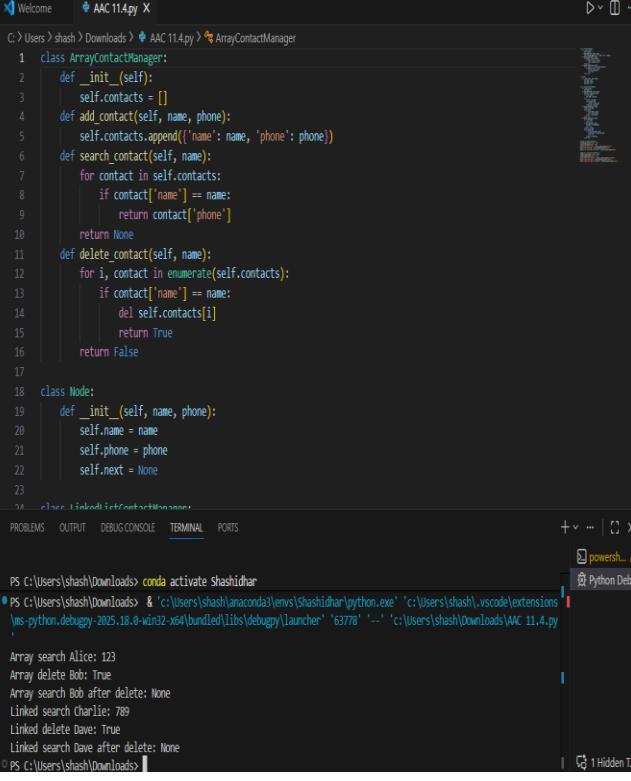


Name:I. Abhinay H.No:2303A51811 Batch:26

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	
Course Coordinator Name		Dr. Rishabh Mittal	
Instructor(s) Name			
		Mr. S Naresh Kumar	
		Ms. B. Swathi	
		Dr. Sasanko Shekhar Gantayat	
		Mr. Md Sallauddin	
		Dr. Mathivanan	
		Mr. Y Srikanth	
		Ms. N Shilpa	
		Dr. Rishabh Mittal (Coordinator)	
		Dr. R. Prashant Kumar	
		Mr. Ankushavali MD	
		Mr. B Viswanath	
		Ms. Sujitha Reddy	
		Ms. A. Anitha	
		Ms. M. Madhuri	
		Ms. Katherashala Swetha	
		Ms. Velpula sumalatha	
		Mr. Bingi Raju	
Course Code	23CS002PC304	Course Title	AI Assisted Coding
Year/Sem	III/II	Regulation	R23
Date and Day of Assignment	Week6 – Wednesday	Time(s)	23CSBTB01 To 23CSBTB52
Duration	2 Hours	Applicable to Batches	All batches
Assignment Number: 11.3(Present assignment number)/ 24 (Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	Lab 11: Data Structures with AI Implementing Fundamental Data Structures using AI Assistance Lab Objectives:	Week6 - Wednesday

	<p>By the end of this lab, students will be able to:</p> <ul style="list-style-type: none"> • Design and implement fundamental data structures in Python using AI assistance. • Effectively prompt AI tools (e.g., GitHub Copilot) for code generation, optimization, and documentation. • Understand and compare core data structures: Arrays, Linked Lists, Stacks, Queues, Priority Queues, Trees, and Graphs. • Improve code readability, efficiency, and maintainability using AI-generated suggestions. <p>Learning Outcomes</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • Apply appropriate data structures to solve real-world problems. • Analyze time and space complexity of different data structure operations. • Use AI tools responsibly to assist (not replace) logical thinking and problem-solving. • Validate, test, and refine AI-generated code. 	
	<p>Task 1: Smart Contact Manager (Arrays & Linked Lists)</p> <p>Scenario</p> <p>SR University's student club requires a simple Contact Manager Application to store members' names and phone numbers. The system should support efficient addition, searching, and deletion of contacts.</p> <p>Tasks</p> <ol style="list-style-type: none"> 1. Implement the contact manager using arrays (lists). 2. Implement the same functionality using a linked list for dynamic memory allocation. 3. Implement the following operations in both approaches: <ul style="list-style-type: none"> ○ Add a contact ○ Search for a contact ○ Delete a contact 4. Use GitHub Copilot to assist in generating search and delete methods. 5. Compare array vs. linked list approaches with respect to: <ul style="list-style-type: none"> ○ Insertion efficiency ○ Deletion efficiency <p>Expected Outcome</p> <ul style="list-style-type: none"> • Two working implementations (array-based and linked-list-based). • A brief comparison explaining performance differences. 	



```

1 class ArrayContactManager:
2     def __init__(self):
3         self.contacts = []
4     def add_contact(self, name, phone):
5         self.contacts.append({'name': name, 'phone': phone})
6     def search_contact(self, name):
7         for contact in self.contacts:
8             if contact['name'] == name:
9                 return contact['phone']
10        return None
11    def delete_contact(self, name):
12        for i, contact in enumerate(self.contacts):
13            if contact['name'] == name:
14                del self.contacts[i]
15                return True
16        return False
17
18 class Node:
19     def __init__(self, name, phone):
20         self.name = name
21         self.phone = phone
22         self.next = None
23
24 class LinkedListContactManager:
25     def __init__(self):
26         self.head = None
27     def add_contact(self, name, phone):
28         new_node = Node(name, phone)
29         if not self.head:
30             self.head = new_node
31         else:
32             current = self.head
33             while current.next:
34                 current = current.next
35             current.next = new_node
36     def search_contact(self, name):
37         current = self.head
38         while current:
39             if current.name == name:
40                 return current.phone
41             current = current.next

```

CHAT

CONTACT MANAGER IMPLEMENTATION WITH ARRAYS AND LINKED LL...

- o Add a contact
- o Search for a contact
- o Delete a contact
- 4. Use GitHub Copilot to assist in generating search and delete methods.
- 5. Compare array vs. linked list approaches with respect to:
 - o Insertion efficiency
 - o Deletion efficiency
 - o Expected Outcome
- Two working implementations (array-based and linked-list-based).
- A brief comparison explaining performance differences.
- give code in python without comments

✓ pwd

✓ Created array_contact_manager.py

✓ Created linked_list_contact_manager.py

✓ Created comparison.txt

I've created three files in your current directory (shash) with the implementations:

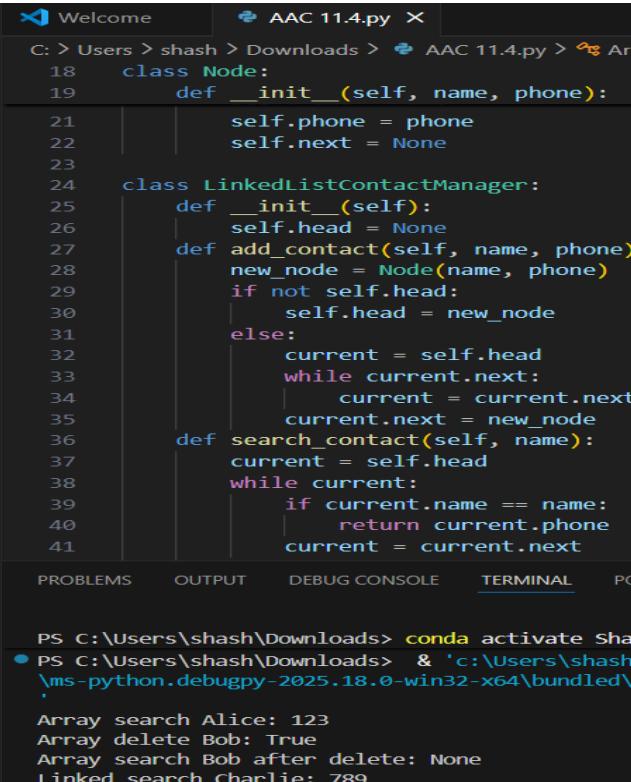
- array_contact_manager.py: Array-based contact manager
- linked_list_contact_manager.py: Linked list-based contact manager
- comparison.txt: Brief performance comparison

The code is provided without comments as requested. You can run these files directly in Python to test the functionality. For example, you could create a simple test script to instantiate the classes and call the methods.

AAC 11.4.py

Describe what to build next

Grok Code Fast 1



```

18 class Node:
19     def __init__(self, name, phone):
20         self.phone = phone
21         self.next = None
22
23 class LinkedListContactManager:
24     def __init__(self):
25         self.head = None
26     def add_contact(self, name, phone):
27         new_node = Node(name, phone)
28         if not self.head:
29             self.head = new_node
30         else:
31             current = self.head
32             while current.next:
33                 current = current.next
34             current.next = new_node
35     def search_contact(self, name):
36         current = self.head
37         while current:
38             if current.name == name:
39                 return current.phone
40             current = current.next

```

TERMINAL

```

PS C:\Users\shash\Downloads> conda activate Shashidhar
PS C:\Users\shash\Downloads> & 'c:\Users\shash\anaconda3\envs\Shashidhar\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' .
Array search Alice: 123
Array delete Bob: True
Array search Bob after delete: None
Linked search Charlie: 789
Linked delete Dave: True
Linked search Dave after delete: None

```

```
C:\> Users > shash > Downloads > AAC 11.4.py > ArrayContactManager
24     class LinkedListContactManager:
36         def search_contact(self, name):
40             return current.phone
41             current = current.next
42         return None
43     def delete_contact(self, name):
44         if not self.head:
45             return False
46         if self.head.name == name:
47             self.head = self.head.next
48             return True
49         current = self.head
50         while current.next:
51             if current.next.name == name:
52                 current.next = current.next.next
53                 return True
54             current = current.next
55         return False
56
57 array_cm = ArrayContactManager()
58 array_cm.add_contact("Alice", "123")
59 array_cm.add_contact("Bob", "456")
60 print("Array search Alice:", array_cm.search_contact("Alice"))

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
```

```
PS C:\Users\shash\Downloads> conda activate Shashidhar
● PS C:\Users\shash\Downloads> & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63778' '--' 'c:\Users\shash\Downloads\AAC 11.4.py'
Array search Alice: 123
Array delete Bob: True
Array search Bob after delete: None
Linked search Charlie: 789
Linked delete Dave: True
Linked search Dave after delete: None
○ PS C:\Users\shash\Downloads>
```

	<p>The screenshot shows the VS Code interface with the following details:</p> <ul style="list-style-type: none"> File Explorer: Shows the file structure: Welcome, AAC 11.4.py (selected), and ArrayContactManager. Code Editor: Displays the code for two contact manager classes: <code>LinkedListContactManager</code> and <code>ArrayContactManager</code>. The code includes methods for adding contacts, searching, and deleting them. Terminal: Shows the command line output of running the code. It activates the environment, runs the script, and prints the results of the search and delete operations for both array and linked list implementations. 	
	<p>Task 2: Library Book Search System (Queues & Priority Queues)</p> <p>Scenario</p> <p>The SRU Library manages book borrow requests. Students and faculty submit requests, but faculty requests must be prioritized over student requests.</p> <p>Tasks</p> <ol style="list-style-type: none"> 1. Implement a Queue (FIFO) to manage book requests. 2. Extend the system to a Priority Queue, prioritizing faculty requests. 3. Use GitHub Copilot to assist in generating: <ul style="list-style-type: none"> o enqueue() method o dequeue() method 4. Test the system with a mix of student and faculty requests. <p>Expected Outcome</p> <ul style="list-style-type: none"> • Working queue and priority queue implementations. • Correct prioritization of faculty requests. 	

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

The SRU library manages book borrow requests. Students and faculty submit requests, but faculty requests must be prioritized over student requests.

Tasks

1. Implement a Queue (FIFO) to manage book requests.
2. Extend the system to a Priority Queue, prioritizing faculty requests.
3. Use GitHub Copilot to assist in generating:
 - o enqueue() method
 - o dequeue() method
4. Test the system with a mix of student and faculty requests.

Expected Outcome

- Working queue and priority queue implementations.
- Correct prioritization of faculty requests.

give code in python without comments just paste in the file

✓ Optimizing tool selection...

✓ Reviewed and created AAC 114.py

✓ Created AAC 114.py

✓ Configured a Python environment

✓ Ran `pylanceRunCodeSnippet - pylance mcp server (MCP Server)`

✓ Reviewed code and provided a brief summary

CHAT

Describe what to build next:

Claude Sonnet 4.6

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

Code-written to AAC 114.py([C:\Users\shash\Downloads\AAC 114.py]) and verified working. Here's what the output shows:

Regular Queue (FIFO): Processes in insertion order — Alice, Dr. Smith, Bob, Dr. Lee, Carol.

Priority Queue: Automatically reorders so faculty always come first — Dr. Smith, Dr. Lee (faculty), then Alice, Bob, Carol (students). Within the same role, arrival order (FIFO) is preserved.

Key classes:

- BookRequest — holds requester name, book title, and role
- LibraryQueue — FIFO using `collections.deque` with `enqueue()`/`dequeue()`
- PriorityLibraryQueue — min-heap via `heapq` assigning priority 0 to faculty and 1 to students

CHAT

Describe what to build next:

Claude Sonnet 4.6 + ta

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

PS C:\Users\shash\Downloads> cd 'C:\Users\shash\Downloads'; & 'C:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'C:\Users\shash\vscodeextensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '59531' '<' 'C:\Users\shash\Downloads\AAC 114.py'

```

C:\>users>shash>Downloads> AAC 114.py >_
 1 from collections import deque
 2 import heapq
 3 class BookRequest:
 4     def __init__(self, requester_name, book_title, role):
 5         self.requester_name = requester_name
 6         self.book_title = book_title
 7         self.role = role
 8     def __repr__(self):
 9         return f'{self.role.upper()} | {self.requester_name} -> {self.book_title}'
10 class LibraryQueue:
11     def __init__(self):
12         self._queue = deque()
13     def enqueue(self, request):
14         self._queue.append(request)
15         print(f'[QUEUE] Added: {request}')
16     def dequeue(self):
17         if not self._queue:
18             print('[QUEUE] No requests in queue.')
19             return None
20         request = self._queue.popleft()
21         print(f'[QUEUE] Processing: {request}')
22         return request
23     def is_empty(self):
24         return len(self._queue) == 0
25     def display(self):
26         if not self._queue:
27             print('[QUEUE] Queue is empty.')
28         else:
29             print('[QUEUE] Current requests:')
30             for req in self._queue:
31                 print(f'  {req}')
32 class PriorityLibraryQueue:
33     PRIORITY_MAP = {"faculty": 0, "student": 1}
34     def __init__(self):
35         self._queue = []
36         self._heap = []
37         self._counter = 0
38     def enqueue(self, request):
39         priority = self.PRIORITY_MAP.get(request.role.lower(), 2)
40         heapq.heappush(self._heap, (priority, self._counter, request))
41         self._counter += 1
42         print(f'[PRIORITY QUEUE] Added: {request}')
43     def dequeue(self):
44         if not self._heap:
45             ...

```

--- Dequeuing all requests ---

[PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'

[PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'

[PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'

[PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'

[PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'

PS C:\Users\shash\Downloads>

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

PS C:\Users\shash\Downloads> cd 'C:\Users\shash\Downloads'; & 'C:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'C:\Users\shash\vscodeextensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '59531' '<' 'C:\Users\shash\Downloads\AAC 114.py'

```

C:\>users>shash>Downloads> AAC 114.py >_
10 class LibraryQueue:
11     def is_empty(self):
12         return len(self._queue) == 0
13     def display(self):
14         if not self._queue:
15             print('[QUEUE] Queue is empty.')
16         else:
17             print('[QUEUE] Current requests:')
18             for req in self._queue:
19                 print(f'  {req}')
20     def __init__(self):
21         self._queue = []
22         self._heap = []
23         self._counter = 0
24     def enqueue(self, request):
25         priority = self.PRIORITY_MAP.get(request.role.lower(), 2)
26         heapq.heappush(self._heap, (priority, self._counter, request))
27         self._counter += 1
28         print(f'[PRIORITY QUEUE] Added: {request}')
29     def dequeue(self):
30         if not self._queue:
31             ...

```

--- Dequeuing all requests ---

[PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'

[PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'

[PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'

[PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'

[PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'

PS C:\Users\shash\Downloads>

The screenshot shows a Visual Studio Code interface with a dark theme. The top navigation bar includes 'Welcome' and 'AAC 11.4.py'. The code editor displays a Python script named 'AAC 11.4.py' which defines a class 'PriorityLibraryQueue' with methods for dequeuing, checking if empty, and displaying requests. It also includes a test section with two enqueue operations and a loop to process all requests. Below the code editor is a terminal window showing the command to run the script and its output, which lists five requests being processed in FIFO order.

```
C: > Users > shash > Downloads > AAC 11.4.py > ...
32     class PriorityLibraryQueue:
33         def dequeue(self):
34             if len(self._heap) == 0:
35                 print("[PRIORITY QUEUE] No requests in queue.")
36                 return None
37             _, _, request = heapq.heappop(self._heap)
38             print(f"[PRIORITY QUEUE] Processing: {request}")
39             return request
40         def is_empty(self):
41             return len(self._heap) == 0
42         def display(self):
43             if not self._heap:
44                 print("[PRIORITY QUEUE] Queue is empty.")
45             else:
46                 sorted_requests = sorted(self._heap, key=lambda x: (x[0], x[1]))
47                 print("[PRIORITY QUEUE] Current requests (by priority):")
48                 for _, _, req in sorted_requests:
49                     print(f" {req}")
50             print("-" * 55)
51         print("REGULAR QUEUE (FIFO) TEST")
52         print("-" * 55)
53         library_queue = LibraryQueue()
54         library_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
55         library_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))
56
57         while not library_queue.is_empty():
58             library_queue.dequeue()
59
56
57         print("-" * 55)
58         print("Dequeueing all requests")
59         for _ in range(5):
60             print(library_queue.dequeue())
61
62         print("-" * 55)
63
64         library_queue.display()
65
66
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

```
PS C:\Users\shash\Downloads> c:; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\laur
: \Users\shash\Downloads\AAC 11.4.py'
```

```
--- Dequeueing all requests ---
[PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
[PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
[PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
[PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
[PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
```

```
C:\> Users > shash > Downloads > AAC 11.4.py > ...
65     library_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))
66     library_queue.enqueue(BookRequest("Bob", "Operating Systems", "student"))
67     library_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty"))
68     library_queue.enqueue(BookRequest("Carol", "Computer Networks", "student"))
69     print()
70     library_queue.display()
71     print()
72     print("--- Dequeuing all requests ---")
73     while not library_queue.is_empty():
74         library_queue.dequeue()
75     print()
76     print("=" * 55)
77     print("      PRIORITY QUEUE TEST (Faculty First)")
78     print("=" * 55)
79     priority_queue = PriorityLibraryQueue()
80     priority_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
81     priority_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))
82     priority_queue.enqueue(BookRequest("Bob", "Operating Systems", "student"))
83     priority_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty"))
84     priority_queue.enqueue(BookRequest("Carol", "Computer Networks", "student"))
85     print()
86     priority_queue.display()
87     print()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\shash\Downloads> c:> cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\<br>'<br>`c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy<br>:\Users\shash\Downloads\AAC 11.4.py'
```

```
--- Dequeuing all requests ---
[PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
[PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
[PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
[PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
[PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
```

The screenshot shows a Microsoft Visual Studio Code interface. The top part displays a Python script named `AAC 11.4.py` with code for a priority queue. The bottom part shows the terminal window with the script's execution and its output.

```

Welcome AAC 11.4.py
C: > Users > shash > Downloads > AAC 11.4.py > ...
76 print("=" * 55)
77 print("      PRIORITY QUEUE TEST (Faculty First)")
78 print("=" * 55)
79 priority_queue = PriorityLibraryQueue()
80 priority_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
81 priority_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))
82 priority_queue.enqueue(BookRequest("Bob", "Operating Systems", "student"))
83 priority_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty"))
84 priority_queue.enqueue(BookRequest("Carol", "Computer Networks", "student"))
85 print()
86 priority_queue.display()
87 print()
88 print("--- Dequeuing all requests ---")
89 while not priority_queue.is_empty():
90     priority_queue.dequeue()
91

```

TERMINAL

```

PS C:\Users\shash\Downloads> c:; cd 'c:\users\shash\Downloads'; & 'c:\users\shash\anaconda3\envs\py37\python' 'c:\users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\client.py' -d :56757 :\\Users\\shash\\Downloads\\AAC 11.4.py'
--- Dequeuing all requests ---
[PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
[PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
[PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
[PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
[PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
○ PS C:\Users\shash\Downloads>

```

Task 3: Emergency Help Desk (Stack Implementation)

Scenario

SR University's IT Help Desk receives technical support tickets from students and staff. While tickets are received sequentially, issue escalation follows a **Last-In, First-Out (LIFO)** approach.

Tasks

1. Implement a **Stack** to manage support tickets.
2. Provide the following operations:
 - o `push(ticket)`
 - o `pop()`
 - o `peek()`
3. Simulate at least **five tickets** being raised and resolved.
4. Use **GitHub Copilot** to suggest additional stack operations such as:
 - o Checking whether the stack is empty
 - o Checking whether the stack is full (if applicable)

Expected Outcome

- Functional stack-based ticket management system.
- Clear demonstration of LIFO behavior.

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

- o push(ticket)
- o pop()
- o peek()
- 3. Simulate at least five tickets being raised and resolved.
- 4. Use GitHub Copilot to suggest additional stack operations such as:
 - o Checking whether the stack is empty
 - o Checking whether the stack is full (if applicable)

Expected Outcome

- Functional stack-based ticket management system.
- Clear demonstration of LIFO behavior.

give code in python without comments

✓ Created AAC 11.5.py

✓ Created AAC 11.5.py

✓ Tried to run command but was blocked

✓ Ran pylancedeSimpler - pylance mcpc server (MCP Server)

✓ Summarized results for the user

Code saved to [AAC 11.5.py](c:/Users/shash/Downloads/AAC 11.5.py) and verified working. Here's what it demonstrates:

Classes:

- SupportTicket — holds ticket ID, requester, issue description, and priority
- HelpdeskStack — list-based stack with MAX_SIZE = 10

∅ + AAC 11.4.py

Describe what to build next

□ Claude Sonnet 46 □

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

- ✓ Tried to run command but was blocked
- ✓ Ran pylancedeSimpler - pylance mcpc server (MCP Server)
- ✓ Summarized results for the user

Code saved to [AAC 11.5.py](c:/Users/shash/Downloads/AAC 11.5.py) and verified working. Here's what it demonstrates:

Classes:

- SupportTicket — holds ticket ID, requester, issue description, and priority
- HelpdeskStack — list-based stack with MAX_SIZE = 10

Operations implemented:

Method	Description
push(ticket)	Adds ticket to top of stack
pop()	Removes and resolves ticket from top (LIFO)
peek()	Views top ticket without removing it
is_empty()	Returns True if no tickets
is_full()	Returns True if at MAX_SIZE
size()	Returns current ticket count
display()	Shows all tickets top-to-bottom

∅ + AAC 11.4.py

Describe what to build next

□ Claude Sonnet 46 □

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

- ✓ Summarized results for the user

Code saved to [AAC 11.5.py](c:/Users/shash/Downloads/AAC 11.5.py) and verified working. Here's what it demonstrates:

Classes:

- SupportTicket — holds ticket ID, requester, issue description, and priority
- HelpdeskStack — list-based stack with MAX_SIZE = 10

Operations implemented:

Method	Description
push(ticket)	Adds ticket to top of stack
pop()	Removes and resolves ticket from top (LIFO)
peek()	Views top ticket without removing it
is_empty()	Returns True if no tickets
is_full()	Returns True if at MAX_SIZE
size()	Returns current ticket count
display()	Shows all tickets top-to-bottom

LIFO behavior confirmed: tickets were pushed in order 101–105, and handled in reverse order 105–101. The final pop() on an empty stack is handled gracefully with a message instead of crashing.

∅ + AAC 11.4.py

Describe what to build next

□ Claude Sonnet 46 □

```
C:\> Users > shash > Downloads > AAC 11.4.py > ...
58
59     print("=" * 60)
60     print("      SR UNIVERSITY IT HELP DESK - TICKET SYSTEM")
61     print("=" * 60)
62
63     help_desk = HelpDeskStack()
64
65     print("\n--- Raising Support Tickets ---")
66     help_desk.push(SupportTicket(101, "Alice (Student)", "Cannot access Wi-Fi", "normal"))
67     help_desk.push(SupportTicket(102, "Bob (Staff)", "Projector not working in Lab3", "high"))
68     help_desk.push(SupportTicket(103, "Carol (Student)", "Email login failure", "normal"))
69     help_desk.push(SupportTicket(104, "Dr. Smith (Staff)", "VPN disconnecting repeatedly", "critical"))
70     help_desk.push(SupportTicket(105, "Eve (Student)", "Printer offline in Library", "normal"))
71
72     print()
73     help_desk.display()
74
75     print(f"\n[INFO] Stack empty? {help_desk.is_empty()}")
76     print(f"[INFO] Stack full? {help_desk.is_full()}")
77     print(f"[INFO] Total tickets in stack: {help_desk.size()}")
78
79     print("\n--- Peeking at top ticket ---")
80     help_desk.peek()
81

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\shash\Downloads> c:& cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51957' '--' ':\\Users\\shash\\Downloads\\AAC 11.4.py'
--- Peeking at top ticket ---
[PEEK] Next to resolve: [#105] Eve (Student) | Printer offline in Library (NORMAL)

--- Resolving Tickets (LIFO Order) ---
[RESOLVED] [#105] Eve (Student) | Printer offline in Library (NORMAL)
[RESOLVED] [#104] Dr. Smith (Staff) | VPN disconnecting repeatedly (CRITICAL)
[RESOLVED] [#103] Carol (Student) | Email login failure (NORMAL)
[RESOLVED] [#102] Bob (Staff) | Projector not working in Lab3 (HIGH)
```

```

C: > Users > shash > Downloads > AAC 11.4.py > ...
72     print()
73     help_desk.display()
74
75     print(f"\n[INFO] Stack empty? {help_desk.is_empty()}")
76     print(f"[INFO] Stack full? {help_desk.is_full()}")
77     print(f"[INFO] Total tickets in stack: {help_desk.size()}")
78
79     print("\n--- Peeking at top ticket ---")
80     help_desk.peek()
81
82     print("\n--- Resolving Tickets (LIFO Order) ---")
83     while not help_desk.is_empty():
84         help_desk.pop()
85
86     print()
87     print(f"[INFO] Stack empty after resolving all? {help_desk.is_empty()}")
88     help_desk.pop()
89

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS C:\Users\shash\Downloads> c;; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\an
'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib
:\Users\shash\Downloads\AAC 11.4.py'
[RESOLVED] [#104] Dr. Smith (Staff) | VPN disconnecting repeatedly (CRITICAL)
[RESOLVED] [#103] Carol (Student) | Email login failure (NORMAL)
[RESOLVED] [#102] Bob (Staff) | Projector not working in Lab3 (HIGH)
[RESOLVED] [#101] Alice (Student) | Cannot access Wi-Fi (NORMAL)

[INFO] Stack empty after resolving all? True
[STACK EMPTY] No tickets to resolve.
PS C:\Users\shash\Downloads>

```

Task 4: Hash Table

Objective

To implement a **Hash Table** and understand collision handling.

Task Description

Use AI to generate a hash table with:

- Insert
- Search
- Delete

Starter Code

```

class HashTable:
    pass

```

Expected Outcome

- Collision handling using chaining
- Well-commented methods

The screenshot shows three instances of the PyCharm code editor for the file 'AAC 11.4.py'. Each instance has a different configuration of toolbars and panels.

- Top Instance:** Shows the code with a standard toolbar and a 'CHAT' panel on the right containing a summary of the task and its requirements.
- Middle Instance:** Shows the code with a more complex toolbar and a 'CHAT' panel on the right.
- Bottom Instance:** Shows the code with a standard toolbar and a 'CHAT' panel on the right.

In all three instances, the code implements a HashTable class with methods for insertion, search, deletion, and display. The code uses a list of lists to handle collisions. The 'CHAT' panel in the middle instance provides a detailed summary of the task requirements and the code's implementation.

```
1 class HashTable:
2     def __init__(self, capacity=16):
3         self.capacity = capacity
4         self.size = 0
5         self.buckets = [{} for _ in range(self.capacity)]
6
7     def hash(self, key):
8         return hash(key) % self.capacity
9
10    def insert(self, key, value):
11        index = self._hash(key)
12        bucket = self.buckets[index]
13        for i, (k, v) in enumerate(bucket):
14            if k == key:
15                bucket[i] = (key, value)
16                return
17        bucket.append((key, value))
18        self.size += 1
19
20    def search(self, key):
21        index = self._hash(key)
22        bucket = self.buckets[index]
23        for k, v in bucket:
24            if k == key:
25                return v
26
27    def delete(self, key):
28        index = self._hash(key)
29        bucket = self.buckets[index]
30        for i, (k, v) in enumerate(bucket):
31            if k == key:
32                del bucket[i]
33                self.size -= 1
34                return True
35        return False
36
37    def load_factor(self):
38        return self.size / self.capacity
39
40    def display(self):
41        print(f"\n[HASH TABLE] capacity={self.capacity}, entries={self.size}, load_factor={self.load_factor()}")
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
```

```
C:\> Users > shash > Downloads > AAC 11.4.py > ...
58     ("bob", "bob@sru.edu"),
59     ("carol", "carol@sru.edu"),
60     ("dave", "dave@sru.edu"),
61     ("eve", "eve@sru.edu"),
62     ("frank", "frank@sru.edu"),
63     ("grace", "grace@sru.edu"),
64     ("heidi", "heidi@sru.edu"),
65     ("ivan", "ivan@sru.edu"),
66     ("judy", "judy@sru.edu"),
67 ]
68 for key, value in entries:
69     ht.insert(key, value)
70     print(f" [INSERT] key='{key}' -> '{value}'")
71
72 ht.display()
73
74 print("\n--- Searching entries ---")
75 for key in ["alice", "eve", "judy", "unknown"]:
76     result = ht.search(key)
77     if result:
78         print(f" [FOUND] key='{key}' -> '{result}'")
79     else:
80         print(f" [MISSING] key='{key}' not found")
81

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\shash\Downloads> c;; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\c:\Users\shash\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\del:\Users\shash\Downloads\AAC 11.4.py'

[HASH TABLE] capacity=7, entries=8, load_factor=1.14
Bucket[ 0]: (frank: frank@sru.edu)
Bucket[ 2]: (alice: alice.new@sru.edu) -> (carol: carol@sru.edu) -> (eve: eve@sru.edu)
Bucket[ 3]: (dave: dave@sru.edu) -> (heidi: heidi@sru.edu)
Bucket[ 4]: (judy: judy@sru.edu)
Bucket[ 6]: (ivan: ivan@sru.edu)
```

```

C:\> Users > shash > Downloads > AAC 11.4.py > ...
74     print(" Searching entries ...")
75     for key in ["alice", "eve", "judy", "unknown"]:
76         result = ht.search(key)
77         if result:
78             print(f" [FOUND] key='{key}' -> '{result}'")
79         else:
80             print(f" [MISSING] key='{key}' not found")
81
82     print("\n--- Updating an existing key ---")
83     ht.insert("alice", "alice.new@sru.edu")
84     print(f" [UPDATE] key='alice' -> '{ht.search('alice')}'")
85
86     print("\n--- Deleting entries ---")
87     for key in ["bob", "grace", "nonexistent"]:
88         success = ht.delete(key)
89         status = "DELETED" if success else "NOT FOUND"
90         print(f" [{status}] key='{key}'")
91
92     ht.display()
93

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```

PS C:\Users\shash\Downloads> c;; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\envs\py38\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\deb\pydevd.py'
: \Users\shash\Downloads\AAC 11.4.py'

[HASH TABLE] capacity=7, entries=8, load_factor=1.14
Bucket[ 0]: (frank: frank@sru.edu)
Bucket[ 2]: (alice: alice.new@sru.edu) -> (carol: carol@sru.edu) -> (eve: eve@sru.edu)
Bucket[ 3]: (dave: dave@sru.edu) -> (heidi: heidi@sru.edu)
Bucket[ 4]: (judy: judy@sru.edu)
Bucket[ 6]: (ivan: ivan@sru.edu)
PS C:\Users\shash\Downloads>

```

Task 5: Real-Time Application Challenge

Scenario

Design a **Campus Resource Management System** with the following features:

- Student Attendance Tracking
- Event Registration System
- Library Book Borrowing
- Bus Scheduling System
- Cafeteria Order Queue

Student Tasks

1. Choose the most appropriate data structure for each feature.
2. Justify your choice in **2–3 sentences**.
3. Implement **one selected feature** using AI-assisted code generation.

Expected Outcome

- Mapping table: Feature → Data Structure → Justification
- One fully working Python implementation

LIVE EDITOR

CHAT LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

Real-Time Application Challenge
Scenario
Design a Campus Resource Management System with the following features:
 • Student Attendance Tracking
 • Event Registration System
 • Library Book Borrowing
 • Bus Scheduling System
 • Cafeteria Order Queue
 Student Tasks
 1. Choose the most appropriate data structure for each feature.
 2. Justify your choice in 2-3 sentences.
 3. Implement one selected feature using AI-assisted code generation.
 Expected Outcome
 • Mapping table Feature -> Data Structure -> Justification
 • One fully working Python implementation
 give code in python without comments

✓ Created AAC 11.7.py
 ✓ Created AAC 11.7.py
 ✓ Tried to run command to verify

shrink the code

TERMINAL

```
PS C:\Users\shash\Downloads> c: cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50953' '--' 'c:\Users\shash\Downloads\AAC 11.7.py'
MAPPING: Feature -> Data Structure -> Justification
-- 1. Attendance (Dictionary) --
from collections import deque
import heapq
from datetime import datetime

print("MAPPING: Feature -> Data Structure -> Justification")
print("- * 60)
print("Attendance -> Dictionary -> O(1) lookup by student ID")
print("Event Reg -> Set+Dictionary -> Deduplication + fast access")
print("Book Borrow -> Queue (FIFO) -> Serve requests in order")
print("Bus Schedule -> Priority Queue -> Earliest departure first")
print("Cafeteria -> Queue (FIFO) -> Orders prepared in sequence")
print("\n-- 1. Attendance (Dictionary) --")
attendance = {}
for sid, name, status in [("S001", "Alice", "present"), ("S002", "Bob", "absent"), ("S003", "Carol", "present"), ("S004", "Dave", "present"), ("S005", "Eve", "present")]:
    attendance.setdefault(datetime(2026, 02, 19), {})[sid] = (name, status)
    print(f" {sid} {name} -> {status}")
print("Reports: ", {sid: s for sid, (n, s) in attendance["2026-02-19"].items()})
print("\n-- 2. Event Registration (Set + Dict) --")
events = {"Tech Fest 2026": {"attendees": set(), "capacity": 5}}
for sid in ["S001", "S002", "S003", "S002", "S004", "S005", "S006"]:
    e = events["Tech Fest 2026"]
    if sid in e["attendees"]:
        print(f" DUPLICATE: {sid}")
    elif len(e["attendees"]) > e["capacity"]:
        print(f" FULL: {sid} rejected")
    else:
        e["attendees"].add(sid)
        print(f" Registered: {sid}")
print("Attendees:", sorted(events["Tech Fest 2026"]["attendees"]))

print("\n-- 3. Library Borrow (Queue) --")
bq = deque([( "S001", "Data Structures"), ( "S003", "Operating Systems"), ( "S005", "Computer Networks"), ( "S002", "Data Structures"), ( "S004", "Algorithms")])
while bq:
    sid, book = bq.popleft()
    print(f" Issued '{book}' to {sid}")

print("\n-- 4. Bus Scheduling (Priority Queue) --")
buses, ctr = [], 0
for route, t in [(("City", "07:30"), ("Airport", "09:00"), ("Mall", "08:15"), ("Station", "07:00"), ("Hospital", "10:30"))]:
    heapq.heappush(buses, (datetime.strptime(t, "%H:%M"), ctr, route))
    ctr += 1
while buses:
    dt, r = heapq.heappop(buses)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
PS C:\Users\shash\Downloads> c: cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50953' '--' 'c:\Users\shash\Downloads\AAC 11.7.py'
MAPPING: Feature -> Data Structure -> Justification
-- 5. Cafeteria Orders (Queue) --
Order #1 ready: Burger for Alice
Order #2 ready: Pasta for Bob
Order #3 ready: Sandwich for Carol
Order #4 ready: Rice for Dave
Order #5 ready: Noodles for Eve
```

TERMINAL

```
PS C:\Users\shash\Downloads> c: cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\ShashiDhar\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50953' '--' 'c:\Users\shash\Downloads\AAC 11.7.py'
MAPPING: Feature -> Data Structure -> Justification
-- 5. Cafeteria Orders (Queue) --
Order #1 ready: Burger for Alice
Order #2 ready: Pasta for Bob
Order #3 ready: Sandwich for Carol
Order #4 ready: Rice for Dave
Order #5 ready: Noodles for Eve
```

```
C:\> Users > shash > Downloads > AAC 11.7.py > ...
33 |     print(f" Issued '{book}' to {sid}")
34 |
35 print("\n-- 4. Bus Scheduling (Priority Queue) --")
36 buses, ctr = [], 0
37 for route, t in [("City", "07:30"), ("Airport", "09:00"), ("Mall", "08:15"), ("Station", "07:00"), ("Hospital", "10:30")]
38 |     heapq.heappush(buses, (datetime.strptime(t, "%H:%M"), ctr, route)); ctr += 1
39 while buses:
40 |     dt, _, r = heapq.heappop(buses)
41 |     print(f" Dispatch: {r} at {dt.strftime('%H:%M')}")
42 |
43 print("\n-- 5. Cafeteria Orders (Queue) --")
44 oq = deque([(Alice:Burger, Bob:Pasta, Carol:Sandwich, Dave:Rice, Eve:Noodles], 1))
45 while oq:
46 |     num, entry = oq.popleft()
47 |     name, item = entry.split(":")
48 |     print(f" Order #{num} ready: {item} for {name}")
49 
```

TERMINAL

```
PS C:\Users\shash\Downloads> c:& cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50953' '---' 'c:\Users\shash\Downloads\AAC 11.7.py'
MAPPING: Feature -> Data Structure -> Justification
-- 5. Cafeteria Orders (Queue) --
Order #1 ready: Burger for Alice
Order #2 ready: Pasta for Bob
Order #3 ready: Sandwich for Carol
Order #4 ready: Rice for Dave
Order #5 ready: Noodles for Eve
```

Note: Report should be submitted as a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots.