EE2410 Data Structure Hw #2 (Chapter 3 Stacks & Queues of textbook)

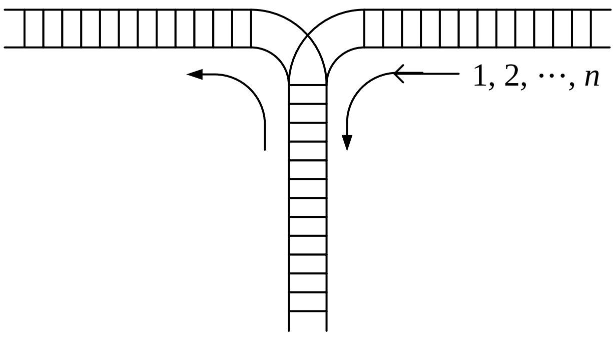
due date 4/14/2024(Sun.) 23:59

Student No.: ____111060005_____

Name: ______胡昱煊________

Note: Use MS Word to **edit this file** by directly typing your student number and name in above blanks and your answer to each homework problem right in the **Sol:** blanks as shown below. Then save your file as **Hw2-SNo.pdf**, where SNo is your student number. Submit the **Hw2-SNo.pdf** file via eLearn. The grading will be based on the correctness of your answers to the problems, and the **format requirement**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask. 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.

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 = 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.

Sol:

(a) 1234, 1243, 1324, 1342, 1432,
2134, 2143, 2314, 2341, 2431,

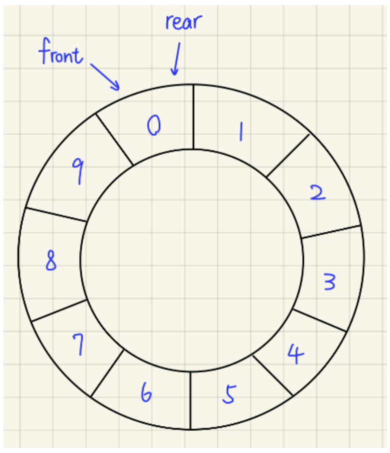
3214, 3241, 3421,
4321

(b) 1423, 2413, 3124, 3142, 3412, 4123, 4132, 4213, 4231, 4312

2. (10%) A linear list of type T objects is being maintained circularly in an array with front and rear set up as for **circular queues** as described in textbook.
   (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.

Sol:

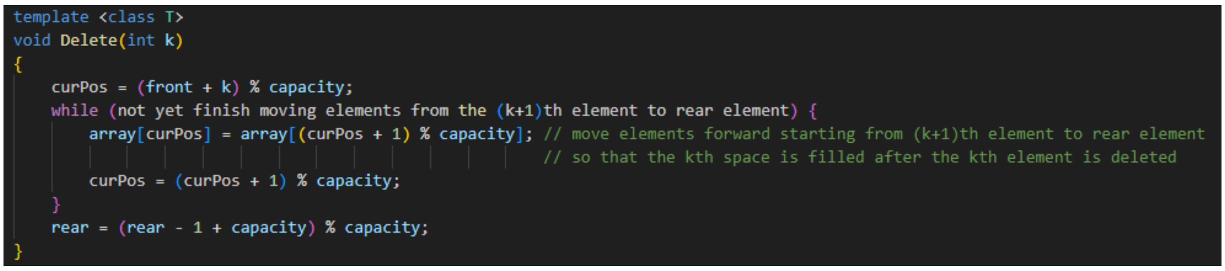
(a)



(b) Number of elements = (rear + capacity – front) % capacity

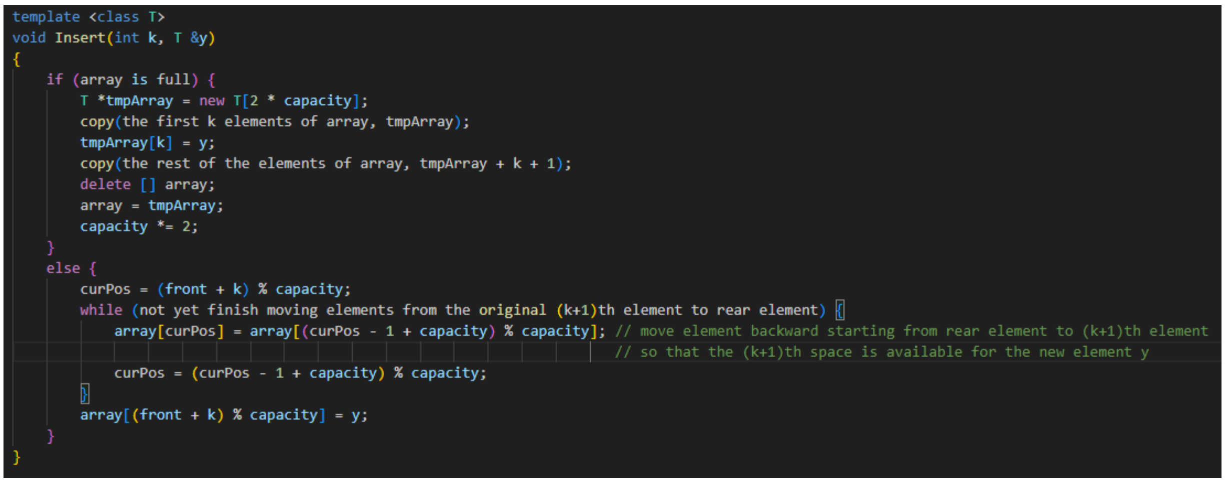
(c) New position $=$ Current position $- 1$, which means

$$array[front + k] = array[front + k + 1].$$



```
template <class T>
void Delete(int k)
{
    curPos = (front + k) % capacity;
    while (not yet finish moving elements from the (k+1)th element to rear element) {
        array[curPos] = array[(curPos + 1) % capacity]; // move elements forward starting from (k+1)th element to rear element
                                                         // so that the kth space is filled after the kth element is deleted
        curPos = (curPos + 1) % capacity;
    }
    rear = (rear - 1 + capacity) % capacity;
}
```

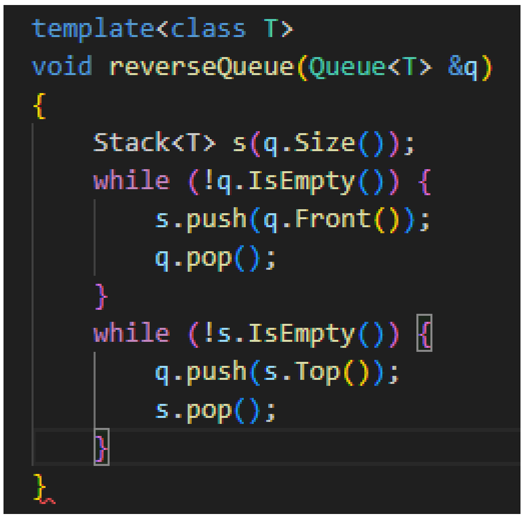
(d) When doubling the array is needed, we first copy the first k elements to tmpArray, insert y at tmpArray[k], which is the $k + 1^{th}$ position in tmpArray, then copy the rest of the elements in the original array to tmpArray starting from the position *(tmpArray + k + 1).



```
template <class T>
void Insert(int k, T &y)
{
    if (array is full) {
        T *tmpArray = new T[2 * capacity];
        copy(the first k elements of array, tmpArray);
        tmpArray[k] = y;
        copy(the rest of the elements of array, tmpArray + k + 1);
        delete [] array;
        array = tmpArray;
        capacity *= 2;
    }
    else {
        curPos = (front + k) % capacity;
        while (not yet finish moving elements from the original (k+1)th element to rear element) {
            array[curPos] = array[(curPos - 1 + capacity) % capacity]; // move element backward starting from rear element to (k+1)th element
                                                                        // so that the (k+1)th space is available for the new element y
            curPos = (curPos - 1 + capacity) % capacity;
        }
        array[(front + k) % capacity] = y;
    }
}
```

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.

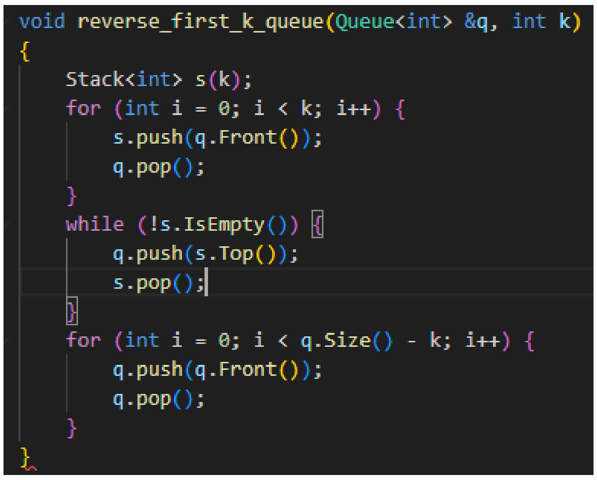
Sol:



```
template<class T>
void reverseQueue(Queue<T> &q)
{
    Stack<T> s(q.Size());
    while (!q.IsEmpty()) {
        s.push(q.Front());
        q.pop();
    }
    while (!s.IsEmpty()) {
        q.push(s.Top());
        s.pop();
    }
}
```

4. (10%) 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.

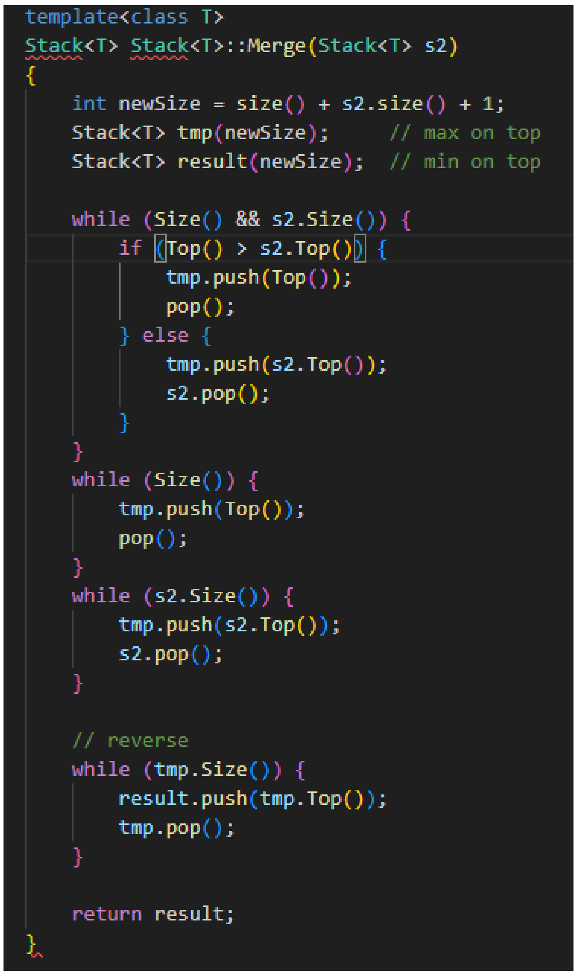
Sol:



```
void reverse_first_k_queue(Queue<int> &q, int k)
{
    Stack<int> s(k);
    for (int i = 0; i < k; i++) {
        s.push(q.Front());
        q.pop();
    }
    while (!s.IsEmpty()) {
        q.push(s.Top());
        s.pop();
    }
    for (int i = 0; i < q.Size() - k; i++) {
        q.push(q.Front());
        q.pop();
    }
}
```

5. (10%) Design a method that will take two sorted stacks A and B (min on top) and create one stack that is sorted (min on top) by merging these two stacks. You are allowed to use only the stack operations such as pop, push, size and top. No other data structure such as arrays are allowed. You are allowed to use stacks only. Assume that elements on the stack can be compared using > operator.

Sol:



```
template<class T>
Stack<T> Stack<T>::Merge(Stack<T> s2)
{
    int newSize = size() + s2.size() + 1;
    Stack<T> tmp(newSize);      // max on top
    Stack<T> result(newSize);  // min on top

    while (Size() && s2.Size()) {
        if (Top() > s2.Top()) {
            tmp.push(Top());
            pop();
        } else {
            tmp.push(s2.Top());
            s2.pop();
        }
    }
    while (Size()) {
        tmp.push(Top());
        pop();
    }
    while (s2.Size()) {
        tmp.push(s2.Top());
        s2.pop();
    }

    // reverse
    while (tmp.Size()) {
        result.push(tmp.Top());
        tmp.pop();
    }

    return result;
};
```

6. (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.

Sol:

(a) $100 \times \big((20-10)+(30-10)\big) = 1000$, earn 1000 dollars.

(b) $100 \times \big((20-30)+(30-50)\big) = -3000$, loss 3000 dollars.

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 e 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?

Sol:

(a)

condition	ans
$n \% 9 < 7$	$\left(\frac{n}{9}\right) * 8 + (n \% 9)$
$n \% 9 \geq 7$	$\left(\frac{n}{9}\right) * 8 + 7$

(b)

condition	ans
$n < 49$	$n + 6$
$n = 49$	55

8. (10%) Write the postfix form and prefix form of the following infix expressions:

(a) –A + B – C + D*A/C

(b) A * -D + B/D – A*B

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

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

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

Sol:

	postfix	prefix
(a)	$\mathrm{A}-\mathrm{B}+\mathrm{C}-\mathrm{DA}*\mathrm{C}/+$	$+-+-\mathrm{ABC}/*\mathrm{DAC}$
(b)	$\mathrm{AD}-*\mathrm{BD}/+\mathrm{AB}*-$	$-+*\mathrm{A}-\mathrm{D}/\mathrm{BD}*\mathrm{AB}$
(c)	$\mathrm{AC}+\mathrm{D}/\mathrm{EFAB}*+/+\mathrm{C}+$	$++/+\mathrm{ACD}/\mathrm{E}+\mathrm{F}*\mathrm{ABC}$
(d)	$\mathrm{AB}\ \&\&\ \mathrm{C}\vert\vert\ \mathrm{EF}>\ !\ \vert\vert$	$\vert\vert\ \vert\vert\ \&\&\ \mathrm{ABC}\ !>\mathrm{EF}$
(e)	$\vert\vert\ !\ \&\&\ \mathrm{A}\ !\ \vert\vert>\mathrm{BC}<\mathrm{CE}>\mathrm{CD}$	$\mathrm{ABC}>\mathrm{CE}<\vert\vert\ !\ \&\&\ !\ \mathrm{CD}>\vert\vert$

9. (10%) Evaluate the following postfix expressions:
   (a) 7 2 + 3 * 16 4 / - =
   (b) 12 25 5 1 / / * 8 7 + - =
   (c) 5 4 3 2 ^ + * 4 * 6 - =

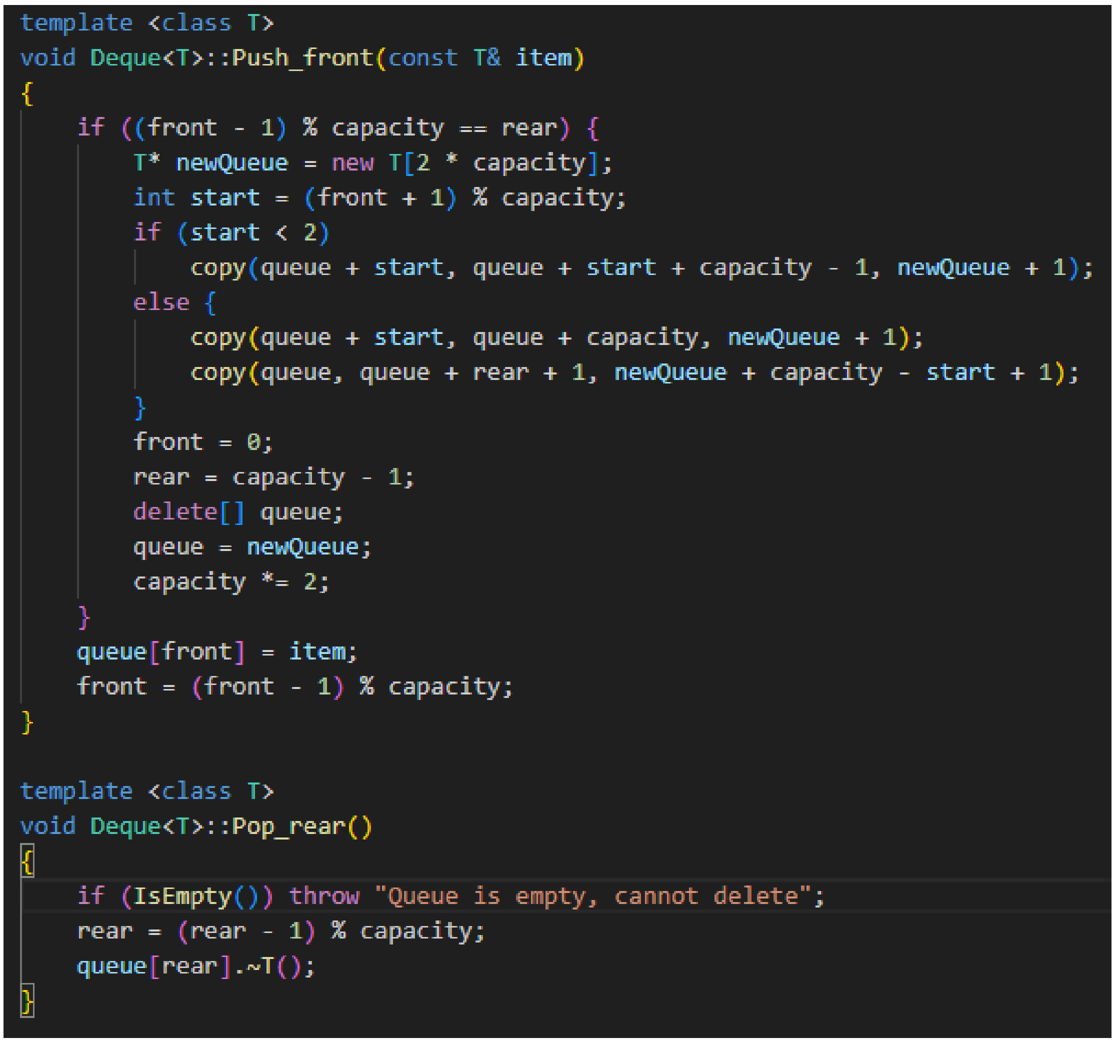
Sol:

(a)	$(7+2)*3-{}^{16}/_{4}=23$
(b)	$12*\left({}^{25}/_{({}^{5}/_{1})}\right)-(8+7)=45$
(c)	$5*(4+3^2)*4-6=254$

10. (10%) Based on the templated circular queue ADT in Textbook, we want to modify it to become a templated deque by adding two more member functions.
    (a) Describe how to accomplish the added two more functions in text or pseudo code.
    (b) Design an ADT of this deque and a C++ class for implementing this ADT. (no need to write C++ codes for member functions.)

Sol:

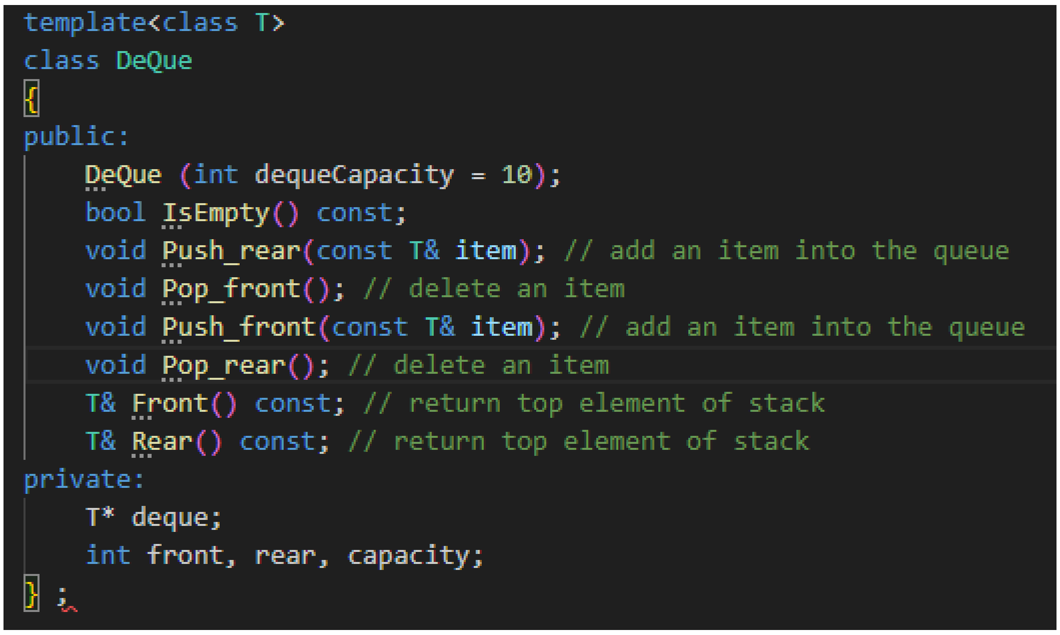
(a)



```
template <class T>
void Deque<T>::Push_front(const T& item)
{
    if ((front - 1) % capacity == rear) {
        T* newQueue = new T[2 * capacity];
        int start = (front + 1) % capacity;
        if (start < 2)
            copy(queue + start, queue + start + capacity - 1, newQueue + 1);
        else {
            copy(queue + start, queue + capacity, newQueue + 1);
            copy(queue, queue + rear + 1, newQueue + capacity - start + 1);
        }
        front = 0;
        rear = capacity - 1;
        delete[] queue;
        queue = newQueue;
        capacity *= 2;
    }
    queue[front] = item;
    front = (front - 1) % capacity;
}

template <class T>
void Deque<T>::Pop_rear()
{
    if (IsEmpty()) throw "Queue is empty, cannot delete";
    rear = (rear - 1) % capacity;
    queue[rear].~T();
}
```

(b)



```
template<class T>
class DeQue
{
public:
    DeQue (int dequeCapacity = 10);
    bool IsEmpty() const;
    void Push_rear(const T& item); // add an item into the queue
    void Pop_front(); // delete an item
    void Push_front(const T& item); // add an item into the queue
    void Pop_rear(); // delete an item
    T& Front() const; // return top element of stack
    T& Rear() const; // return top element of stack
private:
    T* deque;
    int front, rear, capacity;
} ;
```