EE2410 Data Structure Coding HW #2 (Chapter 3 of textbook)
due date 4/14/2024(Sun.) 23:59

You should submit:
(a) All your source codes (C++ file).
(b) Show the execution trace of your program, i.e., write a client main() to demonstrate all functions you designed using example data.

Submit your homework before the deadline (midnight of 4/14). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have SERIOUS consequences.

Stacks & Queues: **due date: 23:59, 4/14/2024 (Sun.)**

1. (30%)

Referring to **Program 3.13** (definition of Bag and Stack):

```
Class Bag
{
public:
   Bag (int bagCapacity = 10);
   virtual ~Bag( );
   virtual int Size( ) const;
   virtual bool IsEmpty( ) const;
   virtual int Element( ) const;
   virtual void Push(const int);
   virtual void Pop( );
protected:
   int *array;
   int top;
};
```

```
class Stack : public Bag
{
public:
   Stack (int stackCapacity = 10);
   ~Stack( );
   int Top( ) const;
   void Pop( );
};
```

(a) (15%) 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) (15%) 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.

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.

Sol:

(a) Execution trace:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"1.1.a_stack"
Enter five integer to push into the stack: 43 78 99 32 100
Stack: [bottom]  43   78   99   32   100   [top]
The size of the stack is: 5
How many element do you want to pop? 5
The Stack after pop: [bottom]  [top]
The size of the stack is: 0
=======================================================
Enter five float to push into the stack: 33.2 2.1 99 3.4 2.045
Stack: [bottom]  33.2   2.1   99   3.4   2.045   [top]
The size of the stack is: 5
How many element do you want to pop? 2
The Stack after pop: [bottom]  33.2   2.1   99   [top]
The size of the stack is: 3
sunpierce@pierces-MacBook-Air output %
```

(b) Execution trace:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"1.1.b_queue"
Enter five integer to push into the queue: 43 78 99 32 100
Queue: [Front]  43   78   99   32   100   [Rear]
The size of the queue is: 5
How many element do you want to pop? 5
The Queue after pop: [Front]  [Rear]
The size of the queue is: 0
=======================================================
Enter five float to push into the queue: 33.2 2.1 99 3.4 2.045
Queue: [Front]  33.2   2.1   99   3.4   2.045   [Rear]
The size of the queue is: 5
How many element do you want to pop? 2
The Queue after pop: [Front]  99   3.4   2.045   [Rear]
The size of the queue is: 3
sunpierce@pierces-MacBook-Air output %
```

2. (35%)

Based on the circular queue and template queue ADT in **ADT 3.2** shown below, write a C++ program to implement the queue ADT using dynamic (circular) array**.** Then add following functions to

(a) (5%) Return the size of a queue (int Size()).

(b) (5%) Return the capacity of a queue (int Capacity()).

(c) (5%) Overload the relational operator == for the class Queue that returns true if two queues of the same type are the same, false otherwise. (Two queues of the same type are the same if they have the same number of elements and their elements at the corresponding positions are the same.)

(d) (10%) Merge two queues into 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 the functions** using at least one example, e.g., queue1=(1,3,5,7), queue2=(2,4,6,8), merged queue=(1,2,3,4,5,6,7,8)

(e) (10%) Reverse the queue (ReverseQueue()), that uses a stack object to reverse the

elements of the queue.

You should **demonstrate the functions** using at least one example, e.g., queue1=(1,3,5,7), stack = (), after reverse, queue1=(7,5,3,1).

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

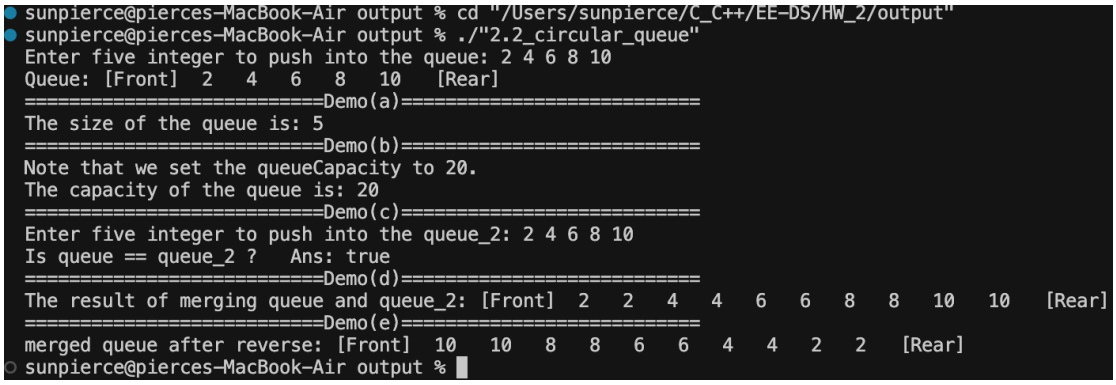
```
template < class T >
class Queue
{
public:
   Queue (int queueCapacity = 0);
      ~Queue();
      bool IsEmpty( ) const;
   void Push(const T& item);    // add an item into the queue
   void Pop( );    // delete an item
   T& Front() const;    // return top element of stack
   T& Rear() const; 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

sol:

The time complexity of my MergeQueue() function is O(m+n) where m and n are the size of the two queues.

Execution trace 1:

This is the case where queue == queue_2:



Execution trace 2:

This is the case where queue != queue_2:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"2.2_circular_queue"
Enter five integer to push into the queue: 1 3 5 7 9
Queue: [Front]  1   3   5   7   9   [Rear]
==========Demo(a)==========
The size of the queue is: 5
==========Demo(b)==========
Note that we set the queueCapacity to 20.
The capacity of the queue is: 20
==========Demo(c)==========
Enter five integer to push into the queue_2: 2 4 6 8 10
Is queue == queue_2 ?   Ans: false
==========Demo(d)==========
The result of merging queue and queue_2: [Front]  1   2   3   4   5   6   7   8   9   10   [Rear]
==========Demo(e)==========
merged queue after reverse: [Front]  10   9   8   7   6   5   4   3   2   1   [Rear]
The capcity of the merged queue: 40
sunpierce@pierces-MacBook-Air output %
```

3. (20%)

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 (using circular array). 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 (add PushFront() and PopRear() functions) and also to return an element from either end. The complexity of each function (excluding array doubling) should be $\Theta(1)$.

Sol:

Execution trace 1:

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"2.3_deque"
Enter eight integers to push into the deque:
1 2 3 4 5 6 7 8
Deque: [Front]  1   2   3   4   5   6   7   8   [Rear]
How many elements do you want to pop from the front?
7
Deque: [Front]  8   [Rear]
Enter ten integers to push into the front:
1 2 3 4 5 6 7 8 9 10
Deque: [Front]  10   9   8   7   6   5   4   3   2   1   8   [Rear]
How many elements do you want to pop from the rear?
5
Deque: [Front]  10   9   8   7   6   5   [Rear]
The element at the front is 10
The element at the rear is 5
sunpierce@pierces-MacBook-Air output %
```

Execution trace 2:

```
sunpierce@pierces-MacBook-Air EE-DS % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"2.3_deque"
Enter eight integers to push into the deque:
1 2 3 4 5 6 7 8
Deque: [Front]  1   2   3   4   5   6   7   8   [Rear]
How many elements do you want to pop from the front?
4
Deque: [Front]  5   6   7   8   [Rear]
Enter ten integers to push into the front:
1 2 3 4 5 6 7 8 9 10
Deque: [Front]  10   9   8   7   6   5   4   3   2   1   5   6   7   8   [Rear]
How many elements do you want to pop from the rear?
10
Deque: [Front]  10   9   8   7   [Rear]
The element at the front is 10
The element at the rear is 7
sunpierce@pierces-MacBook-Air output %
```

4. (15%)

Write a C++ program to implement the maze in textbook using the example codes of **Program 3.15** and **3.16**. 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) (5%) 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) (5%) 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).

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？）

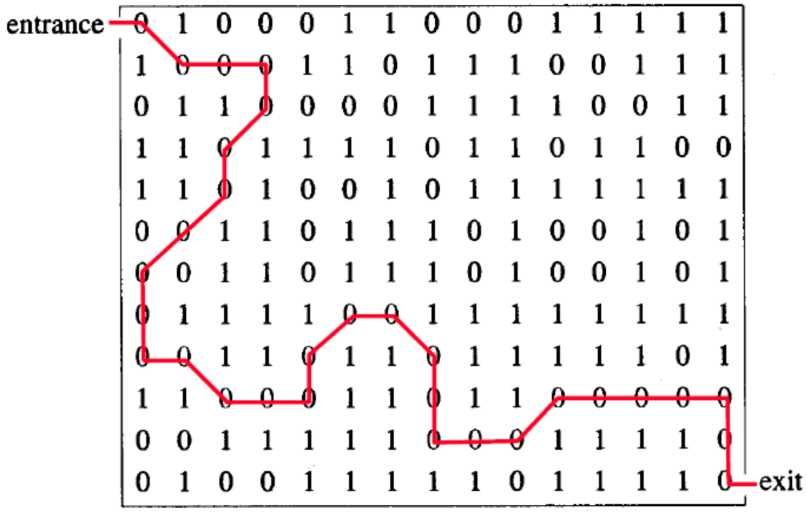
Sol:

(a) Execution trace:

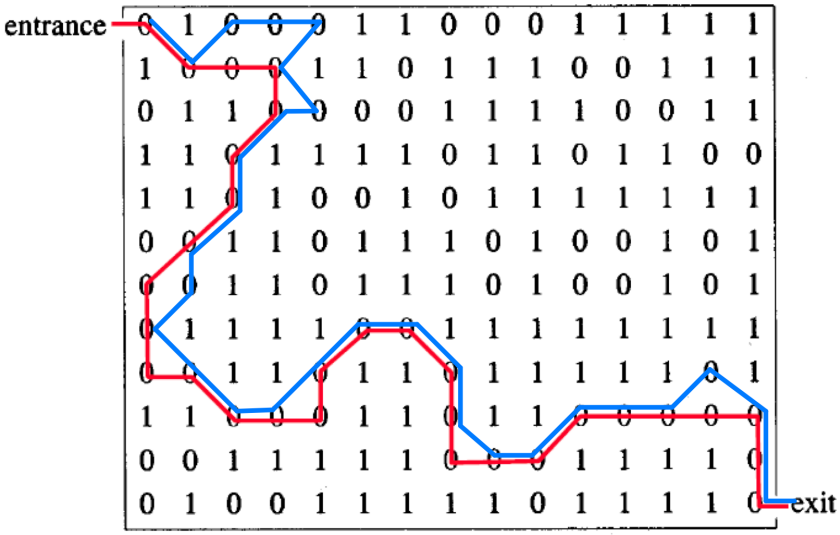
For simplicity, we use stack STL and instead of overloading << operator to stack, we design a print_stack() function to print out the path in the stack.

```
sunpierce@pierces-MacBook-Air output % cd "/Users/sunpierce/C_C++/EE-DS/HW_2/output"
sunpierce@pierces-MacBook-Air output % ./"2.4_maze"
Step 1: (1,1,SE)
Step 2: (2,2,NE)
Step 3: (1,3,E)
Step 4: (1,4,E)
Step 5: (1,5,SW)
Step 6: (2,4,SE)
Step 7: (3,5,W)
Step 8: (3,4,SW)
Step 9: (4,3,S)
Step 10: (5,3,SW)
Step 11: (6,2,S)
Step 12: (7,2,SW)
Step 13: (8,1,SE)
Step 14: (9,2,SE)
Step 15: (10,3,E)
Step 16: (10,4,NE)
Step 17: (9,5,NE)
Step 18: (8,6,E)
Step 19: (8,7,SE)
Step 20: (9,8,S)
Step 21: (10,8,SE)
Step 22: (11,9,E)
Step 23: (11,10,NE)
Step 24: (10,11,E)
Step 25: (10,12,E)
Step 26: (10,13,NE)
Step 27: (9,14,SE)
Step 28: (10,15,S)
The last two positions visited is (11,15) and (12,15).
sunpierce@pierces-MacBook-Air output %
```

(b) Path found manually (in red color):



(c) Path found by the program (in blue color):



For every step, the program find the next direction starting from north and clockwise, so the path found by the program would be different from the path found manually.