Unique Karanjit

Department of Computer Science, University of the Cumberlands

MSCS-532-A01: Algorithms and Data Structure

Dr. Satish Penmatsa

Feb 9, 2025

Table of Contents

Implementation of Data Structures **3**

Performance Analysis **4**

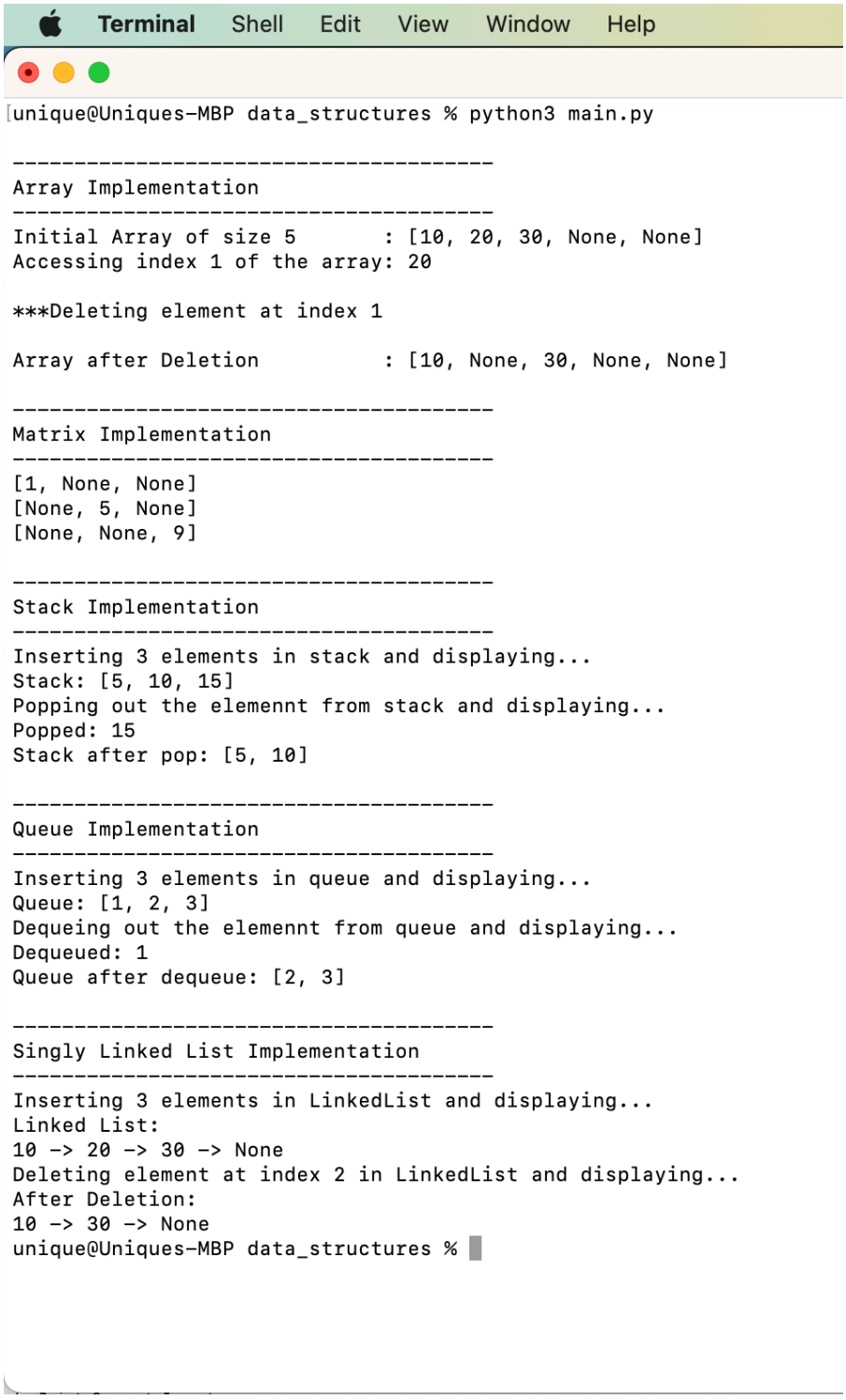
Practical Applications **4**

- Arrays: 4
- Stacks: 4
- Queues: 4
- Linked Lists: 5

Why Choose One Over Another? 5

Implementation of Data Structures

Arrays, Stacks, Queues, Matrices, and Linked List are implemented in this project and the output of the running code is shown below.



Performance Analysis

The time complexity for operation on all data structures can be shown in the table as shown below.

Operation	Array	Stack	Queue	Singly Linked List
Insert – End	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Insert – Middle	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Delete – End	$O(1)$	$O(1)$	$O(n)$	$O(n)$
Delete – Middle	$O(n)$	$O(n)$	$O(n)$	$O(n)$
Access	$O(1)$	$O(1)$	$O(1)$	$O(n)$

Trade Off-s Between Data Structures

- **Memory Usage**: Arrays use contiguous memory, while linked lists use extra memory for pointers.
- **Speed:** Arrays provide O(1) access but O(n) insert/delete (except at the end). Linked lists offer efficient insertions/deletions but slow lookups.
- **Use Cases:**
    - **Array-based Stacks/Queues:** Good when memory is predictable.
    - **Linked List-based Stacks/Queues:** Preferred when dynamic resizing is needed.

Practical Applications

• Arrays:

- Used in databases for storing records.
- Used in graphics processing where fast access is required.
- Used in scheduling problems and caching.

• Stacks:

- The function call stack in programming languages.
- Undo/redo functionality in text editors.
- Expression evaluation (e.g., parsing mathematical expressions).

• Queues:

- CPU scheduling and process scheduling.
- Printer queue management.
- Breadth-first search (BFS) in graphs.

• Linked Lists:

- Used in dynamic memory allocation.
- Implementation of hash tables (chaining for collision resolution).
- Navigation systems where elements need to be inserted/deleted dynamically.

Why Choose One Over Another?

- **When fast access is needed:** Array
- **When dynamic resizing is required:** a linked list
- **When LIFO structure is needed:** stack
- **When FIFO structure is needed:** queue