C Programming II
2021 Spring
Midterm

Instructor: Po-Wen Chi

Date: 2021.11.13 AM 9:00-12:00

Policies:

- Online test.
- Do not forget to include your Makefile. TA will only use the command make to build your program. If make fails, you will get zero points and no room for bargaining. So if you do not know how to solve a problem, please, do not include it in your Makefile.
- I do not care your source code file names, but the executive binary names should be **mid01, mid02, mid03, mid04**.
- You can ask TA if you do not understand the problems.

1 Integrating Two Integers (25 pts)

Consider a 32-bits nonnegative integer n whose digits always appear in ascending order from left to right, such as 24667. In other words, if $d_0 d_1 \cdots d_{m-1}$ represents the sequence of m digits of n, then $d_i \leq d_j$ for $i < j$. Now given two integers n_0, n_1 that satisfy the above requirement, please write a program to combine n_0, n_1 to a new integer that also satisfies the rule that the output number's digits appear in **descending order** from left to right.

```
$ ./mid01
Please enter the 1st integer: 137
Please enter the 2nd integer: 246
Result: 764321
```

For the sake of fairness, **using Array is not allowed!!**

You should give an error message and terminate your program when receiving an invalid input.

2 Pattern Checker (25 pts)

The user will enter a series of integers and you need to implement a program to see if the given integers contain a series of continuous integers which matches the following pattern.

Pattern:

- In the beginning, there must be zero or one integer which is in {13,27,68}.
- Next, there must be one or two integers which are in {-5,19,103,27}.
- Then, there must be an integer -33.
- Repeat the above pattern again.

For example, the following input series contain a sub-series of integers that satisfies the pattern and I use the red color to show the matching part for you.

77 -95 103 -5 -33 27 -5 19 -33 109 33 999 0

The integer 0 is used as the end sign. Your program should read all user input integers until zero. The output should be the position and the value of the first integer of the first matching series. If there is no matching series, just print **None**.

```
$ ./mid02
Please enter the integer: 77
Please enter the integer: -95
Please enter the integer: 103
Please enter the integer: -5
Please enter the integer: -33
Please enter the integer: 27
Please enter the integer: -5
Please enter the integer: 19
Please enter the integer: -33
Please enter the integer: 109
Please enter the integer: 33
Please enter the integer: 999
Please enter the integer: 0
The first matching series is at position 3, integer 103.
```

All integers are guaranteed as 32-bits integers.

3 Snell's Law (25 pts)

Snell's law (also known as Snell–Descartes law and the law of refraction) is a formula used to describe the relationship between the angles of incidence and refraction, when referring to light or other waves passing through a boundary between two different isotropic media, such as water, glass, or air. Figure 1 is an example about the refraction of light at the interface between two media of different refractive indices

In optics, the law is used in ray tracing to compute the angles of incidence or refraction, and in experimental optics to find the refractive index of a material. Snell's law states that the ratio of the sines of the angles of incidence and refraction is equivalent to the ratio of

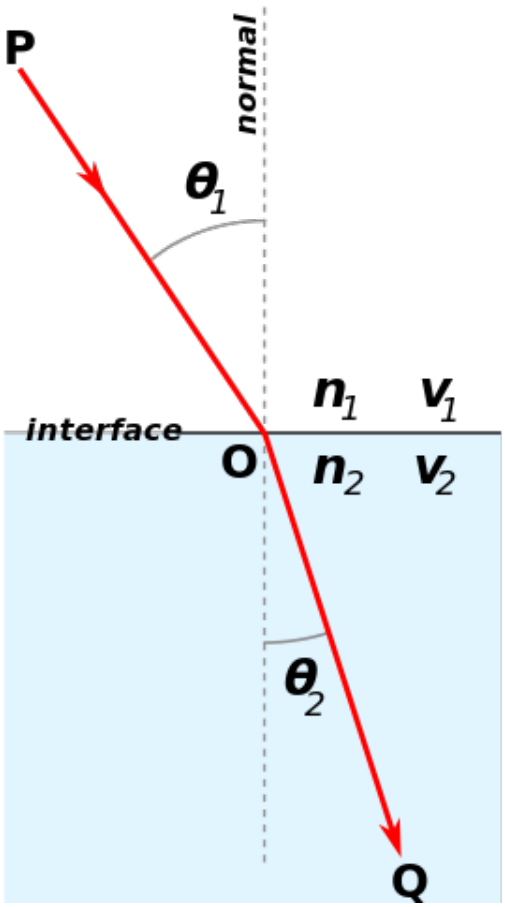

Figure 1: Snell's Law.

phase velocities in the two media, or equivalent to the reciprocal of the ratio of the indices of refraction. We can use the following equation to calculate the refraction angle:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2,$$

where n_1, n_2 are the refractive index (which is unit-less) of the respective medium. Now, given a container with multi-layer mediums, please implement a program to calculate the light position shift from the incidence. Suppose the refractive index of the first medium is 1. Please see the figure 2 for reference.

```
$ ./mid03
Incidence angle: 45
How many layers: 1
Layer 1's refractive index: 1
Layer 1's height: 1
The shift: 1
```

Note that you should use **double** and the precision is not a concern. When receiving the invalid input, you should print a warning message and terminate the program.

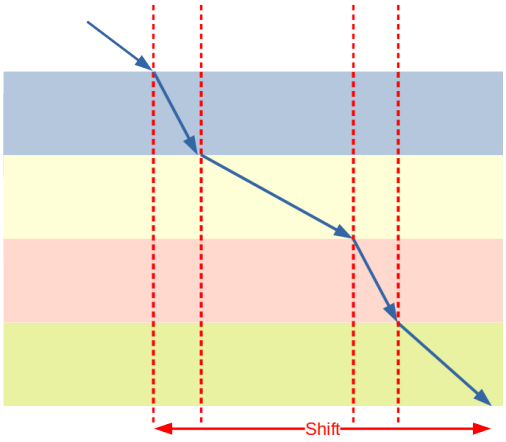

Figure 2: An example.

4 Mission Impossible (25 pts)

Imagine that you are a spy who wants to steal a secret from the enemy. Undoubtedly, you cannot be caught. Please implement this spy game.

The game map is 20×80. First, you need to setup the game environment as follows and I will introduce these attributes later. If there is any input value invalid, you should make the user to input that value again.

```
$ ./mid04
Your movement (3-6): 3
Enemy 1 movement (3-6): 12
Invalid input!!
Enemy 1 movement (3-6): 3
Enemy 1 vision (2-10): 2
Enemy 1 location (2-80): 10
Enemy 2 movement (3-6): 3
Enemy 2 vision (2-10): 2
Enemy 2 location (2-20): 10
```

Figure 3 is the game map. Initially, the secret is in the top left corner and the player is in the bottom right. We use **'S'** and **'P'** to represent the secret and the player respectively. **'1'** and **'2'** are enemy 1 and enemy 2, Enemy 1 starts from the top line while enemy 2 starts from the left line. Their locations are given by the user. Here we use **'*'** to represent the enemy's vision. The vision is radiated, as shown in Fig. 4.

The user should control the player to get the secret without being noticed by the enemies, which means that the player cannot be contacted by the enemies including their visions. This is a turn-based game and the order is **Player** → **Enemy 1** → **Enemy 2**. After each phase, you need to show the map status like Fig. 3. For the player phase, the control menu should be:

```
Move: (1) Up (2) Down? 1
Range (0-3)? 1
Move: (1) Left (2) Right? 1
Range (0-3)? 2
```

Figure 3: Game Map.

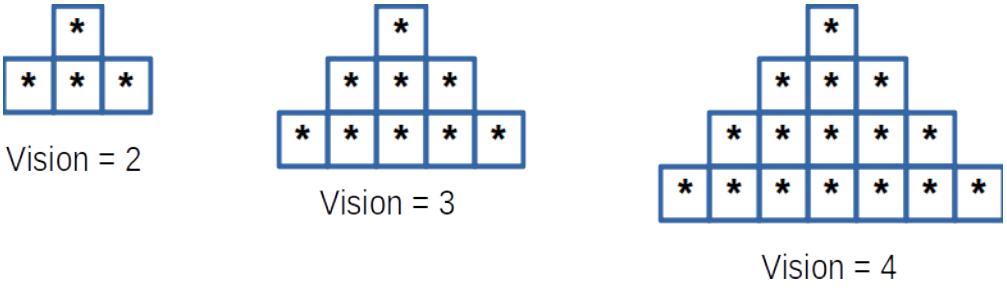

Figure 4: Enemy vision.

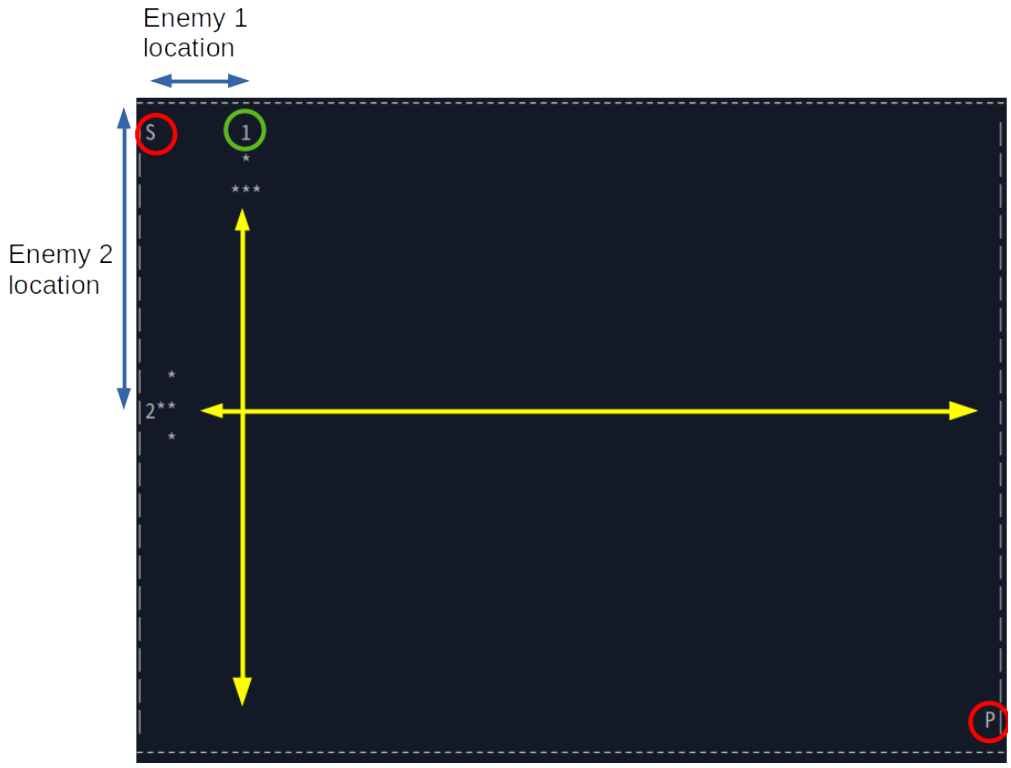

Figure 5: Enemy movement.

Undoubtedly, the player's movement sum cannot be over the setup movement value and the player cannot cross the boundary. If the input is invalid, make the user to input again. BTW, the player can stay on the same location. If the player moves to the secret place, your program should print "Mission Complete!" on the screen and being terminated.

Next is the enemy phase. Enemy 1 will only move on the horizontal line while enemy 2 will only move on the horizontal line, as shown in Fig. 5. Each turn, they move forward according to their movement values. Once their vision touch the boundary, the enemies will turn backward and keeps their patrols next turn like Fig. 6. If the enemy catches the player, your program should print "Mission Fail!" on the screen and being terminated.

The score distribution is as follows:

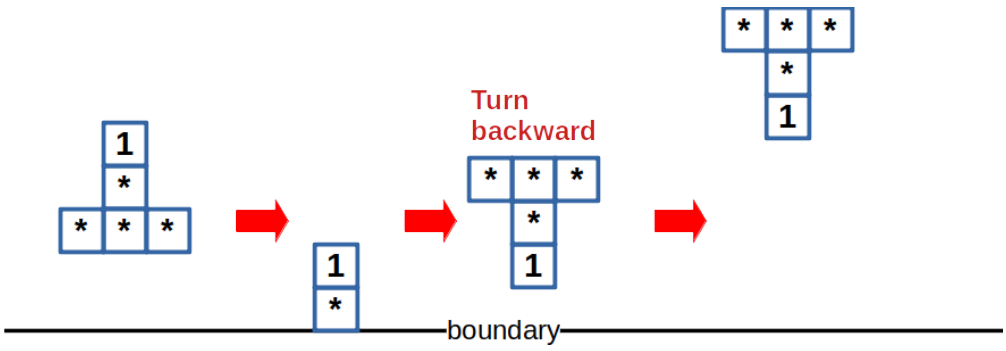

Figure 6: Enemy phase.

- Setup: 5 pts
- Map Printing: 5 pts
- Player Phase: 5 pts
- Enemy Phase: 5 pts
- Game Complete: 5 pts.

5 Bonus: Your Comments (5 pts)

Again, any comments are welcomed. However, you will get nothing if you leave this question blank.