Tutorial – Stack and Queues

1. (**reverseStack**) Write a Python function `reverse_stack()` that reverses a stack using a queue. Note that the `reverse_stack()` function only uses `push()` and `pop()` when adding or removing integers from the stack, and only uses `enqueue()` and `dequeue()` when adding or removing integers from the queue.

   The function prototypes are given as follows:
   ```
   def reverse_stack(stack: list):
   ```

   For example: if the stack is <5, 4, 3, 2, 1>, the resulting stack will be <1, 2, 3, 4, 5>

   Sample test cases are given below:
   ```
   1: Insert an integer into the stack;
   2: Reverse the stack;
   0: Quit;

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 1
   The resulting stack is: 1

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 2
   The resulting stack is: 2 1

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 3
   The resulting stack is: 3 2 1

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 4
   The resulting stack is: 4 3 2 1

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 5
   The resulting stack is: 5 4 3 2 1


   Please input your choice(1/2/0): 2
   The resulting stack after reversing its elements is: 1 2 3 4 5

   Please input your choice(1/2/0):0
   ```

2. (**reverseFirstKItems**) Write a Python function `reverse_first_k_items()` that reverses the order of the first k elements of a queue using a stack, leaving the other elements in the same relative order for a given integer k and a queue of integers. Note that the `reverse_first_k_items()` function only uses `push()` and `pop()` when adding or removing integers from the stack, and only uses `enqueue()` and `dequeue()` when adding or removing integers from the queue.

   The function prototypes are given as follows:
   ```
   def reverse_first_k_items(queue: list, k: int):
   ```

   For example: if the queue is <1, 2, 3, 4, 5, 6> and k =3,

the resulting queue will be <3, 2, 1, 4, 5, 6>

Sample test cases are given below:

```
1: Insert an integer into the queue;
2: Reverse the elements of the queue until the given number;
0: Quit;

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 1
The resulting queue is: 1

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 2
The resulting queue is: 1 2

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 3
The resulting queue is: 1 2 3

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 4
The resulting queue is: 1 2 3 4

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 5
The resulting queue is: 1 2 3 4 5

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the queue: 6
The resulting queue is: 1 2 3 4 5 6

Please input your choice(1/2/0): 2
Enter an integer to reverse the queue until that number: 3
The resulting queue after reversing first 3 elements is: 3 2 1 4 5 6

Please input your choice(1/2/0): 0
```

3. (**sortStack**) Write a Python function `sort_stack()` that sorts a given stack in ascending order using another temporary stack. Note that the `sort_stack()` function only uses `push()` and `pop()` when adding or removing integers from the stack.

   The function prototype is given as follows:

   ```
   def sort_stack(stack: list)
   ```

   For example, if the stack is <6, 5, 4, 3, 2, 1>, the resulting stack will be <1, 2, 3, 4, 5, 6>

   Sample test cases are given below:

   ```
   1: Insert an integer into the stack;
   2: Sort the stack in ascending order ;
   0: Quit;

   Please input your choice(1/2/0): 1
   Input an integer that you want to insert into the stack: 1
   The resulting stack is: 1


   Please input your choice(1/2/0): 1
   ```

```
Input an integer that you want to insert into the stack: 2
The resulting stack is: 2 1

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 3
The resulting stack is: 3 2 1

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 4
The resulting stack is: 4 3 2 1

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 5
The resulting stack is: 5 4 3 2 1

Please input your choice(1/2/0): 1
Input an integer that you want to insert into the stack: 6
The resulting stack is: 6 5 4 3 2 1

Please input your choice(1/2/0): 2
The resulting stack after sorting it in ascending order is:
1 2 3 4 5 6

Please input your choice(1/2/0): 0
```

4. Given the precedence of some operators,

Operators	Precedence
*, /, %	highest
+, -	
<<, >>	
&&	
=	lowest

(a) convert an infix expression, $x = a + b * c\%d >> e$, to a postfix expression
(b) convert a prefix expression, $= y\&\& << ab >> c + de$, to an infix expression
(c) convert a postfix expression, $xabc * d\% + e >>=$, to a prefix expression