تمرین سری دوم ساختمان داده
نیکا سلطانی تهرانی - ۹۷۳۱۷۰۲

سوال اول)

می‌دانیم که یک استک، last in-first out است. هر آیتمی که دیرتر از همه اضافه شود، زودتر از استک خارج می‌شود. همچنین یک صف first in-first out است. هر آیتمی که دیرتر از همه اضافه شود، زودتر از صف خارج می‌شود. فرض کنید سه عدد ۱ و ۲ و ۳ را به ترتیب در یکی از استک‌ها پوش می‌کنیم. اگر سه بار عمل pop را روی این استک انجام دهیم، به ما خروجی ۳ و ۲ و ۱ را می‌دهد درحالیکه اگر ساختمان داده ما صف باشد، خروجی ۱ و ۲ و ۳ را انتظار داریم. یعنی انتظار خروجی برعکس خروجی استک را داریم، پس اگر بخواهیم با استفاده از دو استک یک صف بسازیم، به این صورت عمل می‌کنیم که:

- اگر متد enqueue صدا زده شده باشد، اعداد به سادگی در استک اول push می‌شوند.
- اگر متد dequeue صدا زده شود، ابتدا اعداد موجود در استک اول از آن pop شده و در استک دوم push می‌شوند تا ترتیب آنها برعکس شود. سپس خروجی نهایی با pop شدن اعداد از استک دوم ایجاد می‌شوند.

در تحلیل پیچیدگی زمانی این الگوریتم باید گفت که dequeue کردن عدد جدید روی صف از مرتبه O(1) است؛ زیرا کافی است عدد موردنظر به انتهای استک اول اضافه شود و عمل dequeue روی صف از مرتبه O(n) است؛ چون نیاز است یک بار همه اعداد از استک اول در استک دوم push شوند و بعد عمل pop روی استک دوم انجام شود.

```java
import java.util.Stack;

public class Q1 {
    public static void main(String[] args){
        Stack<Object> stack1 = new Stack<>();
        Stack<Object> stack2 = new Stack<>();
        stack1.push( item: 1);
        stack1.push( item: 2);
        stack1.push( item: 3);

        if(stack1.size() == 0){
            if(stack2.size() == 0){
                System.out.println("Empty");
            }}
        while(stack1.size() > 0){
            Object p = stack1.pop();
            stack2.push(p);
        }
        System.out.println(stack2);
    }
}
```

```
Q1
"C:\Program Files\Java\jdk-15\bin\java.exe" "-javaagent
[3, 2, 1]
```

تمرین سری دوم ساختمان داده
نیکا سلطانی تهرانی - ۹۷۳۱۷۰۲

سوال دوم)

یکی از الگوریتم‌های موجود برای اینکه متوجه شویم در یک لیست پیوندی، حلقه وجود دارد، این است که روی گره‌های لیســـت پیوندی حرکت کنیم و آدرس هر گره را در یک hash set ذخیره کنیم. اگر آدرســـی بود که دوبار تکرار شده بود، نشان می‌دهد که در لیست پیوندی حلقه داریم.

```
static boolean detectLoop(Node h)
{
    HashSet<Node> s = new HashSet<>();
    while (h != null) {
        if (s.contains(h))
            return true;
        s.add(h);
        h = h.next;
    }
    return false;
}
```

چون یکبار نیاز است تا روی همه گره ها پیمایش شود، مرتبه زمانی این الگوریتم O(n) است و پیچیدگی فضایی آن هم به این دلیل که نیاز اســت به تعداد همه گره ها آدرس ذخیره ســازی آن ها ذخیره شــود، از مرتبه O(n) است.

```
import java.util.*;
public class LinkedList {
    static Node head;
    static class Node {
        int data;
        Node next;
        Node(int d)
        {
            data = d;
            next = null;
        }
    }
    static public void push(int new_data)
    {
        Node new_node = new Node(new_data);
        new_node.next = head;
        head = new_node;
    }
    static boolean detectLoop(Node h)
    {
        HashSet<Node> s = new HashSet<>();
        while (h != null) {
            if (s.contains(h))
                return true;
```

```
            s.add(h);
            h = h.next;
        }
        return false;
    }
    public static void main(String[] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(10);
        llist.head.next.next.next.next = llist.head;
        if (detectLoop(head))
            System.out.println("Loop found");
        else
            System.out.println("No Loop");
    }
}
```