Example: Template Class

We want to implement a template `Arrray` class that can

- hold a pre-defined number of values of the template parameter type,
- set values at a certain position,
- get values from a certain position,
- display the array content.

Example Problem 1:

You are given the `Stack` class definition below, implemented as discussed in the lecture.

```
const int MaxSize = 50; //constant capacity
template <class T>
class Stack{
      private:
            T slist [MaxSize];
            int top;
      public:
            Stack(void);
            void Push (const T &item);
            T Pop (void);
            int Stack_Empty (void) const;
            int Stack_Full (void) const;
};
```

Now, you are required to implement a `Queue` class using ONLY stacks (i.e., objects that belong to the class given above,) to store the data items. That is, you can NOT use an array to store the items in the queue, you have to use a stack for that purpose.

a. Complete the private part of the following template-based `Queue` class definition:

```
template <class T>
class Queue{
      private:

      ..............................
      public:
            Queue (void);
            void QInsert (const T &item);
            T QRemove (void);
            int QEmpty (void);
            int QFull (void);
};
```

```
MaxSize = 50; //constant capacity
template <class T>
class Stack{
      private:
            T slist [MaxSize];
            int top;
      public:
            Stack(void);
            void Push (const T &item);
            T Pop (void);
            int Stack_Empty (void) const;
            int Stack_Full (void) const;
}
```

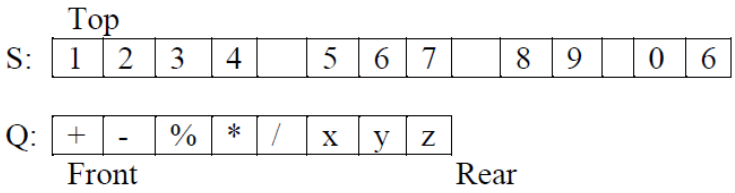
```
template <class T>
class Queue{
      private:

      ..............................
      public:
            Queue (void);
            void QInsert (const T &item);
            T QRemove (void);
            int QEmpty (void);
            int QFull (void);
}
```

b. Give the implementation of the `QInsert` and `QRemove` member functions of the `Queue` class, to insert an item at the rear of a queue and to remove the item at the front of the queue, respectively.

c. Give the implementations of the member functions `QEmpty` and `QFull` of the `Queue` class, to test whether the queue is empty or full, respectively.

Example Problem 2: Stacks and Queues



a. Write the output generated when the code given below is run assuming that the initial contents of S and Q is as given above.

```
char c;
while (!S.StackEmpty()){
    while(!S.StackEmpty() && !(S.Peek() == ' ') ){
        S1.Push(S.Pop());
    }
    if(!S.StackEmpty())
        c=S.Pop();
    if (!Q.QEmpty() && !S.StackEmpty() )
        S1.Push(Q.QDelete());
}
while (!S1.StackEmpty())
    cout << S1.Pop() << "\t";
```

Output

Top

S:

1	2	3	4		5	6	7		8	9		0	6

Q:

+	-	%	*	/	x	y	z

Front Rear

b. Repeat part a. for the code given below assuming that the initial contents of S and Q is as given above.

```
Stack<char> S1, S2;
Queue<char> Q1;
char c;
while (!S.StackEmpty() )
    S1.Push(S.Pop());
while (!S1.StackEmpty()){
    while (!S1.StackEmpty() && !(S1.Peek()==' ') )
        S2.Push(S1.Pop());
    if (!S1.StackEmpty())
        c=S1.Pop();
    if (!Q.QEmpty() )
        S2.Push(Q.QDelete());
    while (!S2.StackEmpty()&&!(S2.Peek()==' ') )
        Q1.QInsert(S2.Pop());
}
while (!Q1.QEmpty() )
    cout << Q1.QDelete() << "\t";
```

Output:
