LAB 2

<u>Lưu ý:</u>

- Làm bài vào file word bằng tiếng Việt hoặc tiếng Anh
- Những bài làm giống nhau sẽ bị 0 điểm
- Với những bài lập trình, cần phải copy mã nguồn và chụp màn hình kết quả, đưa vào file word
- Địa chỉ nộp bài: ctdlgt.bku@gmail.com
- Hạn chót nộp bài: 23 giờ ngày 6/7/2023

Suppose that the following algorithms are implemented:

- `PushStack (ref s <Stack>, val n <data>)` : push the value **n** to the stack **s**.
- `PopStack(ref s <Stack>, ref x <data>)` : remove the top element of the stack **s** and assign the data of that top element to variable **x**
- `EmptyStack(val s <Stack>)` : check whether the stack **s** is empty.

<u>Question 1:</u>

Imagine we have two empty stacks of integers, s1 and s2. Draw a picture of each stack at the end of line 8, 14, 18. The following operations:
(With a1, a2, a3, a4 are 4 respectively last number in your student ID number. Ex: Student ID number is 1416789 → a1=6, a2=7, a3=8, a4=9)

```
1:  PushStack (s1, a1);
2:  PushStack (s1, a2);
3:  PushStack (s1, a3);
4:  PushStack (s1, a4);
5:  while (!EmptyStack (s1)) {
6:    PopStack (s1, x);
7:    PushStack (s2, x);
8:  } //while
9:  PushStack (s1, a1*a3);
10:  PushStack (s1, a2*a4);
11:  while (!EmptyStack (s2)) {
12:    PopStack (s2, x);
13:    PushStack (s1, x);
14:  } //while
15:  PopStack (s1, x);
16:  PushStack (s2, x);
17:  PopStack (s1, x);
18:  PushStack (s2, x);
```

<u>Question 2:</u>

a. Adjust the code in Question 1 at lines 6 and 7 *(you can remove or insert some code statements in there, use temporary other stacks)* to print out 4 last number in your student ID number in order, just use stack and operations on it, the rest must not change.

b. Draw a picture of each stack *(s1 and s2)* after finishing above operations *(were changed in question 2a)*.

Question 3:

Write an algorithm for a function called *RemoveN* that removes the element at position N of a stack (the bottom element is the 1st element in order). The order of other elements in the stack must be the same after the removal.

```
algorithm RemoveN (ref sourceStack<Stack>, val N <data>)
    This algorithm removes the N-th element in the sourceStack. The order of the
    remaing elements must be preserved after the removal.

    Pre None
    Post the sourceStack being removed its N-th element
    Return  None

end RemoveSecond
```

///

Suppose that the following algorithms are implemented:

- `EnQueue (ref q <Queue>, val n <data>)` : push the value n to the queue queue
- `DeQueue(ref q <Queue>, ref x <data>)` : remove the top element of the queue q and assign the data of that top element to x
- `EmptyQueue(val q <Queue>)` : check whether the queue q is empty
- `QueueFront()` : return the first element on the front
- `QueueRear()` : return the last element in the rear

Question 4:

Imagine we have an empty stack of integers S, and two empty queues of integer Q1 and Q2. What would be the value of queues Q1, Q2, and stack S, after the following segment?
(stuID is an integer array of your student ID number)

```
1 i = 0
2 loop (i < length of stuID)
   1 Enqueue (Q1, stuID[i])
   2 i = i + 1
3 end loop
4 loop (not EmptyQueue Q1)
   1 DeQueue (Q1, x)
   2 if (x is 0)
      1 z = 0
      2 loop (not EmptyStack S)
         1 PopStack(S, &y)
         2 z = z + y
      3 end loop
      4 Enqueue (Q2, z)
   3 else
      1 PushStack (S, x)
   4 end if
5 end loop
```

Question 5:

What would be the contents of queue Q1 after the following code is executed and the following data are entered?
(With a1, a2, a3, a4, a5 are 5 respectively last number in your student ID number. Ex: Student ID number is 1416789 → a1=1, a2=6, a3=7, a4=8, a5=9)

a.

```
1 Q1 = createQueue
2 S1 = createStack
3 loop (not end of file)
    1 read number
    2 if (number not 0)
        1 PushStack (S1, number)
    3 else if (not empty S1)
        1 PopStack (S1, x)
        2 loop (not empty S1)
            1 PopStack (S1, x)
            2 EnQueue (Q1, x)
        3 end loop
    4 end if
4 end loop
```

b.

```
Q1 = createQueue
while (not end of file){
  read number
  if (number != 0){
    EnQueue (Q1, number)
  }else{
    QueueRear (Q1, x)
    EnQueue (Q1, x)
      }
 }
```

Line	File	Example
1	a1+a2	7
2	a2+a3	13
3	a3+a4	15
4	a3-a4	-1
5	a4+a5	17
6	a5+a1	10
7	a2-a3	-1
8	a1+a2-a3	0
9	a2-a3+a4	7
10	a3+a4+a5	24
11	a1-a2	-5
12	a1*a2	6
13	a2*a3	42
14	a3*a4	56
15	a1+a2*a3	43
16	a5+a1+a2	16
17	a5-a1	8
18	a2+a3*a4	62
19	0	0
20	a3+a4*a5	79

//

Question 6: Implement a stack along with some basic operations:

a. Create: Creates an empty linked stack.
b. Push: Pushes new data into a stack.
c. Pop: Pops an element from the top of a stack.
d. Top: Retrieves data on the top of a stack without changing the stack.
e. isEmpty: Determines if a stack is empty.
f. isFull: Determines if a stack is full.
g. Clear: Clear a stack to make it empty.
h. Size: Determines the current number of elements in a stack.

Question 7: Implement a queue along with some basic operations:

a. Create: Creates an empty linked queue.
b. EnQueue: Inserts one element at the rear of a queue.
c. DeQueue: Deletes one element at the front of a queue.
d. QueueFront: Retrieves data at the front of a queue without changing the queue.
e. QueueRear: Retrieves data at the rear of a queue without changing the queue.
f. isEmpty: Determines if a queue is empty.
g. isFull: Determines if a queue is full.
h. Clear: Clear a queue to make it empty.
i. Size: Determines the current number of elements in a queue.