DAY 12:

Task 4:

Q) Stack Sorting In-Place

You must write a function to sort a stack such that the smallest items are on the top. You can use an additional temporary stack, but you may not copy the elements into any other data structure such as an array. The stack supports the following operations: push, pop, peek, and is Empty.

EXPLANATION:

Overview

The goal is to sort a stack such that the smallest elements are on the top. This is achieved using an auxiliary stack (temporary stack) to help with the sorting process.

Steps Explained

1. Initialization:

- Start with the original stack that needs to be sorted.
- Create an empty temporary stack that will be used to hold elements in a sorted order.

2. Main Sorting Loop:

- Continue the process until the original stack is empty.
- Pop an element from the original stack (let’s call this element current).
- Compare current with the elements in the temporary stack.

3. Placing Elements in Sorted Order:

- If the temporary stack is empty or the top element of the temporary stack is less than or equal to current, push current onto the temporary stack.
- If the top element of the temporary stack is greater than current, pop elements from the temporary stack and push them back onto the original stack until you find the correct position for current.

- Push current onto the temporary stack.

4. Transferring Back to Original Stack:

- After the original stack is empty, the temporary stack contains all elements in sorted order, but in reverse (smallest at the bottom).

- Pop all elements from the temporary stack and push them back onto the original stack. This will result in the original stack being sorted in ascending order with the smallest elements at the top.

Example Workflow

Consider a stack with elements [34, 3, 31, 98, 92, 23]:

1. First Pass:

- Pop 23 from the original stack and push it onto the temporary stack.

- Pop 92 from the original stack and push it onto the temporary stack.

- Pop 98 from the original stack and push it onto the temporary stack.

- Pop 31 from the original stack. Since 31 is less than 98, 92, and 23 (elements in the temporary stack), move 98, 92, and 23 back to the original stack in that order.

- Push 31 onto the temporary stack, then move back 23, 92, and 98 onto the temporary stack in sorted order.

- Continue this process until all elements are moved in sorted order.

2. Subsequent Passes:

- Continue this process of popping from the original stack, comparing with the temporary stack, and placing in the correct position until the original stack is empty.

3. Final Transfer:

- Transfer all elements from the temporary stack back to the original stack to maintain the sorted order.

Result

At the end of this process, the original stack will be sorted with the smallest element at the top.

Conclusion

The algorithm effectively uses a secondary stack to sort the elements of the original stack by leveraging the properties of stack operations (push and pop). By carefully transferring element between the two stacks, it ensures that the elements are placed in the correct order, resulting in a sorted stack.