CSP2348/CSP5243 Data Structures

**Solutions to Tutorial 09: Abstract Data Types (ADTs) (1): Stacks and Vector**

**Tasks:**

Complete the following.

**Task 1:** The following is a simplified algorithmthat tests whether a *phrase* is   
 well-bracketed

|  |
| --- |
| 1. Make *bracket-stack* empty. 2. For each symbol *sym* in phrase (scanning from left to right), repeat:    1. If *sym* is a left bracket:       1. Add *sym* to the top of *bracket-stack*.    2. If *sym* is a right bracket:       1. If *bracket-stack* is empty, terminate with answer false.       2. Remove a bracket from the top of *bracket-stack* into *left*.       3. if *left* and *sym* are not matched brackets, terminate with answer false. 3. Terminate with answer true if *bracket-stack* is empty, or false if otherwise |

Hand-test this algorithm with the following phrases:

(a). main(String[] args) {System.out.print (arg[0];}

(b). [(a + b) – (c – d)

****



****

**Task 2:** Test the push() and pop()methods of the Stack Class implemented   
 using Java Vector Class (Download the Java code WS0901 from   
 Blackboard).

1. Explain what will be resulted from each of the statements in the code;
2. Check with your explanation by running the Java Program.

**Running result:**

[**Brazil, Canada, France, Mexico, Russia, Sweden, Brazil, Turkey]**

**stack.size() = 8**

**stack.peek() = Turkey**

**stack.search("Brazil") = 2**

**stack.pop() = Turkey**

**stack.pop() = Brazil**

**[Brazil, Canada, France, Mexico, Russia, Sweden]**

**stack.size() = 6**

**stack.peek() = Sweden**

**stack.search("Brazil") = 6**

**Task 3:** Test the Vector class using WS0902 (download the Java code from the BB)

a. Note how to construct vector objects;

**private static Vector v = new Vector();**

**private static Vector w = new Vector();**

b. Observe the operation of some vector methods by analysing the executed   
 results.

**Running result:**

**v = []**

**v = [Perth, Sydney, Melbourne, Brisbane, Adelaide]**

**w = [Perth, Sydney, Melbourne, Brisbane, Adelaide]**

**w.equals(v) = true**

**v = [Perth, Sydney, Melbourne, Canberra, Adelaide]**

**w = [Perth, Sydney, Melbourne, Brisbane, Adelaide]**

**w.equals(v) = false**

**v = [Perth, Sydney, Melbourne, Hobart, Canberra, Adelaide]**

**w = [Perth, Sydney, Melbourne, Brisbane, Adelaide]**

**w.equals(v) = false**

**w = [Perth, Melbourne, Brisbane]**

**v = [Perth, Sydney, Melbourne, Hobart, Canberra, Perth, Melbourne, Brisbane, Adelaide]**

**v.indexOf("Perth") = 0**

**v.indexOf("Perth",2) = 5**

**v.indexOf("Canberra") = 4**