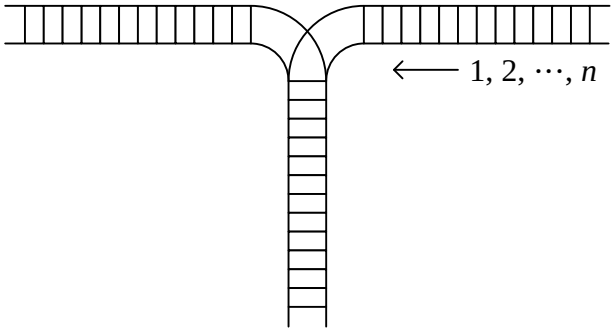
EECS2040 Data Structure Hw #2 (Chapter 3 Stack/Queue)
due date 4/11/2022 by 109070025 林泓錩

Part 1 (2% of final Grade)

1. (10%) Consider the railroad switching network shown below. (textbook pp.138-139)



Railroad cars can be moved into the vertical track segment one at a time from either of the horizontal segments and then moved from the vertical segment to any one of the horizontal segments. The vertical segment operates as a **stack** as new cars enter at the top and cars depart the vertical segment from the top. Railroad cars numbered 1,2,3…,n are initially in the top right track segment. Answer the following questions for n=3 and 4 cases:

(a) What are the possible permutations of the cars that can be obtained?

(b) Are any permutations not possible? If no, simply answer no. If yes, list them all.

Ans:

(a) If n=3, ans = 3! = 3*2*1 = 6.
If n=4, ans = 4! = 4*3*2*1 = 24.

(b) No, for instance Tower of Hanoi can build any permutations.

2. (15%) A linear list of type T objects is being maintained circularly in an array with front and rear set up as for **circular queues**.

(a) Draw a diagram of the circular array showing the initial status (values of front, rear) when the linear list is created (constructed) with no elements stored yet (assume capacity = 10 is used for creating the array)

(b) Obtain a formula in terms of the array capacity, front, and rear, for the number of elements in the list.

(c) Assume the kth element in the list is to be deleted, the elements after it should be moved up one position. Give a formula describing the positions of those

elements to be moved up one position in terms of k, front, rear, capacity. Design an algorithm (pseudo code) for this Delete(int k) member function.

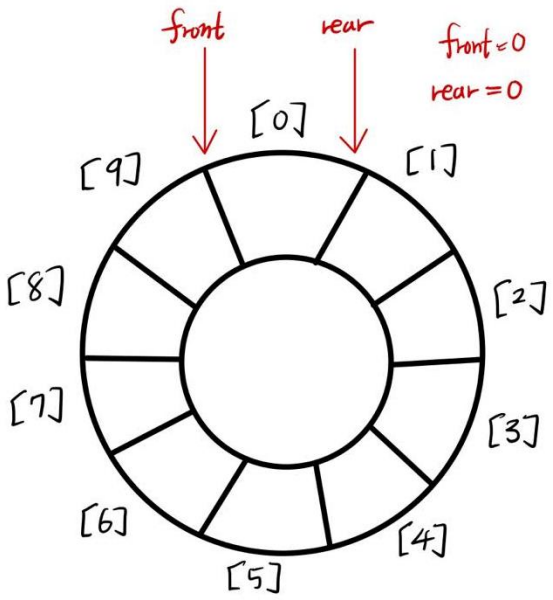
(d) Assume that we want to insert an element y immediately after the kth element. So the elements from the (k+1)th element on should be moved down one position in order to give space for y, which might cause insufficient capacity case in which array doubling will be needed. Describe the situation when array doubling is needed (in terms of k, front, rear, capacity). Design an algorithm (pseudo code) for this Insert(int k, T& y) member function.

(e) The following code segment is used for Push member function in circular queue described in textbook. Please explain the code in detail. (using a graphical illustration and explanation)

```
template <class T>
void Queue<T>::Push(const T& x)
{// add x to queue
    if ((rear + 1) % capacity == front) //resize
    {   T* newQu = new T[2*capacity];
        int start = (front+1) % capacity;
        if(start<2)
            copy(queue+start, queue+start+capacity-1, newQu);
        else{
            copy(queue+start, queue+capacity, newQu);
            copy(queue, queue+rear+1,newQu+capacity-start);
        }
        front = 2*capacity – 1;
        rear = capacity -2;
        delete[] queue;
        queue = newQu; capacity *=2;
    }
    rear = (rear+1)%capacity;    queue[rear] = x;
}
```

Ans:

(a)



(b) if (front > rear) number = capacity – front + rear;
else number = rear – front;

(c) When rear > front :
if (k == rear) nothing need to be moved.
else queue[k+1] to queue[rear] need to be moved.

When rear < front :
if (k == rear) nothing need to be moved.
else if (k < capacity–1 && k > front) queue[k+1] to queue[capacity - 1] and queue[0] to queue[rear] need to be moved.
else queue[(k+1)%capacity] to queue[rear] need to be moved.
Those elements need to be moved will moved from initial position i to position (i + capacity – 1) % capacity.

(d) When (front == rear), it means that the queue is going to be filled with data.
So when (rear + 1) % capacity == front, it's time to double the queue.

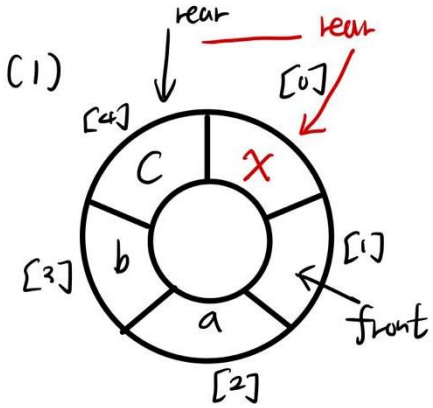
```
Insert(int k, T& y){
    if ((rear + 1) % capacity == front)
        queue.resize();    // ensure enough space
    // move first element to [0]
    for (i = rear; i > (front + k); i--) {
        data[(i + 1) % capacity] = data[i];
    }
    data[front + k] = y;
    if (rear == 0) rear = capacity – 1;
    else rear--;
```

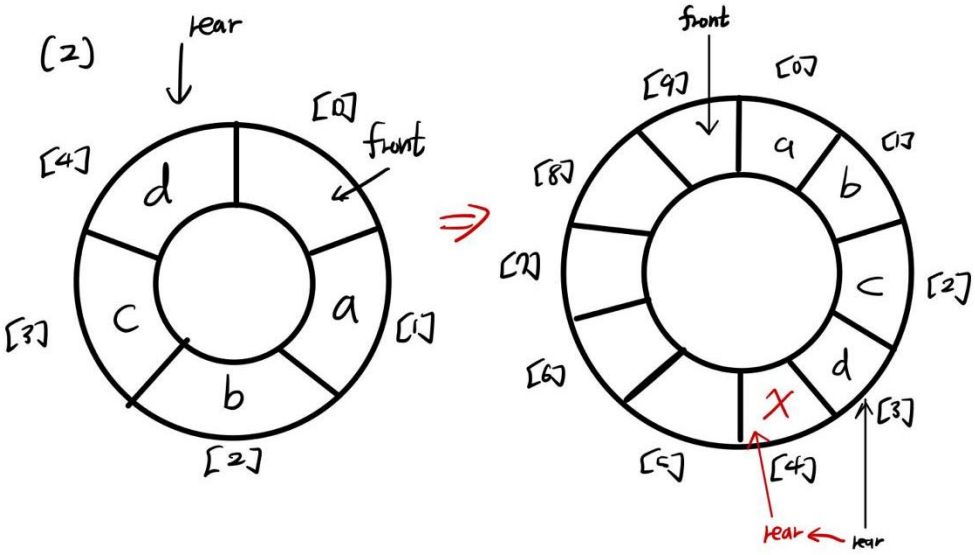
}

(e)

(1)If the space is enough, just added one element.

(2)If the space is not enough, call a new queue with 2 times of capacity, then start insert old data from [0], then delete old queue. Finally add one element in new queue.





3. (10%) Design an algorithm, 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.

Ans:

(1) First, we construct a new stack with capacity = the old queue's total terms,

(2) Then, push and pop all items form the old queue into stack.

(3) Next, create the new queue with same capacity of the old queue
(4) After that, push and pop all item from stack into the new queue.
(5) Finally, delete stack and we get the new queue with revered order respect to the old queue.

4. (5%) Given an integer k and a queue of integers, how do you reverse the order of the first k elements of the queue, leaving the other elements in the same relative order? For example, if k=4 and queue has the elements [10, 20, 30, 40, 50, 60, 70, 80, 90]; the output should be [40, 30, 20, 10, 50, 60, 70, 80, 90]. Design your algorithm for this task.

Ans:

(1) First, we create a stack, and the stack has capacity >= k (k is the number would like to reverse, in this question k = 4)
(2) Then, we pop out k data from the queue to the stack we constructed.
(3) After that, build the new queue that has the capacity = the original queue (in this question, it will be 9)
(4) Next, we pop out the data in the stack to the new queue.
(5) Last, pop out the remaining data in the original queue to the new one.
Then, the result will be [40, 30, 20, 10, 50, 60, 70, 80, 90].

5. (10%) Suppose that you are a financier and purchase 100 shares of stock in Company *X* in each of January, April, and September and sell 100 shares in each of June and November. The prices per share in these months were

Jan	Apr	Jun	Sep	Nov
$10	$30	$20	$50	$30

Determine the total amount of your capital gain or loss using (a) FIFO (first-in first-out) accounting and (b) LIFO (last-in, first-out) accounting [that is, assuming that you keep your stock certificates in (a) a queue or (b) a stack.]
The 100 shares you still own at the end of the year do not enter the calculation.

Ans:

(a) FIFO

Month / buy or sell	**Capital gain or loss**
January / buy	No gain or loss
April / buy	No gain or loss

June / sell	(20-10) *100 = 1000
September / buy	No gain or loss
November / sell	(30-30) *100 = 0
Total Capital Gain(Loss)	**1000**

(b)LIFO

Month / buy or sell	**Capital gain or loss**
January / buy	No gain or loss
April / buy	No gain or loss
June / sell	(20-30) *100 = -1000
September / buy	No gain or loss
November / sell	(30-50) *100 = -2000
Total Capital Gain(Loss)	**(3000)**

6. (15%) For the maze problem,
   (a) What is the maximum path length from start to finish for any maze of dimensions *m* x *p*?
   (b) Design a recursive version of algorithm for Path().
   (c) What is the time complexity of your recursive version?

Ans:

(a) If every block of the maze is available to walk, we will take m*p times to go through every block and this is also the longest path.
Maximum = m*p

(b) When we arrived one block, we have to choose the next direction to go, which we will have 8 choices, and later we move onto the next block, however, it is possible we meet the dead end, when that happened, we have to jump out of the recursive and back to the first block to try another direction. By doing so, we can check every block.

(c) 8*m*p

7. (10%) Using the operator priorities of Figure 3.15 (shown below**)** together with those for '(' and '#' to answer the following:

priority	operator
1	Unary minus, !
2	*, /, %
3	+, –
4	<, <=, >, >=
5	==, !=
6	&&
7	||

(a) In function Postfix (Program 3.19, pptx **Infix to Postfix Algorithm**), what is the maximum number of elements that can be on the stack at any time if the input expression has n operators and delimiters?

(b) What is the answer to (a) if the input expression e has n operators and the depth of nesting of parentheses is at most 6?

Ans:

(a) (n/9)*8 + (n%9), if (n%9) < 7
(n/9)*8 + 7, if (n%9) >= 7

(b) n+6 , n<49
55, n=49 <= max

8. (20%) Write the postfix form and prefix form of the following infix expressions:

(a) –A + B – C + D*A/B

(b) A * -B + C/D – A*C

(c) (A + B) /D + E / (F + A * D) + C

(d) A && B || C || !(E > F)

(e) !(A && !((B < C) || (C > E))) || (C < D)

Ans:

(a) Postfix form : A–B+C-DA*B/+
Prefix form : +-+-ABC/*DAB

(b) Postfix form : AB-*CD /+AC*-
Prefix form : +*A-B/CD

(c) Postfix form : AB+D/EFAD*+/+C+
Prefix form : ++/+ABD/E+F*ADC

(d) Postfix form : AB&&C||EF>!||
Prefix form : || || &&ABC!>EF

(e) Postfix form : ABC<CE>||!&&!CD<||
Prefix form : ||!&&A||!<BC>CE<CD

9. (10%) Evaluate the following postfix expressions:
(a) 8 2 + 3 * 16 4 / - =
(b) 12 25 5 1 / / * 8 7 + - =

Ans:

(a) 8 2 + 3 * 16 4 / - = (8 + 2) * 3 - (16 / 4) = 30 – 4 = 26
(b) 12 25 5 1 / / * 8 7 + - = 12 * (25 / (5 / 1)) – (8 + 7) = 60 – 15 = 45