MTRN4110 21T2 Phase B Task Description (Week 4-6)

Updated 28/6/2021:

- Added to 3.5 requirement on generating “Output.txt” file storing all the messages printed to console
- Updated folder structure in 5.1 with “Output.txt” added
- Updated marking criteria in 5.2 by replacing mark addition with mark deduction

Updated 23/6/2021:

- Updated folder structure in 5.1 to avoid confusion

Updated 22/6/2021:

- Added “and the target cell” and “’x’ - target” to 3.1
- Added “The use of a dynamic data structure (e.g., vector in C++, dynamically allocated array in C) is recommended.” to 4.2
- Corrected an error in 5.2 (the world file and the motion plan file -> the map file)

Released 22/6/2021

1. Overview of the Course Project:

The main project of MTRN4110 21T2 is a simulation-based project adapted from the Micromouse competition. Webots will be used as the simulation platform throughout the project. You will design a mobile robot and implement a controller and a vision program to negotiate a maze autonomously in Webots. The project will contribute 55% to your final mark of this course.

The project consists of four sequential phases, which are connected but attempting one phase is not dependent on the completion of another:

- Phase A: Driving and Perception (week 1-3, 12%, individual)
- Phase B: Path Planning (week 4-6, 12%, individual)
- Phase C: Computer Vision (week 7-9, 12%, individual)
- Phase D: Integration and Improvement (week 10-11, 19%, group)

This document will describe the tasks of Phase B.

2. Overview of Phase B – Path Planning:

Phase B aims to develop a path planning module for the maze-solving robot. You are required to complete the tasks of this phase by 17:00 Monday Week 7.

2.1. Expectations:

By the end of Phase B, you are expected to have been able to:

- read in a map containing a maze layout, a robot's initial location and heading, and its target position from a text file;
- understand how to represent such a map in a program;
- implement a path planning algorithm to find an optimal path for the maze-solving robot;
- generate a sequence of motion commands and write it into a text file.

2.2. Learning Outcomes Associated with this Assignment:

- **LO1**: Apply relevant theoretical knowledge pertaining to mobile robots, including locomotion, perception and localisation using onboard sensors, navigation and path planning, for practical problem-solving
- **LO3**: Demonstrate practical skills in mechatronics design, fabrication, and implementation

3. Phase B Task Description:

You are supposed to develop a new controller for the robot E-puck used in Phase A. The controller should plan an optimal path for the robot given a layout of the maze and the robot's initial state and final target.

Technically, the implementation can be done independently of Webots as no robotic simulation is involved in this phase. However, for the sake of easy integration in Phase D and convenience of assessment, you must implement the controller in Webots (refer to Section 4.1).

The controller should complete the following tasks once started.

3.1. Read in a map from a text file and display it in the console

You will be given a text file named "Map.txt" containing a map of the maze layout, the robot's initial location and heading, and its target position, e.g.,

Fig. 1. An example map in the text file containing the maze layout, the robot's initial location and heading, and its target position

As shown in Fig. 1:

- each horizontal wall of the maze is described by three dashes "---";
- each vertical wall of the maze is represented by a vertical bar "|";
- the location and heading of the robot is indicated by a sign ("^" - north, ">" - east, "v" – south, "<" - west);
- the target position is expressed by a character "x" (note that we don't restrict the robot's heading at the target position in this phase; any orientation (N, E, S, or W) is acceptable).

Note that we refer to the top, bottom, left, and right of the maze throughout this phase as North, South, West, and East, respectively. All the characters that could appear in the text file are illustrated as below, except for spaces and newlines (only lowercase v and x will be used).

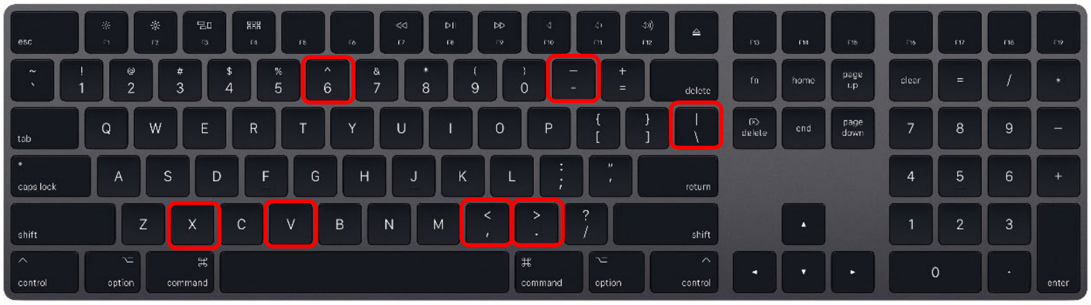

Fig. 2. All the characters that could appear except for spaces and newlines (only lowercase v and x will be used)

All the empty cells are represented by 3 spaces "   ". For the cell that has a robot in it <u>and the target cell</u>, it is described by 1 space plus 1 sign ("^" - north, ">" - east, "v" - south, "<" – west, <u>"x" - target</u>) followed by 1 space, making a total of 3 characters as well.

All the empty walls (horizontal or vertical) are expressed by spaces, and the number of characters always match when there is a wall in place.

All the square blocks between adjacent walls are also depicted by spaces.

You should read the information from the text file into your controller and print the map into the console. For example,

```
[z1234567_MTRN4110_PhaseB] Reading in map from ../../Map.txt...
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- --- 
[z1234567_MTRN4110_PhaseB] | v             |                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           |           |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | x                 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- --- 
[z1234567_MTRN4110_PhaseB] Map read in!
```

Fig. 3. Reading map into the program from a text file

If you find difficulty implementing reading information from a text file, you can choose to hard-code the map into your program and forfeit the marks associated with it. In this case, you must define a variable to store the map at the beginning of your program (so that a tutor can replace it when assessing your work). You should also explicitly indicate you are hard-coding the map in the Header Comment of your program. Failing to do so would affect the assessment of your submission (incurring a penalty).

In the Header Comment, you should also indicate the platform (Windows/MacOS/Linux) you used to develop your code.

```
/*
 * File:          z1234567_MTRN4110_PhaseB.cpp
 * Date:          XX/XX/XX
 * Description:   Controller of E-puck for Phase B - Path Planning
 * Author:        XXX
 * Modifications:
 * Platform:      Windows (or MacOS or Linux)
 * Notes:         The map is had-coded into the program.
 */
```

Fig. 4. Explicitly indicate hard-coding in the Header Comment of your program

3.2. Find all the shortest paths for the robot from its initial location to the target position

After reading the map, you should run a path planning algorithm to find all the shortest paths from the initial location of the robot to the target position.

Here the shortest paths mean the paths in which the number of cells that the robot will traverse is the smallest among all the possible paths.

For the example above, there are 6 such shortest paths with which the number of cells traversed is 17 (and no other paths can achieve a number smaller than 17).

Display all the paths in the console. Note that:

- the order of the paths printed does not need to match the following;
- no duplicate paths should be displayed;
- the map should be in an exact same format as below (the target cell should have an index 0; the initial cell should have a sign as in the original map; a two-digit index should align to the right).

```
[z1234567_MTRN4110_PhaseB] Map read in!
[z1234567_MTRN4110_PhaseB] Finding shortest paths...
[z1234567_MTRN4110_PhaseB] Path - 1:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 2:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v             |                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] | 15  14    |           |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |     13|   |   | 0   1   2   3     |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   | 12  11  10  9     | 5   4     |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           | 8   7   6 |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 3:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12        |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   | 11| 0   1   2   3     |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |         10  9     | 5   4     |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           | 8   7   6 |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
```

```
[z1234567_MTRN4110_PhaseB] Path - 4:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|     9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                     ---
[z1234567_MTRN4110_PhaseB] |           | 12  11  10|   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---         ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 5:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7     |
[z1234567_MTRN4110_PhaseB]      ---                     ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   | 6   5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---         ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 6:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|     9   8   7     |
[z1234567_MTRN4110_PhaseB]      ---                     ---
[z1234567_MTRN4110_PhaseB] |           | 12  11  10|   | 6   5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---         ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] 6 shortest paths found!
```

Fig. 5. Find all the shortest paths

3.3. Find the shortest path with the least turns and generate a motion sequence

Compare all the found shortest paths and find the one with the least turns, i.e., the one with which the robot needs the minimum number of turns.

If there is more than one such path existing, you can select one arbitrarily, e.g., the first one, as the found path with the minimum turns.

Generate a sequence of motion commands as defined in Phase A and calculate the total number of steps needed.

Display the found path, the total number of steps, and the motion sequence in the console.

For the example above, both Path - 1 and Path - 4 have 7 turns, while the other paths have 9 turns. You can select either Path - 1 or Path - 4 as the optimal path and generate the motion sequence (here Path - 1 is chosen):

```
[z1234567_MTRN4110_PhaseB] 6 shortest paths found!
[z1234567_MTRN4110_PhaseB] Finding shortest path with least turns...
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Shortest path with least turns found!
[z1234567_MTRN4110_PhaseB] Path Plan (23 steps): 00SLFFFRFLFLFRFFFFRFFRFFFF
```

Fig. 6. Find the shortest path with the least turns and generate a motion sequence

3.4. Write the found path to a text file

Write the generated motion sequence to a text file named "PathPlan.txt". Note the generated motion sequence must follow the conventions defined in Phase A. A motion sequence using a different convention (e.g., using "L" to represent a motion "turn left" plus a motion "forward"), even if it is essentially correct, will not get the marks associated with this task.

Display a message in the console showing this task is done once the motion sequence is written into the text file:

```
[z1234567_MTRN4110_PhaseB] Path Plan (23 steps): 00SLFFFRFLFLFRFFFFRFFRFFFF
[z1234567_MTRN4110_PhaseB] Writing path plan to ../../PathPlan.txt...
[z1234567_MTRN4110_PhaseB] Path plan written to ../../PathPlan.txt!
```

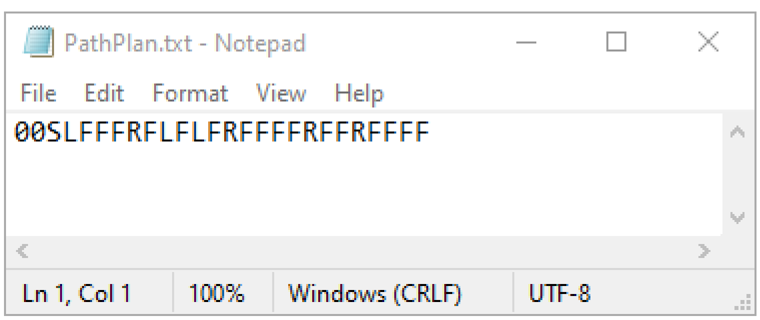

Fig. 7. Motion sequence written into "PathPlan.txt"

Overall, the messages printed in the console for the example should be exactly the same as shown below. Also note that each message should have a prefix "[z1234567_MTRN4110_PhaseB]" where z1234567 is replaced with your zID. Using a different format (except for a different order of paths), even essentially correct, will only get half of the full marks associated with them.

```
[z1234567_MTRN4110_PhaseB] Reading in map from ../../Map.txt...
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v             |                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           |           |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | x                 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Map read in!
[z1234567_MTRN4110_PhaseB] Finding shortest paths...
[z1234567_MTRN4110_PhaseB] Path - 1:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 2:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v             |                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] | 15  14    |           |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |     13|   |   | 0   1   2   3     |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   | 12  11  10  9     | 5   4     |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           | 8   7   6 |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 3:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|                   |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12        |   |       |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   | 11| 0   1   2   3     |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |         10  9     | 5   4     |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           | 8   7   6 |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
```

```
[z1234567_MTRN4110_PhaseB] Path - 4:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|     9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11  10|   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 5:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7     |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   | 6   5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Path - 6:
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13|     9   8   7     |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11  10|   | 6   5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] 6 shortest paths found!
[z1234567_MTRN4110_PhaseB] Finding shortest path with least turns...
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] | v   15  14  13| 10  9   8   7   6 |
[z1234567_MTRN4110_PhaseB]      ---                 ---
[z1234567_MTRN4110_PhaseB] |           | 12  11    |   |     5 |
[z1234567_MTRN4110_PhaseB]  ---             --- ---     ---
[z1234567_MTRN4110_PhaseB] |       |   |   | 0   1   2   3   4 |
[z1234567_MTRN4110_PhaseB]          ---     --- --- ---
[z1234567_MTRN4110_PhaseB] |   |                   |           |
[z1234567_MTRN4110_PhaseB]          ---         ---     ---
[z1234567_MTRN4110_PhaseB] |   |           |           |       |
[z1234567_MTRN4110_PhaseB]  --- --- --- --- --- --- --- --- ---
[z1234567_MTRN4110_PhaseB] Shortest path with least turns found!
[z1234567_MTRN4110_PhaseB] Path Plan (23 steps): 00SLFFFRFLFLFRFFFFRFFRFFFF
[z1234567_MTRN4110_PhaseB] Writing path plan to ../../PathPlan.txt...
[z1234567_MTRN4110_PhaseB] Path plan written to ../../PathPlan.txt!
```

Fig. 8. Messages printed to console

3.5. Task summary:

Task	Description
1	Read in a map from a text file and display it in the console
2	Find all the shortest paths for the robot from its initial location to the target position
3	Find the shortest path with the least turns and generate a motion sequence
4	Write the found path to a text file

Updated 28/6/2021:

In the tasks listed above wherever there is a requirement on printing a message to the console, you need to print the same message to a text file named "Output.txt", stored in the same folder as "Map.txt".

The messages printed to "Output.txt" should be EXACTLY the same as what is printed to the console, e.g., as in Fig. 8 for the example. An "Output.txt" has been added to the provided starting files for your reference.

You should define a path variable at the beginning of your controller file indicating the path of this text file.

```
const std::string OUTPUT_FILE_NAME = "../../Output.txt";
```

where the relative path ../../Output.txt will allow you to access the file if the folder structure specified in Section 5.1 is followed.

The "Output.txt" file is CRUCIAL for marking. Therefore, failing to generate a correct file that contains exactly the same messages as printed in the console will incur a SIGNIFICANT penalty.

4. Specifications and Hints:

4.1. Specifications:

Maze:

1. At the beginning of Phase B, you will be given a text file named “Map.txt”, in which the maze layout is the same as shown in the example.
2. This map is for your practice. For assessment, you may be tested with a different (but similar) map. The maze layout and the robot’s initial location and heading and its target position may be different from the example.
3. The maze will always be 5 rows by 9 columns with closed borders.
4. The initial location of the robot can be any of the cells.
5. The initial heading of the robot can be any of the four orientations (N, E, S, or W).
6. The target position can be any of the cells other than the initial location of the robot.
7. In the map text file given to you:
    - Each horizontal wall of the maze is represented by 3 dashes “---”.
    - Each vertical wall of the maze is represented by 1 vertical bar “|”.
    - Each empty cell is represented by 3 spaces “   ”.
    - Each empty horizontal wall is represented by 3 spaces “   ”.
    - Each empty vertical wall is represented by 1 space “ ”.
    - Each square block between adjacent walls is represented by 1 space “ ”.
8. There will always be a valid path in the given map.

Robot:

1. The initial location and heading of the robot are indicated by a sign (“^” - north, “>” - east, “v” – south, “<” - west). Note that throughout this phase we refer to the top, bottom, left, and right of the maze as North, South, West, and East, respectively. Only lowercase v is used.
2. The target position of the robot is indicated by “x”. Only lowercase x is used.
3. The orientation of the robot at the target position is not restricted.
4. On the left and right sides of the sign (“^”, “>”, “v”, “<”, or “x”) are 2 spaces, so 3 characters are used to represent a cell with a robot in it.
5. A turn is accounted when the robot needs to rotate at any cell along the path, including the starting cell.
6. Prioritise “L” over “R” when the two are equivalent, e.g., in a choice between “LL” and “RR”.

Implementation:

7. You must implement the controller in Webots using the same language as in Phase A (either C or C++).
8. For portability, you must use the built-in compilers for Windows or macOS.
9. The text file storing the map should be named “Map.txt”. You should define a path variable at the beginning of your controller program indicating the path of this text file, e.g.,

   const std::string MAP_FILE_NAME = “../../Map.txt”;

   where the relative path ../../Map.txt will allow you to access the file if the folder structure specified in Section 5.1 is followed.

10. The text file storing the found path plan should be named "PathPlan.txt". You should define a path variable at the beginning of your controller file indicating the path of this text file.

    const std::string PATH_PLAN_FILE_NAME = "../../PathPlan.txt";

    where the relative path ../../PathPlan.txt will allow you to access the file if the folder structure specified in Section 5.1 is followed.
11. The generated motion sequence must follow the conventions defined in Phase A. A motion sequence using a different convention (e.g., using "L" to represent a motion "turn left" followed by a motion "forward"), even if it is essentially correct, will not get the marks associated with this task.

4.2. Hints:

1. Consult the lecturer/demonstrators if you are unclear about anything.
2. A tutorial on the flood fill algorithm will be provided. However, you are free to use any path planning algorithms for the tasks.
3. The number of feasible paths/shortest paths/shortest paths with least turns in the assessment is not necessarily the same as in the example. No assumption on the maximum number of paths/length of the motion sequence should be made.
4. There is no simulation involved in this phase so you don't need to change the Webots world file. However, you will need to add the controller you developed to the robot so that you can run the path planning algorithm and see the results in the console.
5. You should use forward slash / or double backward slash \\ instead of single backward slash \ to define the path of the file.
6. The use of a dynamic data structure (e.g., vector in C++, dynamically allocated array in C) is recommended.

5. Assessment:

5.1. Submission of your work

You should zip your project folder and name it as "z*******_MTRN4110_PhaseB.zip" where z******* is your zID. Submit this zip file to Moodle.

In the folder, you should include both the <worlds> folder and the <controllers> folder.

In the <worlds> folder, you should include your world file, named as "z*******_MTRN4110_PhaseB.wbt" where z******* is your zID.

In the <controllers> folder, you should include your controller folder, named as "z*******_MTRN4110_PhaseB" where z******* is your zID.

You should have the following folder structure in your submission:

```
z*******_MTRN4110_PhaseB.zip
    |-- z*******_MTRN4110_PhaseB
        |--controllers
        |    |--z*******_MTRN4110_PhaseB
        |        |--build
        |        |--Makefile
        |        |--z*******_MTRN4110_PhaseB.cpp
        |        |--z*******_MTRN4110_PhaseB.exe
        |--worlds
        |    |--z*******_MTRN4110_PhaseB.wbt
        |--Map.txt
        |--Output.txt
        |--PathPlan.txt
```

These are the essential folders and files you should include. You can also include other files if needed, but you should not change the name/location of these folders and files. Besides, only the finalised version of your controller/code should be included. It is your responsibility to make sure your submission is self-contained and up-to-date.

As we have developed an automatic tool to do the plagiarism check on your code, it is crucial that you strictly follow the file structure specified above. Any violation in the submission format would incur a penalty, as that adds much more work to the assessors (we will still do the plagiarism check!).

5.2. Marking criteria:

This assignment will contribute 12% to your final mark.

You will be assessed five times with different setups Among them, one test will be on the example given to you for practice. Your final mark will be calculated as the following:

$$mark_{final} = (mark_{example} + mark_{newtest1} + mark_{newtest2} + mark_{newtest3} + mark_{newtest4}) / 5$$

Each attempt will be assessed by using the following criteria.

Task	Description	Marking (out of 100%)

1	Read in a map from a text file and display it in the console	+10% if all correct and zero marks if all wrong, otherwise
		• +10% by default
		  ○ -1% each for incorrectly displaying a line of the maze
		  ○ If the deduction above results in a mark < 2%, mark = 2% if at least one line was correctly displayed (excluding the outer walls)
		• (zero marks)
		  ○ If hard-coding the map into program

2	Find all the shortest paths for the robot from its initial location to the target position	+50% if all correct and zero marks if all wrong, otherwise
		• +50% by default
		  ○ -5% for missing a shortest path
		  ○ If the deduction above results in mark < 10%, mark = 10% if at least finding a shortest path, and mark = 5% if at least finding a valid path but not the shortest

3	Find the shortest path with the least turns and generate a motion sequence	+30% if all correct, otherwise • +10% for finding a shortest path with the least turns • +20% for generating correct motion commands, otherwise ○ (16% maximum) +2% each for correctly generating 3 motion commands[1].
4	Write the found path to a text file	+10% if all correct, otherwise • (8% maximum) ○ +2% each for correctly writing 3 characters[2].

[1]A motion command is considered correctly generated if and only if this motion command and all the preceding motion commands are correctly generated. You can only get marks for the motion generation if you have successfully found a shortest path with the least turns.

[2]A character is considered correctly written if and only if this character and all the preceding characters are correctly written.

You should make sure your submitted project is self-contained and up-to-date. Demonstrators will only replace the ~~world file and the motion plan~~ map file for different tests; no debugging/changes to your code (except in the case of hard-coding the map into the program) should be expected from the assessor. The demonstrators will not deal with dependencies either, so you should configure your submission correctly if any external library has been used. If your code did not run correctly (e.g., crashing after started), you could get zero marks for that test.

5.3. Deadline

The submission will be open from 17:00 AEST 5 July 2021 (Monday Week 6) and the deadline is 17:00 AEST 12 July 2021 (Monday Week 7).

If your assignment is submitted after this date, each 1 hour it is late reduces the maximum mark it can achieve by 2%. For example, if an assignment worth 74% were submitted 10 hours late, the late submission would have no effect. If the same assignment were submitted 30 hours late, it would be awarded 40%, the maximum mark it can achieve at that time.

5.4. Progress Check

You will have your progress checked with your demonstrator in a 5 min meeting between 12:00 and 14:00 on Friday Week 5 (or another time on weekdays Week 5 agreed by you and your demonstrator).

During the session, please show your progress by sharing your screen to your demonstrator. If you don't know how to do this, please watch this short video:

https://www.youtube.com/watch?v=DoJqHnpytUU&ab_channel=Microsoft365

To pass the progress check, you must demonstrate that you can generate at least one shortest path from the given map. You need to display the shortest path in a certain form (printing a map with indices as in Fig. 5 or just showing the path coordinates, e.g. for Path - 1, you can show (0,0)(0,1)(0,2)(0,3)(1,3)(1,4)(0,4)(0,5)(0,6)(0,7)(0,8)(1,8)(2,8)(2,7)(2,6)(2,5)(2,4) as the found path). The path must be generated automatically by using a path planning algorithm (hard-coding is not acceptable).

5.5. Plagiarism

You would get zero marks for the assignment if you were found:

- Knowingly providing your work to anyone and it was subsequently submitted (by anyone), or
- Copying or submitting any other persons' work, including code from previous students of this course (except general public open-source libraries/code).

You will be notified and allowed to justify your case before such a penalty is applied.

6. Additional Resources:

- Tutorials on the Flood Fill algorithm (attend the tutorial session)
- https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/
- https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/
- https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-using-priority_queue-stl/
- https://www.geeksforgeeks.org/a-search-algorithm/
- https://www.geeksforgeeks.org/bellman-ford-algorithm-dp-23/