**EECS2040 Data Structure Hw #2 (Chapter 3 Stack/Queue)**
**due date 4/23/2021**
**by 107061123, 孫元駿**

## Part 2 Coding (50%)

You should submit:

(a) All your source codes (C++ file).

(b) Show the execution trace of your program.

1. (30%) Based on the circular queue and template queue ADT in **ADT 3.2** in textbook (or pptx pp.79), write a C++ program to implement the queue ADT**.** Then add two more functions to
   (a) Return the size and capacity of a queue.
   (b) Merge two queues into a one by alternately taking elements from each queue. Te 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.

Ans:

In ADT 3.2 mention 1 Constructor and 5 functions, and I write three more functions.

Functions in ADT 3.2: *IsEmpty, Front, Rear, Push, Pop*

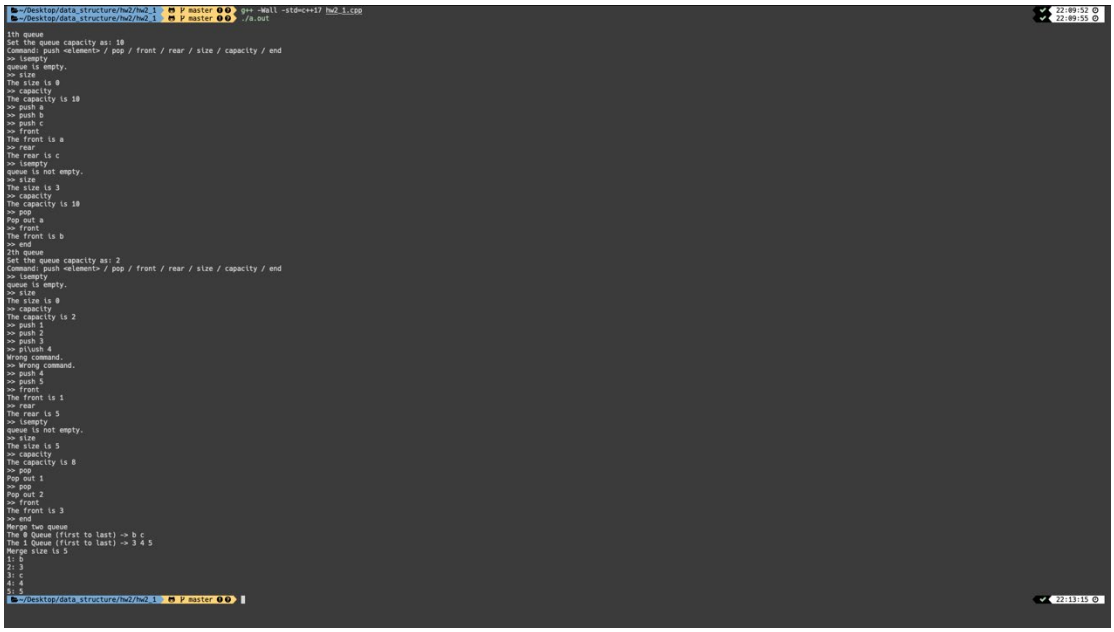
Functions addition: *Capacity, Size, MergeQ(for merging two queue)*

Command Table:

*isempty -> IsEmpty / front -> Front / rear -> Rear / push <element> -> Push*

*pop -> Pop / capacity -> Capacity / size -> Size*

*do it after finish inserting two queue -> MergeQ*



About my program:
You are going to create two queues and the program will merge them together. First, you need to insert an integer as the queue capacity. Then the program will create first queue container. After that you can insert command as the *command table* above. You can finish pushing and popping the element by command "end". Then, you can create the second queue and repeat the step above.
In the end, the program will print out the left element respectively and also merge the two queue together and print out the queue after merging.

2. (35%) Referring to **Program 3.13** in textbook (pptx pp.94),
   (a) 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) 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) 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 $\Theta(1)$.
   **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:

(a)

Three functions to show, *Pushes, Pops, Size*

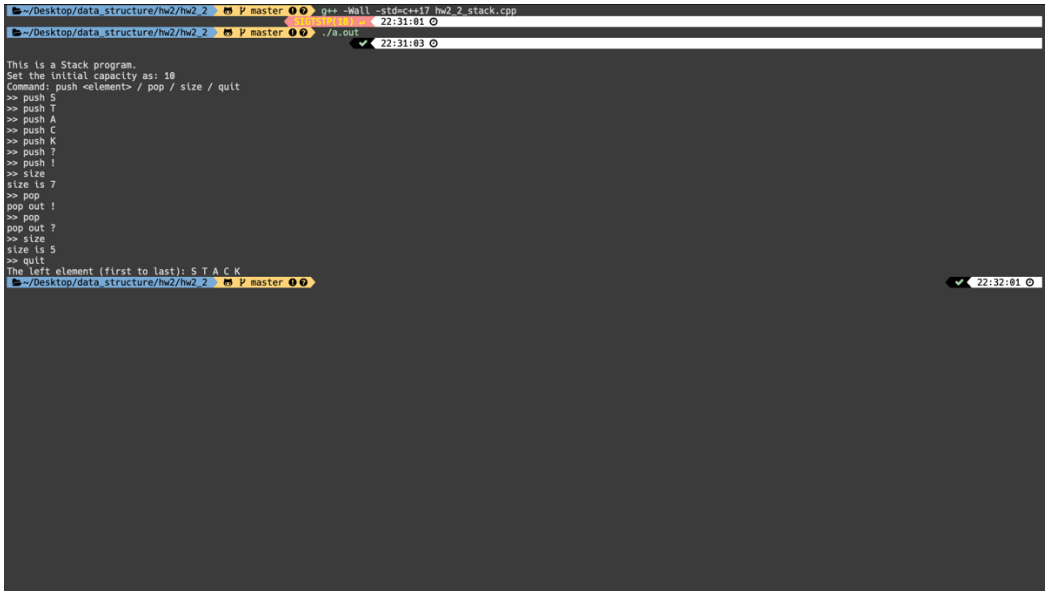
Command Table:

*push <element> -> Pushes*

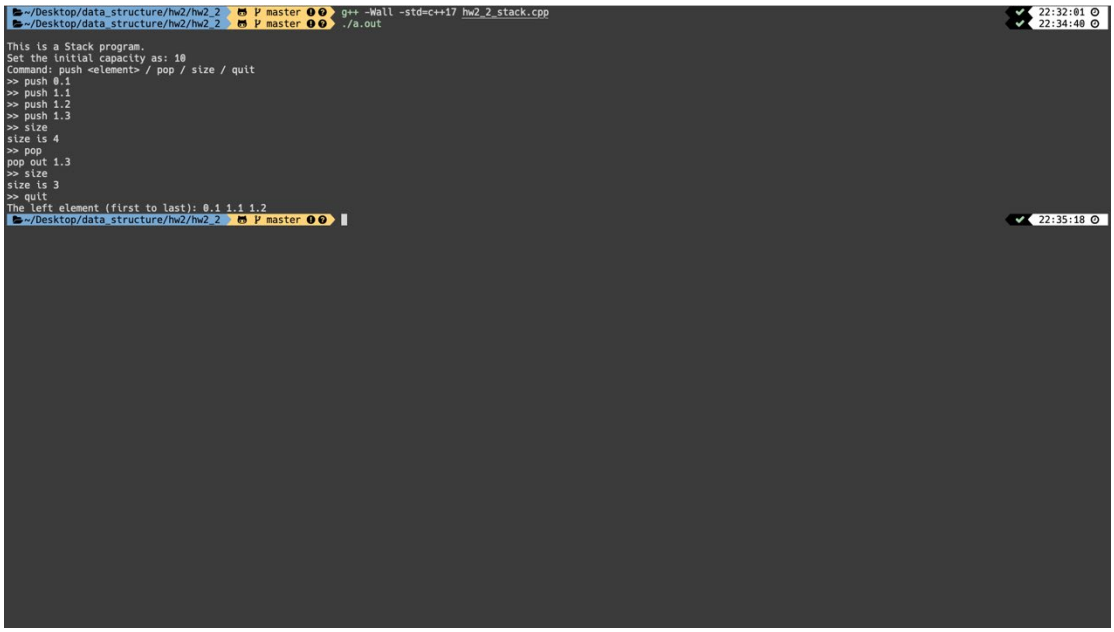
*pop -> Pops*

*size -> Size*

Type 1: char




Type 2: float



(b)

Three functions to show, *Pushes, Pops, Size*

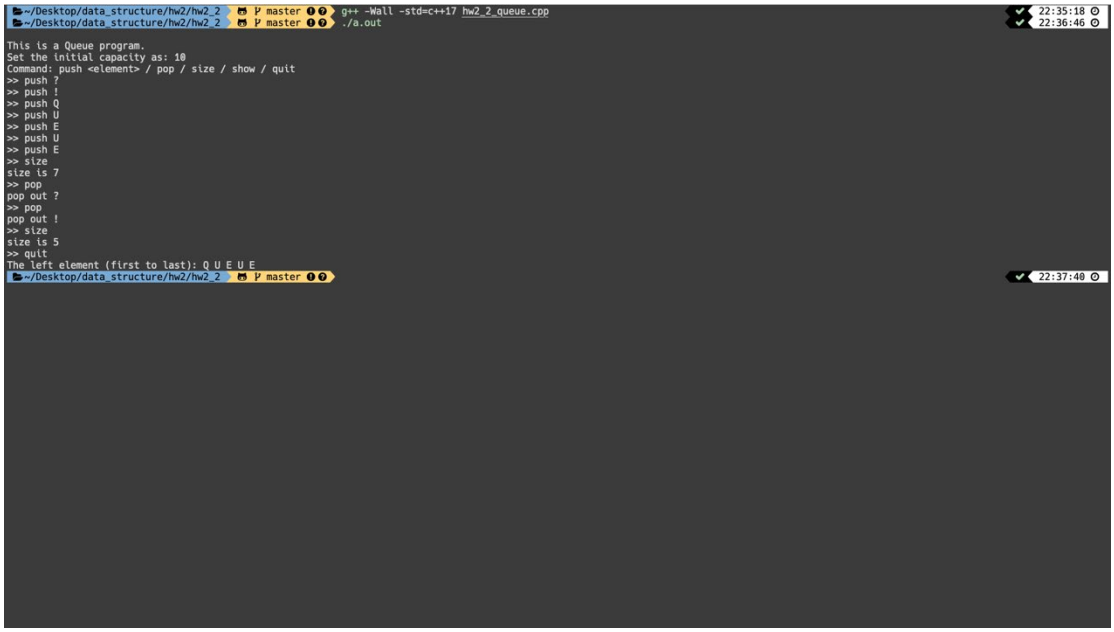
Command Table:

*push <element> -> Pushes*

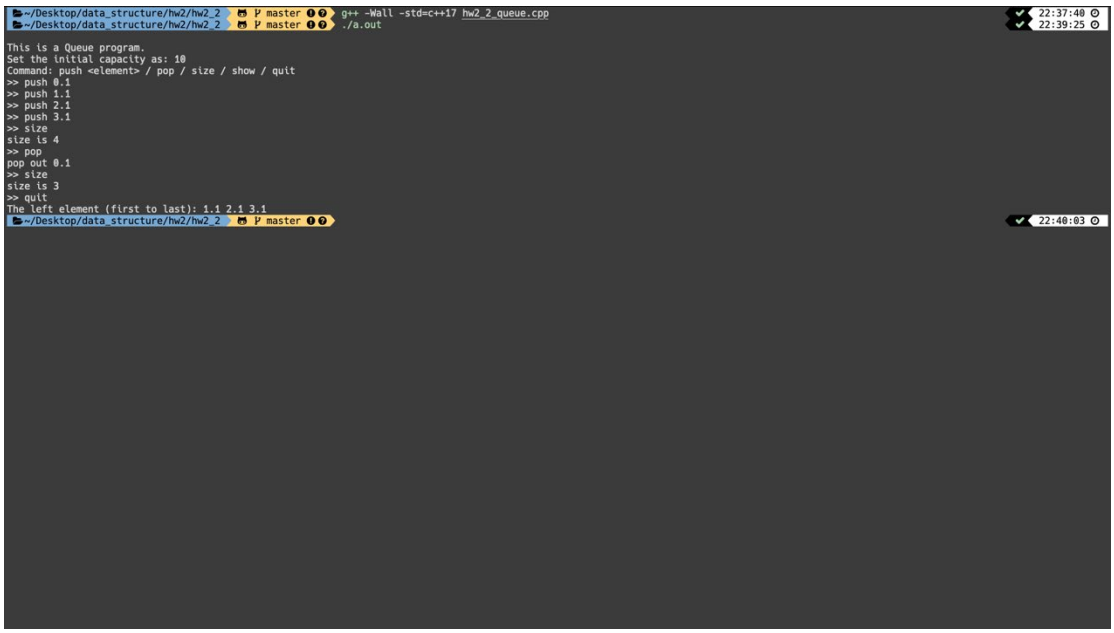
*pop -> Pops*

*size -> Size*

Type 1: char




Type 2: float



(c)

Five functions to show, *Pushes(Stack), Pushes(Queue), Pop(Stack), Pop(Queue), Size*

Command Table:

*push_top <element> -> Pushes(Stack)*

*push_bottom <element> -> Pushes(Queue)*

*pop_top -> Pops(Stack)*

*pop_bottom -> Pops(Queue)*

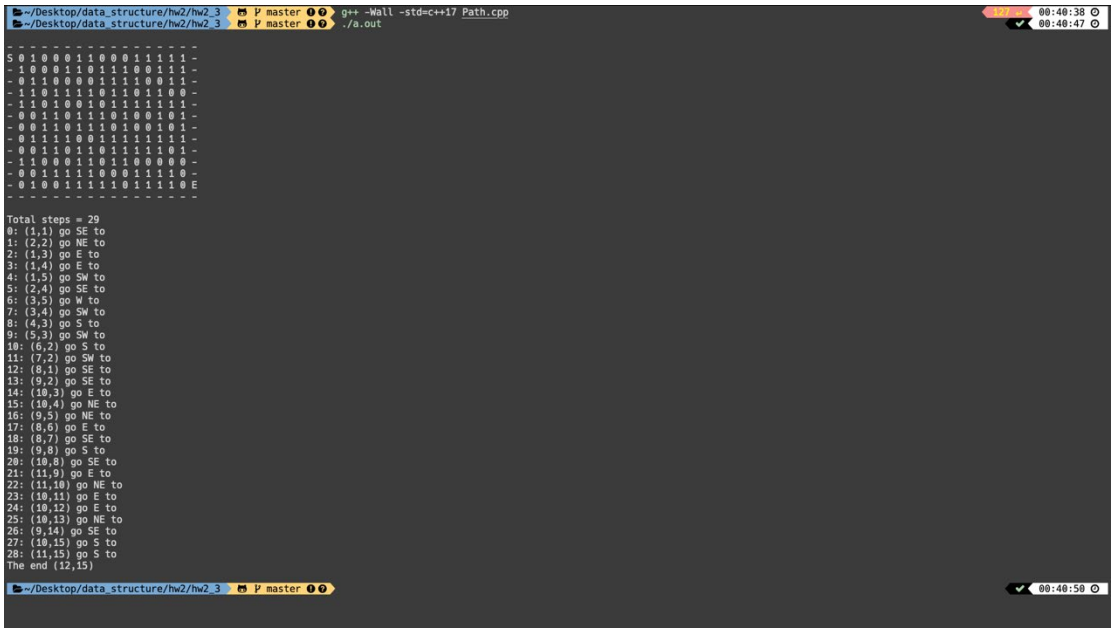
*size -> Size*

Type 1: char

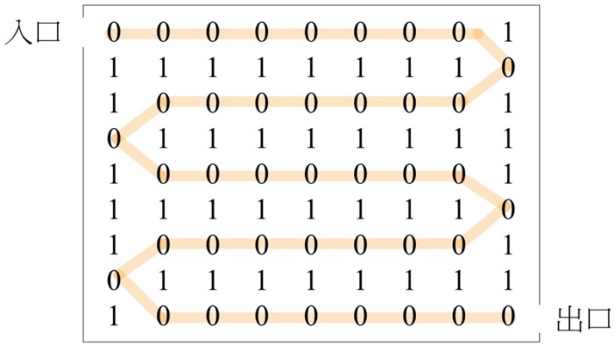

Type 2: float

About my program (a)(b)(c):
You are going to create a stack/queue/deque.
First, you need to insert an integer as the stack/queue/deque capacity. Then the program will create a stack/queue/deque container. After that you can insert command as the *command table* above. You can finish pushing and popping the element by command “quit”.
Then, the program will print out the stack/queue/deque.

3. (35%) 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) Demonstrate your maze program using the maze shown in **Figure 3.11** in textbook.




   (b) Find a path through the maze shown **Figure 3.14** in textbook



(c) Trace out the action of function path (**Program 3.16**) on the maze shown. Compare this to your own attempt in (b).


About my program (a)(c):

You are going to get the path by (x,y) and direction every step to solve the maze.

First, you need to have a maze txt file and this cpp file under the same folder.

Then run the program, you will get the step and the path.

---

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 |

出口

---

**Figure 3.11：** 一個迷宮的例子（你能找出一條路徑嗎？）

入口

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

出口

**Figure 3.14：**唯一路徑的迷宮圖