n", name);
    }
}
```