Name: Fatima Manzar

1. The PrefixStack class utilizes standard stack operations (‘push’, ‘pop’, ‘peek’ and ‘isEmpty’). The ‘main’ method demonstrates creating an instance of PrefixStack and performing various operations on it; elements are added by pushing them onto the stack before their values can be retrieved using peek and pop operations.

The demonstration shows how to verify whether a stack is completely empty after moving and extracting elements from it.

1. There are two stack implementations: ArrayDeque (for array-based stacks) and LinkedList (for linked list-based piles). Our arrayBasedStackSpeedTest and linkedListBasedStackSpeedTest methods conduct millions of push and pop operations on each of these types of stacks before performing our tests.

Measured using System.currentTimeMillis(), these operations' times can be measured to understand their relative speeds.

1. The listElements function accepts a stack of elements as input and prints them one by one. The main method shows how to create a stack of elements by pushing them onto the stack and listing these elements with listElements. As we remove the elements from the stack, reverse the order in which they are output.
2. The algorithm uses dynamic programming to calculate the lengths for LIS at every index of an array. After these lengths are calculated, they are reconstructed using a stack and compared to previous elements round-robin style. The algorithm iterates through each element, until they are all successfully processed.
3. The input sentence is split into an array of words by using space characters to delimit it. These words are then stored on a stack for retrieval later by an algorithm that pops the words from the stack, appends them onto a StringBuilder and inverts their order.

The output string will have all the words in our sentence reversed.

1. The algorithm employs a stack to keep track of current pathways through a maze, beginning from its entrance and exploring possible exit routes by pushing and popping locations onto and from it. Its goal is to locate one between its entrance and exit points.

An algorithm explores a maze until it either finds an exit or determines there is no valid path through. The exact nature and details of its maze-solving logic depend on its structure and rules.