German University in Cairo
Media Engineering and Technology
Prof. Dr. Slim Abdennadher

Data Structures and Algorithms, Winter term 2020
Practice Assignment 3

Exercise 3-1 Search in a Stack

Write a method to find the position of a given element in a stack counting from the top of the stack. More precisely, the method should return `0` if the element occurs on the top, `1` if there is another element on top of it, and so on. If the element occurs several times, the topmost position should be returned. If the element doesn't occur at all, `-1` must be returned.

You are asked to write this method in **two different ways**; one way is to implement it **internally** inside the `ArrayStack` class and the other way is to implement it **externally** in a separate class. **Important**: At the end the stack should be returned to the original state (i.e. no elements should be removed and the order of the elements should not change).

Exercise 3-2 Stack Decompose

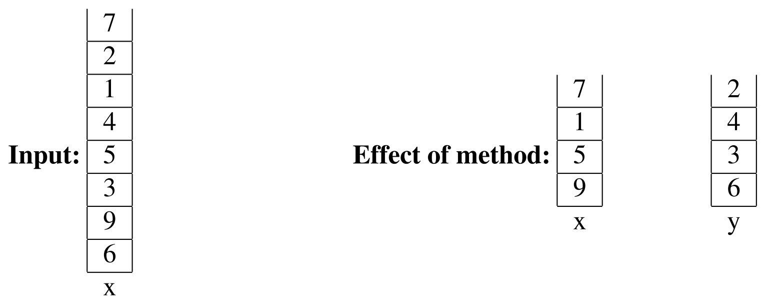
You are required to implement a method `public static ArrayStack decompose(ArrayStack x)` where `x` is a stack of `int`s. The method is required to decompose the values stored in the stack `x` into two groups:

- values in the odd positions remain in the stack `x`
- values in the even positions should be stored in a new stack, say `y`

The method should finally return the newly created stack `y` containing the values in the even positions. We will assume that the value on top of the stack is in position 1.

Make sure that the order of elements in both stacks will remain the same as the order of elements in the initial stack.

The following is a sample method run:



Only stacks can be used!.

Implement the method `decompose` **externally**, i.e. using the methods available in the stack class.

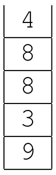
Exercise 3-3 Stack Sorting

Write an external method to sort a stack by putting the smaller elements towards the bottom.

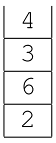
Exercise 3-4 Cube Game

For this problem you are going to implement a method for a simple game. In this game you are given a stack of cubes which can only be accessed from the top. You are required to determine whether the sum of the elements in the top half of the stack is equal to the sum of the elements in the bottom half. If the number of elements is odd, ignore the middle element. You have to end your check with the contents of the stack being in the same order as they were given to you. For the purpose of this problem implement the method `static boolean check(ArrayStack x)`, which performs the check operation described above over a stack `x` of integers and returns `true` if both halves are equal and `false` otherwise. **Note:** you are not allowed to use any data structure except stacks for solving this problem.

Sample run:

Input: 

Output: `true`, with $8 + 4 = 9 + 3 = 12$

Input: 

Output: `false`, with $4 + 3 \neq 6 + 2$

Exercise 3-5 Reverse a Stack

For this exercise, you are required to reverse the contents of a `Stack`. Use the `ArrayStack` implementation posted on the website.

a) First do it **internally**, i.e. as an instance method inside the `ArrayStack` class

b) Then do it **externally**, you should implement 3 **external** methods:

1. Write a method `static ArrayStack reverse1(ArrayStack s)` that takes a stack as a parameter and returns its reverse, you are allowed to destruct the original stack.
2. Write a method `static ArrayStack reverse2(ArrayStack s)` that takes a stack as a parameter and returns its reverse, but this time at the end the stack should be returned to the original state.
3. Write a method `static void reverse3(ArrayStack s)` that takes a stack as a parameter and reverses it and puts the reversed stack in the original.

You should display the stack elements before and after calling each method.

Exercise 3-6 Remove n^{th} Element

You are given a stack with the methods `pop, push, top, isEmpty, isFull and size`. You are required to implement a method `static void removeNth(ArrayStack s, int n)` which removes the n^{th} element from the bottom of the stack `s`. Assume that the value of n will be between 1 and size of the given stack, inclusive. For instance, the call `removeNth(s, 1)` should remove the lowermost item of the stack s.

The stack `s` should be unchanged except for the deleted element. You may only use stacks to solve this problem. Do not re-implement the stack operations.

Exercise 3-7 Postfix Evaluation

Write a class `PostfixEv.java` that allows for the evaluation of postfix expressions using a static method `int evaluate(String s)`. Your `evaluate` method will be passed a string representing a postfix expression containing space-separated elements ("2 3 4 + *",etc.) and is required to evaluate the expression and return the resulting value. The valid set of characters is {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, *, / }. Assume that that the syntax of the postfix expression is correct. Make use of a stack to solve this problem.

The postfix evaluation algorithm can be described as follows:

a) Initialise an empty stack.
b) Read the postfix expression from left to right.
c) If the character is an operand, push it onto the stack.
d) If the character is an operator, pop two operands, perform the appropriate operation, and then push the result onto the stack.
e) At the end of the postfix expression, pop the result from the stack.

Hint: You may use the built-in Java method `split` in the `String` class:
http://docs.oracle.com/javase/7/docs/api/java/lang/String.html

Exercise 3-8 Special Stack

We want to implement a data type `SpecialStack` that supports all the stack operations like `push()`, `pop()`, `isEmpty()`, `isFull()` and an additional operation `getMin()` which should return minimum element from the `SpecialStack`. All these operations of `SpecialStack` must be O(1). To implement `SpecialStack`, you should only use standard `Stack` data structure and no other data structure like arrays.

Stack ADT

```
public class Stack {

public Stack(int s);
public void push(int k);
public int pop();
public int size();
public int top();
public boolean isEmpty();
public boolean isFull();
}
```

Exercise 3-9 Midterm Exam 2015

Suppose we have a stack of integers. Initially the integers are not sorted. We would like to rearrange the stack using the insertion sort algorithm presented in lectures for arrays. The elements should be in ascending order where the smallest element will be on top of the stack and the largest at the bottom of the stack.

a) Write an external Java method `void stackSort(ObjectStack s)` that takes a stack of integers `s` and sorts the stack based on the insertion sort algorithm for arrays. **Please note that you are not allowed to use except stacks and the following constructor and methods:**

```
public ObjectStack(int maxSize)
public void push(Object o)
public Object pop()
public Object top()
public boolean isEmpty()
public boolean isFull()
public int size()
```

b) Given an initially unordered stack

1. What is the best case complexity of the algorithm? Justify your answer.
2. What is the worst case complexity of the algorithm? Justify your answer.
3. What is the invariant of the algorithm? Justify your answer.