Week 7 worksheet: Stacks, Elementary Search, Queues I

Total points: 10

Out: March 11 (Tuesday)

Due: March 14 (Friday end of day [2359 CDT according to D2L])

## What to submit?

Upload only one Word or PDF file to the designated D2L folder.

## Running time assumptions

* Assume that basic arithmetic operations (+, -, \*, /, %) run in constant time.
* Assume that the System.out.println( … ) statement runs in constant time.

## Generic Node <E> assumptions

* Assume that generic class E has an equals method.

# Exercise 1: Elementary Search

[3 pts] Consider the following sorted array. Suppose you want to search for the number 42 in this array using binary search. Indicate the portion of the array that is being searched in each of the following steps. Write L to indicate the lowest element of the search range, M to indicate the middle element (the one being probed) and H to indicate the highest element of the search range. I do the first one for you. You do steps 2, 3, and 4.

Step 1:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **5** | **8** | **13** | **21** | **34** | **55** | **89** | **144** | **233** | **377** | **610** | **987** |

L M H

Step 2:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **5** | **8** | **13** | **21** | **34** | **55** | **89** | **144** | **233** | **377** | **610** | **987** |

L M H

Step 3:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **5** | **8** | **13** | **21** | **34** | **55** | **89** | **144** | **233** | **377** | **610** | **987** |

L M H

Step 4:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **5** | **8** | **13** | **21** | **34** | **55** | **89** | **144** | **233** | **377** | **610** | **987** |

LMH

# Exercise 2: Implement countOccurrences in a Stack

[2 pts] Implement a **static** method such as you might put in a driver class, called countOccurrences, that takes two input parameters, a **Stack** called **s,** and an **Integer** value called **val**. The method returns as output an int value that represent how many times **val** appears in **s**. At the end of the method, the Stack must be unchanged (it’s OK to change the Stack while running the method, so long as you put it back at the end).

Here is a UML for the “pure” Stack class I want you to use. Don’t use the java.util.Stack<E> class with its extra methods or its ability to use linked list or array methods to simplify things.

|  |
| --- |
| **Stack<E>** |
| * top: Node<E> |
| * Stack<E>() * push(E): void * pop(): E * peek(): E * isEmpty(): boolean |

Assume that the signature of the method is

**public static int countOccurrences( Stack<Integer> s, Integer val ) {**

**Stack<Integer> tempStack = new Stack<>();**

**int count = 0;**

**// Count occurrences and transfer elements to tempStack**

**while (!s.isEmpty()) {**

**Integer current = s.pop();**

**if (current.equals(val)) {**

**count++;**

**}**

**tempStack.push(current);**

**}**

**// Restore the original stack**

**while (!tempStack.isEmpty()) {**

**s.push(tempStack.pop());**

**}**

**return count;**

**}**

# Exercise 3: Classical Stack algorithm #2.

[3 pts] Consider the following infix arithmetic expression: ( **( 8 + 3 ) \* 5 ) – 2 + 9**

Show how **Algorithm 2** (module-07-stack.ppt, slide #34) can be applied to evaluate the infix arithmetic expression. Use a table as shown in slide #36

|  |  |  |  |
| --- | --- | --- | --- |
| Input | Action | Number Stack (NS) | Operations Stack (OS) |
| ( | PUSH ( TO OS |  | ( |
| ( | PUSH ( TO OS |  | (,( |
| 8 | PUSH TO NS | 8 | (,( |
| + | PUSH + TO OS |  | (,(,+ |
| 3 | PUSH 3 TO NS | 8,3 | (,(,+ |
| ) | PUSH ) TO OS, POP (+),POP (8),POP (3)PUSH 8\*3 TO NS | 11 | ( |
| \* | PUSH \* TO OS | 11 | (,\* |
| 5 | PUSH 5 TO NS | 11,5 | (,\* |
| ) | PUSH ) TO OS, POP (\*),POP (11),POP (5)PUSH 11\*5 TO NS | 55 |  |
| - | PUSH – TO OS | 55 | - |
| 2 | PUSH 2 TO NS | 55,2 |  |
| + | PUSH + TO OS, POP (-),POP (55),POP (2)PUSH 55-2 TO NS | 53 | + |
| 9 | PUSH 9 TO NS | 53,9 |  |
| END | POP (+),POP (53),POP (9)PUSH 53-9 TO NS | 62 |  |

# Exercise 4: Queue implementation using a Linked List

[2 pts] Show the contents of the linked list at the beginning of each iteration of the for loop. I do the first one (j=0) for you. You do j = 1, 2, 3, and 4.

Queue<Integer> qq = new LinkedList<Integer>();

qq.add( 1 );

qq.add( 2 );

for ( int j = 0; j < 5; j++ ) { 0: 1: 2: 3: 4:

Integer a = qq.remove(); 1 3 5 11 41

Integer b = qq.remove(); 2 2 6 30 330

qq.add( a + b ); 3 5 11 41 371

qq.add( a \* b ); 2 6 30 330 13530

qq.add( a % b ); 1 1 5 11 41

}