Tut 4 – C++ Stack & Queue

With the following struct:

```
struct node {
	int data;
	node *next;
};

struct list {
	node *pHead;
	node *next;
};

struct stack {
	node *top;
	int count;
};

struct queue{
	node *front;
	node *rear;
	int count;
};
```

Problem 1 - Stack

Suppose that the following algorithms are implemented, , use for problem from 1 to 3:

- `void PushStack(stack *s, int n)`: push the value n to the stack s.
- `void PopStack(stack *s, int &x)`: remove the top element of the stack s and assign the data of that top element to variable x.
- `bool IsStackEmpty(stack *s)`: check whether the stack s is empty.

Imagine we have two empty stacks of integers, s1 and s2. Show the state of each stack at the end of line 17, 25, 30 in the following operations: (With a1, a2, a3, a4 are 4 respectively last number in your student ID number. Ex: Student ID number is 1616789, so we have a1=6, a2=7, a3=8, a4=9)

```
1   void main(){
2
3       int a1 = 6, a2 = 7, a3 = 8, a4 = 9;//if your student ID is 1616789
4       stack *s1 = new stack();
5       stack *s2 = new stack();
6       int x;
7
8       PushStack(s1, a1);
9       PushStack(s1, a2);
10      PushStack(s1, a3);
11      PushStack(s1, a4);
12
13      while (!IsStackEmpty(s1)){
14          PopStack(s1, x);
15          PushStack(s2, x);
16      } //While
17      //TODO1: show the state of s1 and s2
18      PushStack(s1, a1*a3);
19      PushStack(s1, a2*a4);
20
21      while (!IsStackEmpty(s2)){
22          PopStack(s2, x);
23          PushStack(s1, x);
24      } //While
25      //TODO2: show the state of s1 and s2
26      PopStack(s1, x);
27      PushStack(s2, x);
28      PopStack(s1, x);
29      PushStack(s2, x);
30      //TODO3: show the state of s1 and s2
31
32      system("pause");
33  }
```

Problem 2 - Stack

a) Adjust the code in Problem 1 at lines 14 and 15 *(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) Show the state of each stack *(s1 and s2)* after finishing above operations *(were changed in problem 2a).*

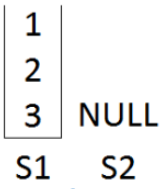
Problem 3 - Stack

Write a function to copy a stack with the prototype:

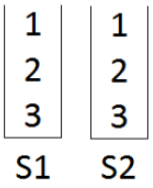
```
stack* copyStack(stack* s)
```

For example:

```
stack* s2 = NULL;
```




```
s2 = copyStack(s1);
```



Problem 4 - Stack

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 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 element

Return None

end RemoveN

Problem 5 - Queue

Suppose that the following algorithms are implemented, use for problem from 4 to 6:

- `void EnQueue(queue *q, int n)` : push the value n to the queue.
- `void DeQueue (queue *q, int &x)` : remove the top element of the queue q and assign the data of that top element to x.
- `bool IsQueueEmpty(queue *q)` : check whether the queue q is empty.
- `int GetFront(queue *q)`: return the first element on the front.
- `int GetRear(queue *q)`: return the last element in the rear.

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 and size is stuID length*)

```
queue* Q1 = new queue();
queue* Q2 = new queue();
stack* S = new stack();
for (int i = 0; i<size; i++) {
      EnQueue(Q1, stuID[i]);
}

while (!IsQueueEmpty(Q1)) {
      int x;
      DeQueue(Q1, x);
      if (x % 3 == 2) {
            int z = 1;
            int y;
            while (!IsStackEmpty(S)) {
                  PopStack(S, y);
                  z *= y;
            }
            EnQueue(Q2, z);
      }
      else {
            PushStack(S, x);
      }
}
```

Problem 6 - Queue

Assume that we have a queue of integers. Write a function to remove all values greater than the mean value of the queue.

```
void* removeElement (queue* q)
```