EECS2040 Data Structure Hw #2 (Chapter 3 Stack/Queue)

due date 4/17/2022 by 109070025 林泓錩

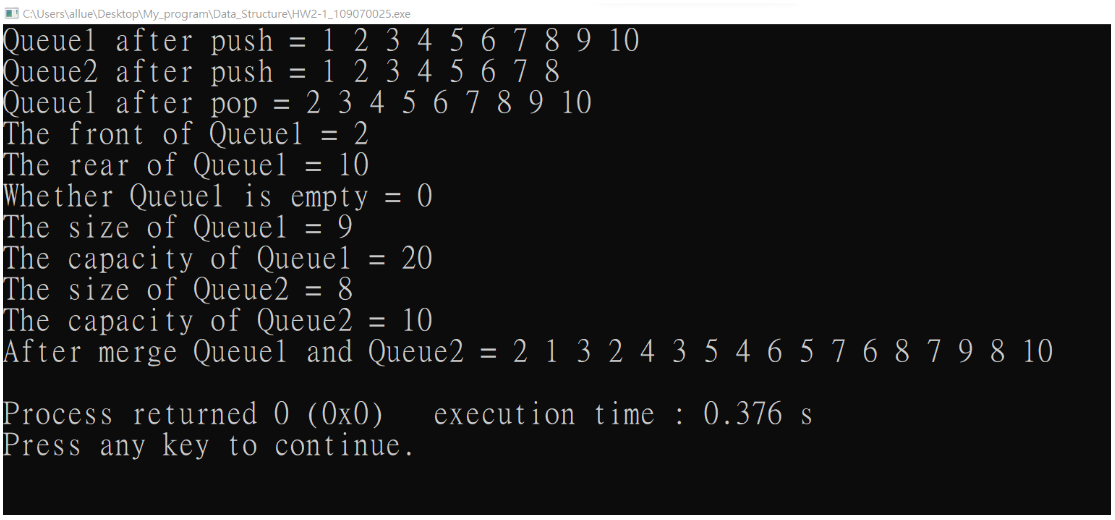
Part 2 Coding (5% of final Grade)

1. (30%) Based on the circular queue and template queue ADT in **ADT 3.2** shown below (or pptx pp.82), write a C++ program to implement the queue ADT using dynamic (circular) array (10%**).** Then add three more functions to
   (a) (5%) Return the size of a queue (int Size()).
   (b) (5%) Return the capacity of a queue (int Capacity()).
   (c) (10%) Merge two queues into a one by alternately taking elements from each queue. The relative order of queue elements is unchanged. What is the complexity of your function?

   You should **demonstrate all the functions** using at least one example.

```
 template < class T >
 class Queue
 {
 public:
        Queue (int queueCapacity = 0);
        bool IsEmpty( ) const;
        void Push(const T& item);       // add an item into the queue
        void Pop( );        // delete an item
        T& Front() const;        // return top element of stack
        T& Rear() const;        // return top element of stack
} ;
```

Ans:



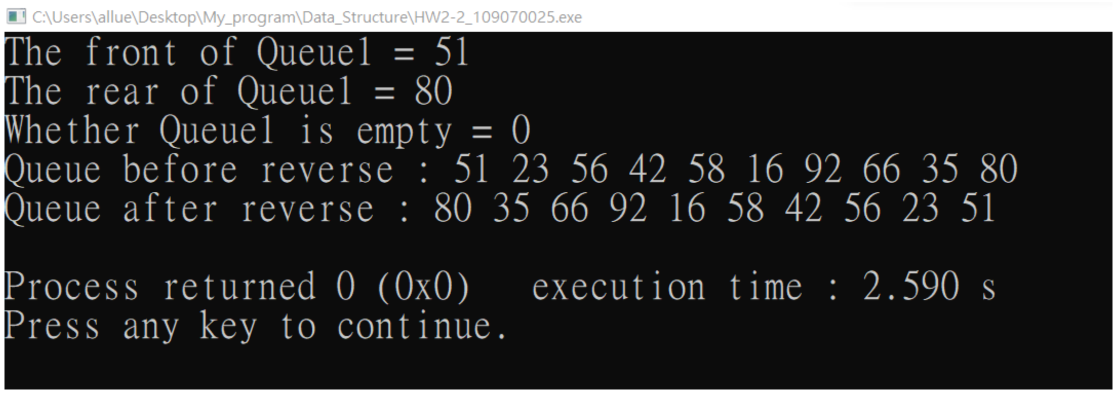
C:\Users\allue\Desktop\My_program\Data_Structure\HW2-1_109070025.exe

```
Queue1 after push = 1 2 3 4 5 6 7 8 9 10
Queue2 after push = 1 2 3 4 5 6 7 8
Queue1 after pop = 2 3 4 5 6 7 8 9 10
The front of Queue1 = 2
The rear of Queue1 = 10
Whether Queue1 is empty = 0
The size of Queue1 = 9
The capacity of Queue1 = 20
The size of Queue2 = 8
The capacity of Queue2 = 10
After merge Queue1 and Queue2 = 2 1 3 2 4 3 5 4 6 5 7 6 8 7 9 8 10

Process returned 0 (0x0)   execution time : 0.376 s
Press any key to continue.
```

(a) is in line 7 & 9、(b) is in line 8 & 10、(c) is in line 11

2. (20%) Design a C++ function template, reverseQueue, that takes as a parameter a queue object and uses a stack object to reverse the elements of the queue. The operations on queue and stack should strictly follow the ADT 3.2 Queue ADT and ADT 3.1 Stack ADT. You should **demonstrate the functions** using at least one example, e.g., queue1=(1,3,5,7), queue2=(2,4,6,8), mergedqueue=(1,2,3,4,5,6,7,8)

   Ans:

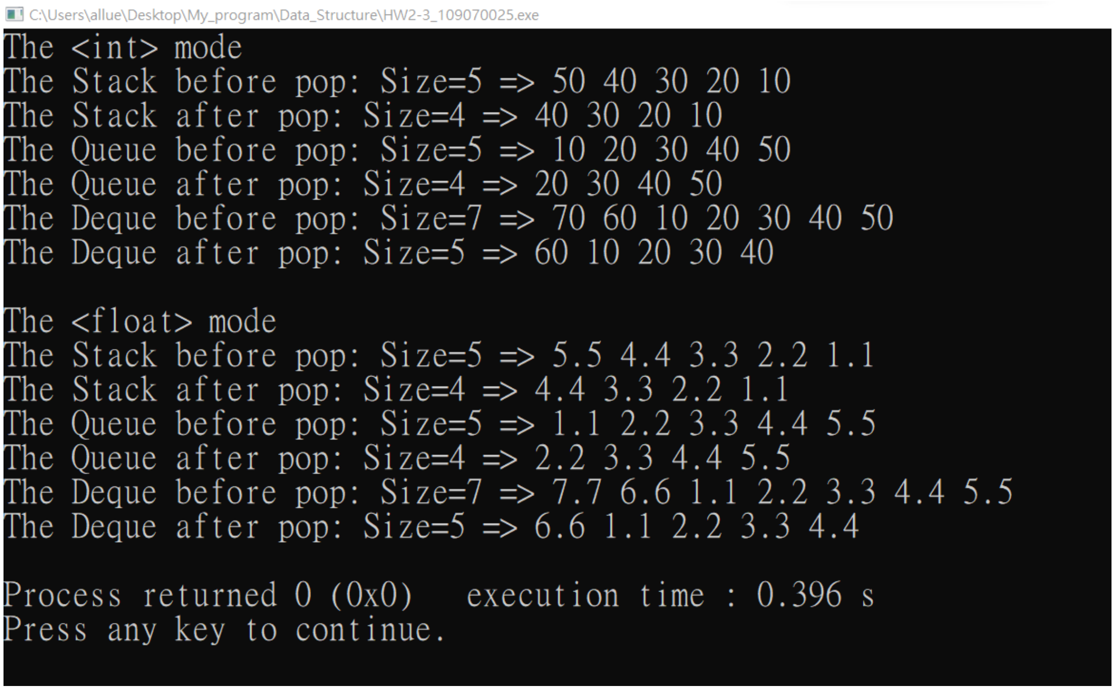


```
C:\Users\allue\Desktop\My_program\Data_Structure\HW2-2_109070025.exe
The front of Queue1 = 51
The rear of Queue1 = 80
Whether Queue1 is empty = 0
Queue before reverse : 51 23 56 42 58 16 92 66 35 80
Queue after reverse : 80 35 66 92 16 58 42 56 23 51

Process returned 0 (0x0)   execution time : 2.590 s
Press any key to continue.
```

3. (25%) Referring to **Program 3.13** in textbook (pptx pp.98),
   (a) (5%) Implement Stack as a publicly derived class of Bag using template. **Demonstrate** your C++ code using at least two element types (e.g., int, float,…). **Show results** of a series of Pushes and Pops and Size functions.
   (b) (5%) Implement Queue as a publicly derived class of Bag using template. **Demonstrate** your C++ code using at least two element types (e.g., int, float,…). **Show results** of a series of Pushes and Pops and Size functions.
   (c) (15%) A template double-ended queue (deque) is a linear list in which additions and deletions may be made at either end. Implement the class Deque as a publicly derived templated class of Queue. The class Deque must have public functions (either via inheritance from Queue or by direct implementation in Deque) to add and delete elements from either end of the deque and also to return an element from either end. The complexity of each function (excluding array doubling) should be Θ(1).
   Demonstrate your C++ code using at least two element types (e.g., int, float,…). **Show results** of a series of two types of Pushes and Pops and Size functions to illustrate your code is working.

   Ans:

   I do the int and float types. And, at each type, the line 1&2 shows the Stack function, 3&4 shows the Queue function, and 7&8 shows the Deque function.



```
C:\Users\allue\Desktop\My_program\Data_Structure\HW2-3_109070025.exe
The <int> mode
The Stack before pop: Size=5 => 50 40 30 20 10
The Stack after pop: Size=4 => 40 30 20 10
The Queue before pop: Size=5 => 10 20 30 40 50
The Queue after pop: Size=4 => 20 30 40 50
The Deque before pop: Size=7 => 70 60 10 20 30 40 50
The Deque after pop: Size=5 => 60 10 20 30 40

The <float> mode
The Stack before pop: Size=5 => 5.5 4.4 3.3 2.2 1.1
The Stack after pop: Size=4 => 4.4 3.3 2.2 1.1
The Queue before pop: Size=5 => 1.1 2.2 3.3 4.4 5.5
The Queue after pop: Size=4 => 2.2 3.3 4.4 5.5
The Deque before pop: Size=7 => 7.7 6.6 1.1 2.2 3.3 4.4 5.5
The Deque after pop: Size=5 => 6.6 1.1 2.2 3.3 4.4

Process returned 0 (0x0)   execution time : 0.396 s
Press any key to continue.
```

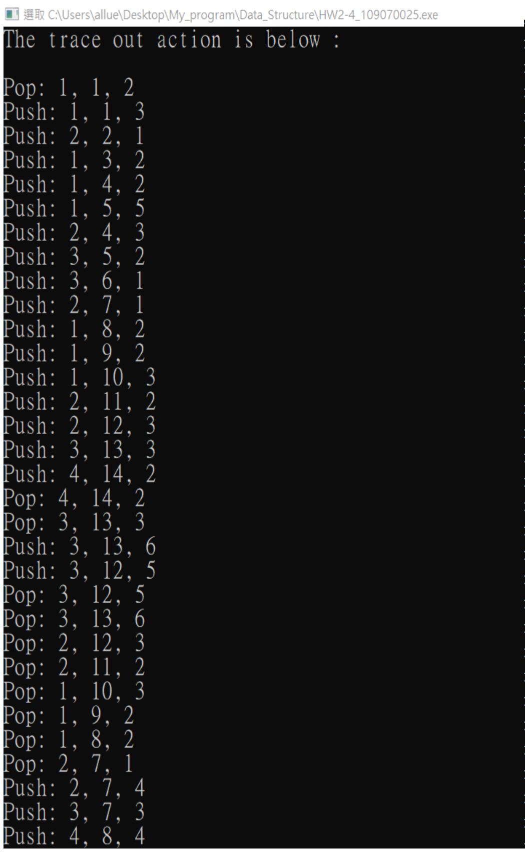
4. (25%) Write a C++ program to implement the maze in textbook using the example codes of **Program 3.15** (pptx pp.106 Algorithm()) and **3.16 (pptx void Path(const int m, const int p)**. You should use a text editor to edit a file containing the maze matrix and then **read in the file to establish the maze matrix** in your program. The default entrance and exit are located in the upper left corner and lower right corner, respectively as shown in textbook.
   (a) (10%) Demonstrate your maze program using the maze shown in **Figure 3.11**.
   (b) (5%) Find a path manually through the maze shown in **Figure 3.11**.
   (c) (10%) Trace out the action of function path (**Program 3.16**) on the maze shown in Figure 3.11. Compare this to your own attempt in (b).

$$
\begin{array}{c|ccccccccccccccc|c}
\text{入口} & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & \\
 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & \\
 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \\
 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & \\
 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & \\
 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & \\
 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & \\
 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & \\
 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & \\
 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & \text{出口}
\end{array}
$$

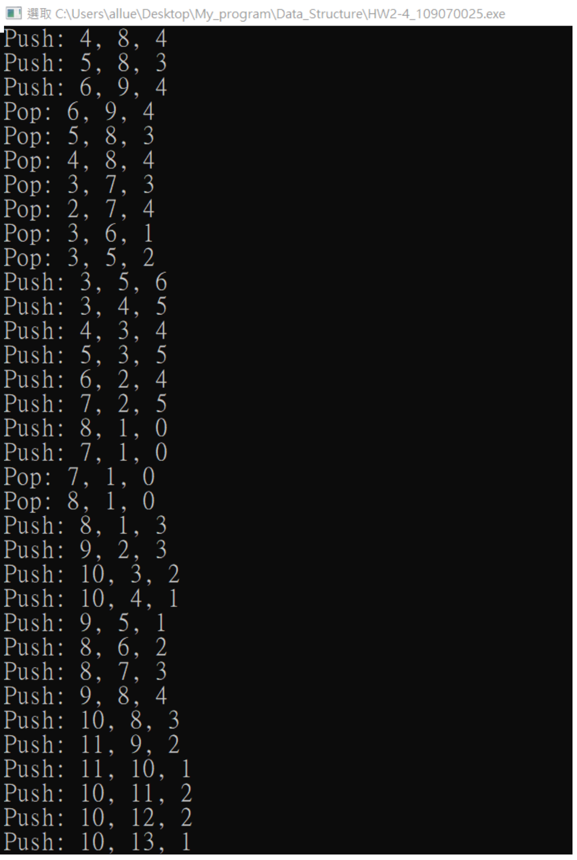
Figure 3.11：一個迷宮的例子（你能找出一條路徑嗎？）

Ans:

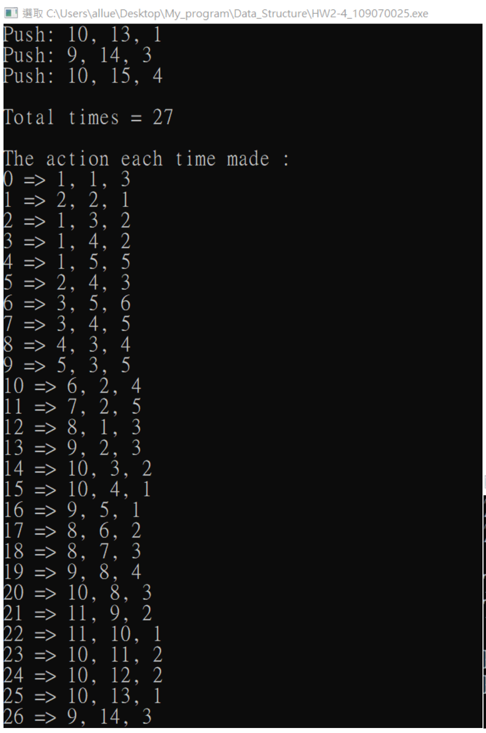
The trace out path is below:


選取 C:\Users\allue\Desktop\My_program\Data_Structure\HW2-4_109070025.exe

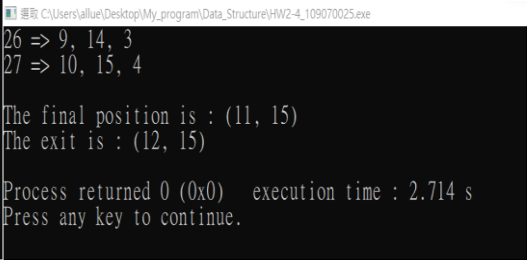
```
The trace out action is below :

Pop: 1, 1, 2
Push: 1, 1, 3
Push: 2, 2, 1
Push: 1, 3, 2
Push: 1, 4, 2
Push: 1, 5, 5
Push: 2, 4, 3
Push: 3, 5, 2
Push: 3, 6, 1
Push: 2, 7, 1
Push: 1, 8, 2
Push: 1, 9, 2
Push: 1, 10, 3
Push: 2, 11, 2
Push: 2, 12, 3
Push: 3, 13, 3
Push: 4, 14, 2
Pop: 4, 14, 2
Pop: 3, 13, 3
Push: 3, 13, 6
Push: 3, 12, 5
Pop: 3, 12, 5
Pop: 3, 13, 6
Pop: 2, 12, 3
Pop: 2, 11, 2
Pop: 1, 10, 3
Pop: 1, 9, 2
Pop: 1, 8, 2
Pop: 2, 7, 1
Push: 2, 7, 4
Push: 3, 7, 3
Push: 4, 8, 4
```


選取 C:\Users\allue\Desktop\My_program\Data_Structure\HW2-4_109070025.exe

```
Push: 4, 8, 4
Push: 5, 8, 3
Push: 6, 9, 4
Pop: 6, 9, 4
Pop: 5, 8, 3
Pop: 4, 8, 4
Pop: 3, 7, 3
Pop: 2, 7, 4
Pop: 3, 6, 1
Pop: 3, 5, 2
Push: 3, 5, 6
Push: 3, 4, 5
Push: 4, 3, 4
Push: 5, 3, 5
Push: 6, 2, 4
Push: 7, 2, 5
Push: 8, 1, 0
Push: 7, 1, 0
Pop: 7, 1, 0
Pop: 8, 1, 0
Push: 8, 1, 3
Push: 9, 2, 3
Push: 10, 3, 2
Push: 10, 4, 1
Push: 9, 5, 1
Push: 8, 6, 2
Push: 8, 7, 3
Push: 9, 8, 4
Push: 10, 8, 3
Push: 11, 9, 2
Push: 11, 10, 1
Push: 10, 11, 2
Push: 10, 12, 2
Push: 10, 13, 1
```

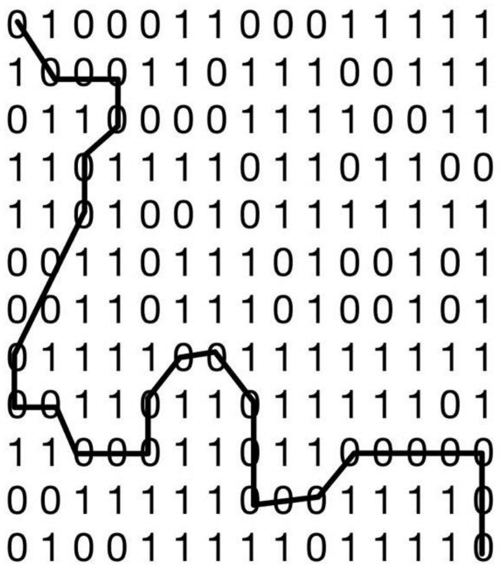

選取 C:\Users\allue\Desktop\My_program\Data_Structure\HW2-4_109070025.exe

```
Push: 10, 13, 1
Push: 9, 14, 3
Push: 10, 15, 4

Total times = 27

The action each time made :
0 => 1, 1, 3
1 => 2, 2, 1
2 => 1, 3, 2
3 => 1, 4, 2
4 => 1, 5, 5
5 => 2, 4, 3
6 => 3, 5, 6
7 => 3, 4, 5
8 => 4, 3, 4
9 => 5, 3, 5
10 => 6, 2, 4
11 => 7, 2, 5
12 => 8, 1, 3
13 => 9, 2, 3
14 => 10, 3, 2
15 => 10, 4, 1
16 => 9, 5, 1
17 => 8, 6, 2
18 => 8, 7, 3
19 => 9, 8, 4
20 => 10, 8, 3
21 => 11, 9, 2
22 => 11, 10, 1
23 => 10, 11, 2
24 => 10, 12, 2
25 => 10, 13, 1
26 => 9, 14, 3
```

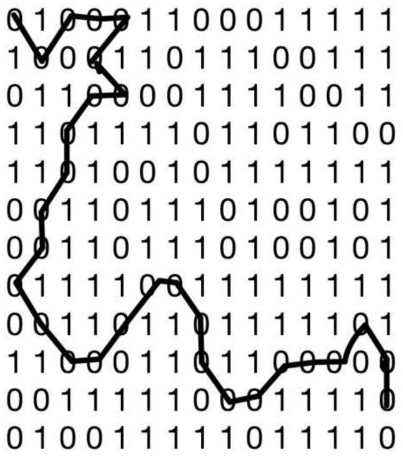

選取 C:\Users\allue\Desktop\My_program\Data_Structure\HW2-4_109070025.exe

```
26 => 9, 14, 3
27 => 10, 15, 4

The final position is : (11, 15)
The exit is : (12, 15)

Process returned 0 (0x0)   execution time : 2.714 s
Press any key to continue.
```

Manual:



Program Trace out:


0 1 0 0 0 1 1 0 0 0 1 1 1 1 1
1 0 0 0 1 1 0 1 1 1 0 0 1 1 1
0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 0 1 1 1 1 0 1 1 0 1 1 0 0
1 1 0 1 0 0 1 0 1 1 1 1 1 1 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 0 1 1 0 1 1 1 0 1 0 0 1 0 1
0 1 1 1 1 0 0 1 1 1 1 1 1 1 1
0 0 1 1 0 1 1 0 1 1 1 1 1 0 1
1 1 0 0 0 1 1 0 1 1 0 0 0 0 0
0 0 1 1 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 1 1 1 1 1 0 1 1 1 1 0