CSP2348/CSP5243 Data Structures

Solutions to Review Questions 09

Topics: Stacks and Vector

1. Would it make sense to call a stack a FILO (first-in-last-out) structure?

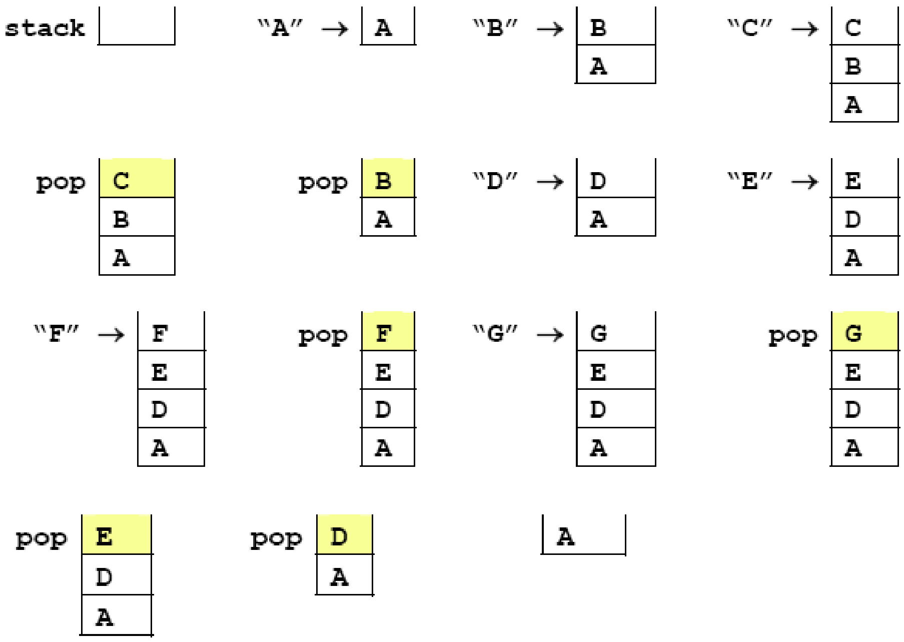
   Yes. FILO = LIFO

2. What is the difference between a SLL and a stack?

   It is possible to insert a node anywhere in a SLL and remove a node from anywhere in a SLL. Nodes in a stack may only be inserted at the top of the stack and removed from the top of the stack.

3. Trace the following code, showing the contents of the `stack` after each invocation [note: `push()= addLast(); pop() = removeLast()`]

```
Stack stack = new Stack();
stack.push(“A”);
stack.push(“B”);
stack.push(“C”);
stack.pop();
stack.pop();
stack.push(“D”);
stack.push(“E”);
stack.push(“F”);
stack.pop();
stack.push(“G”);
stack.pop();
stack.pop();
stack.pop();
```



4. Solution:

Array is a static data structure. When its capacity is reached, a new and larger array must be created (using the method `addLast()`) to host the existing values transferred from the 'old' array, and then to add any new value into the new array. SLL is a dynamic data structure. Its front end is used as the 'open end' to add and delete the top value whenever it is needed in a stack.

For `ArrayStack`, assume the current array has *n* components. If its capacity is not reached, it takes O(1) time to add a new value into the array (as its new last element) when `addLast()` is invoked. If its capacity is reached, `expand()` is invoked which performs *n* copies to transfer the *n* values from the current/old array into a new array of capacity of 2*n*, and then `addLast()` takes O(1) time to add a new value into the new array (as its new last element). Therefore, in this case, without considering the effort of creating a new array, `addLast()` has a time complexity of O(n) (i.e., O(n) + O(1) = O(n)).

For `LinkedStack`, its capacity is dynamic, and `addLast()` performs like inserting a new value into the front end of a SSL (i.e., as its new first node). Therefore, it has a time complexity of O(1) in all circumstances.