1. Implement a function with signature transfer(S,T) that transfers all elements from Stack S onto Stack T, so that that elements that starts at the top of S is the first to be inserted into T, and element at the bottom of S ends up at the top of T.
2. Implement a function that reverses a list of elements by pushing them onto a stack in one order, and write them back to the list in reversed order.
3. Modify ArrayStack implementation so that the stack’s capacity is limited to maxlen elements. If push is called when the stack is at full capacity, throw a Full exception.
4. Implement a transfer function and two temporary stacks, to replace the contents of a given stack S with those same elements, but in reverse order.
5. HTML generally allows optional attributes to be expressed as part of an opening tag. The general form used is <name attribute1 = “value1” attribute2 = “value2”>. Write function that checks whether tags or matched properly, even when an opening tag may include one or more attributes.
6. Suppose you have three nonempty stacks R, S and T. Describe a sequence of operations that results in S storing all elements originally in T below of S’s original elements, with both sets of those elements in their original configuration. For example, if R = [1,2,3], S = [4, 5] and T = [6, 7, 8, 9], the final configuration should have R = [1, 2, 3] and S = [6, 7, 8, 9, 4, 5].
7. Design a stack using a single queue as an instance variable, and only constant additional local memory within the method bodies.
8. Design a queue using two stacks as instance variables, such that all queue operations execute in amortized O(1) time.
9. Design a double-ended queue using two stacks as instance variables.
10. Suppose you have a stack S containing n elements and a queue Q that is initially empty. Write function that use Q to scan S to see if it contains a certain element x, with the additional constraint that your algorithm must return the elements back to S in their original order. You may use S, Q and a constant number of other variables.
11. In certain applications of the queue, it is common to repeatedly dequeue an element, process it in some way, and then immediately enqueuer the same element. Modify ArrayQueue implementation to include a rotate( ) method that has semantics identical to the combination, Q.enqueue (Q.dequeue()). However, your implementation should be more efficient than making two separate calls.
12. Give an array based implementation of double-ended queue supporting behaviors like len (), add\_first(), add\_last(), delete\_first(), delete\_last(), first() and last(). Double-ended queue should be of fixed capacity. When queue is full, inserting element from one end should cause an element to be lost from the opposite side.
13. Implement LeakyStack. This stack should be of fixed size. When push is invoked with the stack at full capacity, rather than throwing a Full exception, accept the pushed element at the top while “leaking” the oldest element from the bottom of the stack to make a room.
14. When a share of common stock of some company is sold, the capital gain (or, sometimes, loss) is the difference between the share’s selling price and the price originally paid to buy it. This rule is easy to understand for a single share, but if we sell multiple shares of stock bought over a long period of time, then we must identify the shares actually being sold. A standard accounting principle for identifying which share of stock were sold in such a case to use a FIFO protocol – the shares sold are the ones that have been held the longest period. For example, suppose we buy 100 shares at Rs 20 each on day 1, 20 shares at Rs 24 on day 2, 200 shares at Rs 36 on day3. And sell 150 shares on day 4 at Rs 30 each. Then applying FIFO principle means that of the 150 shares sold, 100 were bought on day1, 20 were bought on day2, and 30 were bought on day3. The capital gain in this case would therefore be 100\*10 + 20\*6 + 30 \* (-6), or Rs 940. Write a program that takes as input a sequence of transactions of the form “buy x share(s) at Rs y each” or “sell x share(s) at Rs y each” , assuming that transactions occur on consecutive days and the values x and y are integers. Given this input sequence output should be the total capital gain (loss) for the entire sequence, using the FIFO protocol to identify shares.
15. Design an ADT for a two-color, double-stack ADT that consists of two stacks – one “red” and one “blue” – and has as its operations color-coded versions of the regular stack ADT operations. For example, this ADT should support both a red push operation and a blue push operation. Give an efficient implementation of this ADT using single array whose capacity is set at some value N that is assumed to always be larger than the sizes of the red and blue stacks combined.