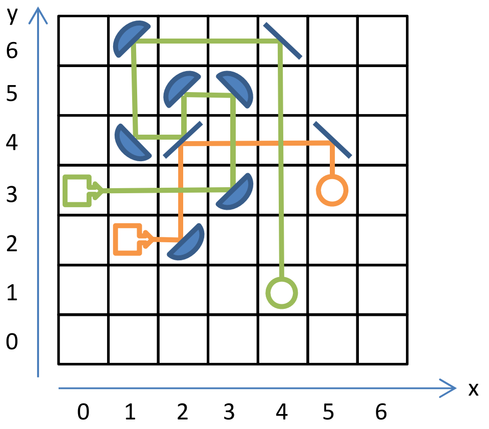
ENGG1111 Computer Programming and Applications - Assignment 1
Second Semester, 2016-2017

The Problem

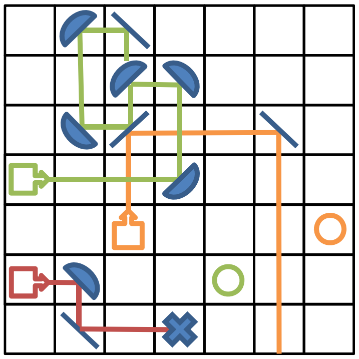
In this assignment, we are going to develop a program to identify the destinations of a few laser beams.

The square laser maze

Consider a sample laser maze shown on the left below:



Maze 1



Maze 2

In maze 1, the orange laser beam starts from coordinates (1,2) with laser going along the East direction. The laser first hits a single-sided mirror at (2,2), and then travels North, hits a double-sided mirror at (2,4), and then goes East, hits another double-sized mirror at (5,4), and direction changes to South. Finally, arriving the destination at (5,3). The green laser beam starts from (0,3) with laser going along the East direction. If you follow the path of the laser beam, you will notice the laser hits a few mirrors and finally arriving the destination at (4,1).

In maze 1, both laser beams arrive at a destination. On the other hand, all beams in maze 2 cannot reach a destination. The orange laser beam goes off the boundary, the green laser beam is blocked at (2,5) and cannot go further, and the red laser beam is blocked at (3,0) as well.

A typical laser maze contains these items:

- At least 1 and at most 5 laser beam sources. Laser beam source will point to one of these directions: North, East, South, West.
- At least 1 and at most 5 destinations. Number of destinations can be different from the number of sources.
- Zero or more single-sided mirrors. There are 4 kinds of single-sided mirror, facing NW, SE, SW, or NE.
- Zero or more double-sided mirrors. There are 2 kinds of double-sided mirror.
- Zero or more laser blocks. This is the cross shown in maze 2 above. A laser block will stop any laser beam at the space that it is located.

Note that in a laser maze, there will be at most 20 items, and no two items share the same coordinates. Smallest possible laser maze is 2x2.

Converting the graphical maze into words

Somehow the laser maze needs to be entered into the program. We will give each item in the laser maze a name. Below is the conversion table:

Symbol	What is it?
single nw	Single-sided mirror facing the direction NW ◤
single se	Single-sided mirror facing the direction SE ◢
single sw	Single-sided mirror facing the direction SW ◣
single ne	Single-sided mirror facing the direction NE ◥
double nw	Double-sided mirror facing NW and SE /
double ne	Double-sided mirror facing NE and SW \
block	Laser block ✖
source *direction*	Laser beam source facing one of N, E, S, W (e.g. ⇨ corresponds to source east)
dest	Destination ○

Using this conversion table, together with the coordinates of each item, maze 1 and maze 2 can be represented non-graphically, like this:

```
single se 1,6
double ne 4,6
single nw 3,3
source east 1,2           source #1
single se 2,5
source east 0,3           source #2
double ne 5,4
dest 4,1              destination #1
single sw 3,5
single ne 1,4
single nw 2,2
dest 5,3              destination #2
double nw 2,4
```

Maze 1

```
single se 1,6
dest 4,1              destination #1
single se 2,5
source east 0,3           source #1
double ne 2,6
single nw 3,3
source north 2,2          source #2
double ne 5,4
single sw 3,5
double ne 1,0
dest 6,2              destination #2
block 3,0
source east 0,1           source #3
single ne 1,4
double nw 2,4
single sw 1,1
```

Maze 2

Source and destination are assumed to be labelled starting from 1. For example, in maze 1 above, there are two sources and two destinations, where the laser beam from source 1 will arrive at destination 2, and the beam from source 2 will hit destination 1. For maze 2, all three sources will not arrive at any destination.

Your task

Develop a program which identifies the destination of each laser beam source. Format of input is shown below:

Line 1: size of the square laser maze (e.g. both maze 1 and maze 2 has size 7)

Line 2: number of items in this maze (e.g. maze 1 has 13 items)

Line 3 onwards: information about the items, one per line, in lowercase. Each line starts with the item name (with necessary direction) followed by the x and y coordinates of that item. e.g. single se 1,6

Output of the program are *n* lines displaying the match for each laser beam source (from 1 to *n*) to a destination. For example, if beam from source 1 arrives at destination 2, **1-2** will be displayed. If there is no match for a laser beam, capital X is used for the destination. For example, if beam from source 2 doesn't arrive at any destination, display **2-X**

A few sample input and output are shown below.

Sample input	Sample output
7 13 single se 1,6 double ne 4,6 single nw 3,3 source east 1,2 single se 2,5 source east 0,3 double ne 5,4 dest 4,1 single sw 3,5 single ne 1,4 single nw 2,2 dest 5,3 double nw 2,4	1-2 2-1
7 16 single se 1,6 dest 4,1 single se 2,5 source east 0,3 double ne 2,6 single nw 3,3 source north 2,2 double ne 5,4 single sw 3,5 double ne 1,0 dest 6,2 block 3,0 source east 0,1 single ne 1,4 double nw 2,4 single sw 1,1	1-X 2-X 3-X
5 7 source west 2,2 source east 1,1 dest 4,3 double ne 1,2 block 3,0 double nw 1,3 double ne 3,1	1-1 2-X
3 6 source west 1,2 source east 2,2 source south 2,1 source north 1,1 dest 2,0 double ne 0,0	1-X 2-X 3-1 4-X

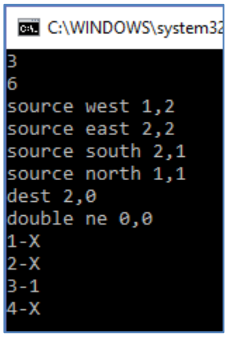
Program testing

Apart from the above sample input and output, please generate your own data for testing. Solution executable will be made available for testing purpose. Please download from Moodle according to your Operating System. Do not attempt to reverse engineer the executable file.

Assumptions you can / cannot made in the program

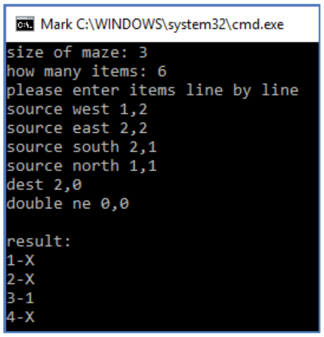
Apart from the assumptions written above, these are additional assumptions you can make when writing the program:

- All user input are valid with correct format.
- Each input item occupies a space within the given boundary.
- No two input items share the same space in the given laser maze.
- Laser beam should stop once it hits a non-reflective object.
- Size of laser maze can be as large as the boundary of an integer.
- Appropriate size of array must be created. Do not create oversized array. For example, there are a maximum of 20 items for a maze, creating an array of size 21 for storing the items will be considered oversized.
- Do not assume any default initialization of variables and array. Write your own initialization codes.
- No need to generate error message.
- Output will be marked by computer, formatting must be observed strictly. Note that "1-X", "1_X", "1-x" are considered different.



```
3
6
source west 1,2
source east 2,2
source south 2,1
source north 1,1
dest 2,0
double ne 0,0
1-X
2-X
3-1
4-X
```

Figure 1 correct input/output format



```
size of maze: 3
how many items: 6
please enter items line by line
source west 1,2
source east 2,2
source south 2,1
source north 1,1
dest 2,0
double ne 0,0

result:
1-X
2-X
3-1
4-X
```

Figure 2 wrong input / output format

The above screens output are considered to be different. Input and output format in Figure 1 follows the given instructions and will be marked correct; Figure 2 will be marked wrong because of the improper format.

Rules and regulations

[I/O formats] Your program will be marked by machines. Therefore, input and output formats should be followed strictly. Each test case will be marked as "correct" or "incorrect", no partial marks will be given to a test case.

[Programming skills] You must demonstrate the use of functions and arrays in your program. Use only 1D array. Other programming skills outside the scope (lecture 1-5) cannot be used. String can be used as a data type only, do not use any kind of string library functions. Include only <iostream> for this assignment. Improper logic, such as using "break" to force a loop to terminate, will cause mark deduction. If you are not sure, please ask.

[Plagiarism] Plagiarism is strictly prohibited! Faculty is very concern about plagiarism and all suspicious cases will be reported to faculty for investigation and penalty. Maximum penalty is zero coursework. You must not copy or let others copy your work. It's your own responsibility to protect your files and prevent others from copying your program directly or indirectly (e.g. obtain your program through another person). The following constitutes an act of plagiarism:

- submitting another student's work as your own, or outsourcing;
- direct copying (full or partial);
- studying someone else's program and then rewriting it as your own;
- group work (i.e. working side by side, and discuss the program line by line);
- providing your assignment as the source for any of the above actions.

Submission (Late submission will not be marked)

Only limited test cases are made public for testing. You are encouraged to create other test cases for testing purpose. Your program will be tested against a number of cases, and each case will be worth a certain number of marks.

Submission of your program (*yourUNo*_a1.cpp**)** by **6:00pm 13 Mar (Mon)** via **Moodle**
Put your name and your 10-digit UNo as comment in the first line of your cpp program. Submit a single **.cpp file** (i.e. *yourUNo*_a1.cpp). Unless Moodle is proved to be out of service within 30 minutes before the deadline, the deadline will not be extended under all circumstances. Submitting a wrong file is not an excuse for late submission. Make sure you have double checked before submission.

Submission of your paper report during lecture on **15 Mar (Wed)**
Prepare the followings (not more than 2 pages of one A4 paper):

- Your name, university number, email.
- Share one technical skill and one debugging skill you have learned in this assignment.
- Screen captures showing: (1) all function declarations, (2) arrays and variables declared in main and global. There is one sample screen capture in Moodle.
- How does your program match the source to the destination? Explain your logic design.
- [Bonus] Consider a new item: a see through mirror. This see through mirror has 2 properties: (1) same property as a single-sided mirror, (2) able to transmit laser beam through the mirror as if the whole mirror does not exist. How to modify the program logic to cater for this new item? Share your idea. No need to code.

Marking scheme [total mark: 100]

- 70 marks for program testing
- 10 marks for functions (at least 2 meaningful functions)
- 10 marks for arrays and variables
  - Define variables with meaningful names
  - Do not create oversized array
- 10 marks for the report
- 10 marks for the bonus
- Deduction (max 20 marks) for incorrect input/output format, wrong filename.
- Deduction (max 50 marks) for the use of break or skills outside our scope.