Data Structure Homework 2 (20 pts)

1. Due Time: March 26th (Thursday), 23:59PM.
2. Students may work in pairs.
3. Late submission will be heavily penalized.
4. If you cannot finish all, submit solutions for some problems to get partial credits.

Problem description:

The purpose of the project is to implement a Stack ADT in the two most common ways, an array and a linked list. Your stack implementation will be used to manipulate a string of number to solve the pancake flipping problem (ACM-ICPC contest problem).

The cook at the Frobbozz Magic Pancake House sometimes falls asleep on the job while cooking pancakes. As a result, one side of a stack of pancakes is often burned. Clearly, it is bad business to serve visibly burned pancakes to the patrons. Before serving, the waitress will arrange the stacks of pancakes so that the burned sides are facing down. You must write a program to aid the waitress in stacking the pancakes correctly.

We start with a stack of N pancakes of distinct sizes, each of which is burned on one side. The problem is to convert the stack to one in which the pancakes are in size order with the smallest on the top and the largest on the bottom and burned side down for each pancake. To do this, we are allowed to flip the top k pancakes over as a unit (so the kth pancake is now on top and the pancake previously on top is now in the kth position and the burned side goes from top to bottom and vice versa). For example (+ indicates burned bottom, - a burned top):
+1 -3 -2 [flip 2] => +3 -1 -2 [flip 1] => -3 -1 -2 [flip 3] => +2 +1 +3 [flip 1] => -2 +1 +3 [flip 2] => -1 +2 +3 [flip 1] => +1 +2 +3

You must write a program with a stack ADT for flipping, which finds a sequence of at most (3n-2) flips, which converts a given stack of pancakes to a sorted stack with burned sides down.

Input: A sequence of numbers separated by spaces, with the first number being the number, N, of pancakes in the data set. The remainder of the data set is the numbers 1 through N in some order, each with a plus or minus sign, giving the initial pancake stack. The numbers indicate the relative sizes of the pancakes and the signs indicate whether the burned side is up (-) or down (+). M can be an arbitrary positive integer.

Output: You should generate one line of output with the following values: the number of flips (K, where K≥0) required to sort the pancakes and a sequence of K numbers, each of which gives the number of pancakes to flip on the corresponding sorting step. There may be several correct solutions

for some datasets. For instance '3 2 3' is also a solution to the first problem below.

Sample Input	Sample Output
3 +1 -3 -2	6 2 1 3 1 2 1 (or 3 2 3)
4 -3 +1 -2 -4	6 4 1 4 3 1 2
5 +1 +2 +3 +4 -5	3 3 5 1 5

Score breakdown:

1. [10] Implement the `Stack<E>` interface found on page 229 of the textbook (also available on Blackboard) using an array and linked list, called `ArrayStack` and `ListStack`, respectively. Your array implementation have to start with a small array (say, 3) and resize to use an array twice as large whenever the array becomes full, copying over the elements in the smaller array. While there are convenient Java library methods for copying arrays, for this assignment, use your own loop to copy array elements manually (so you can "see" the work involved in copying).
2. [10] Design an algorithm for sorting, which should find a sequence of at most (3n-2) flips. Implement the `FlipPancake.java` program, with the command line argument:

```
array/list N [pancakes]
```

In summary, you have to create the following java files, named as follows:

- `FlipPancake.java`
- `ArrayStack.java`
- `ListStack.java`
- `Stack.java`

About Submission (Please read carefully, otherwise you may lose points).

3. Make sure you have read the programming guidelines posted on Blackboard for coding conventions, in order not to lose credit.
4. Only submit source files. Do NOT include any executables. All files should be saved in a folder and then packed into a single .zip (NOT .rar or .tar.gz) file and be submitted via blackboard (NOT emailing).
5. The folder name (before compression) as well as the final zip file name should be "LastNames-HW2".
6. Only one submission per team, and **full names** of the members should be included in the *Comments* box of the submission page on Blackboard.
7. Ensure your code can be compiled and executed in command line (not a java IDE). Otherwise, you

will NOT get any credits.

8. In the zip file, include a text file README.txt for the following information:
   - Description of your algorithm for flipping the pancakes and the analysis of the worse-case being (3n-2) flips.
   - On which platform (mac, linux, window) the code is compiled and executed
   - "Resources that Helped Me", as discussed in the syllabus.
9. This is NOT something you can finish in three days. To understand the problem itself takes quite some time. You have to start as early as possible.