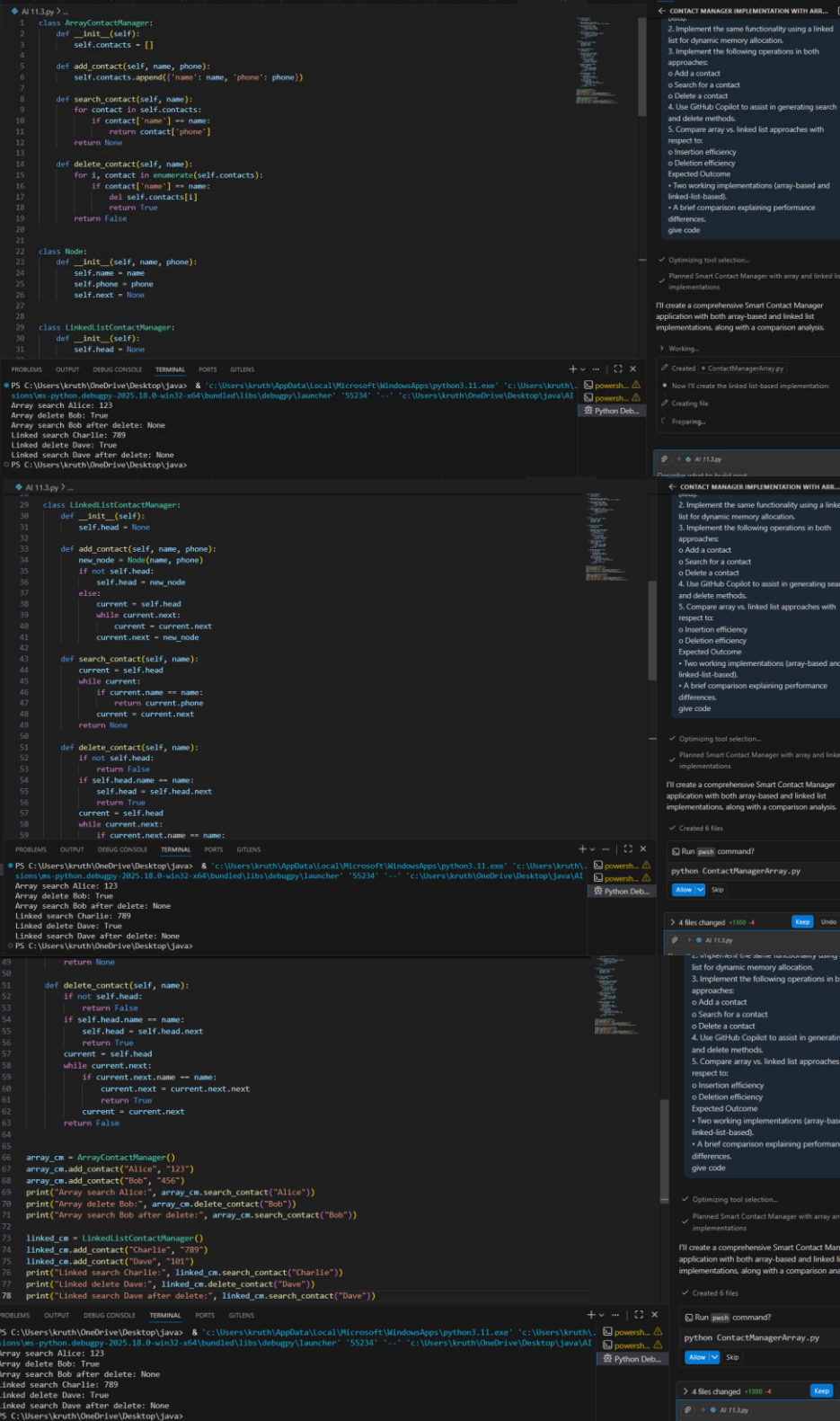


**NAME:CH.Kruthankiran****H.NO:2303A51404****BATCH:26**

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: B. Tech		Assignment Type: Lab	Academic Year:2025-2026
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	Week6 - Wednesday	

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



## Task 2: Library Book Search System (Queues & Priority Queues)

### Scenario

The SRU Library manages book borrow requests. Students and faculty submit

requests, but **faculty requests must be prioritized** over student requests.

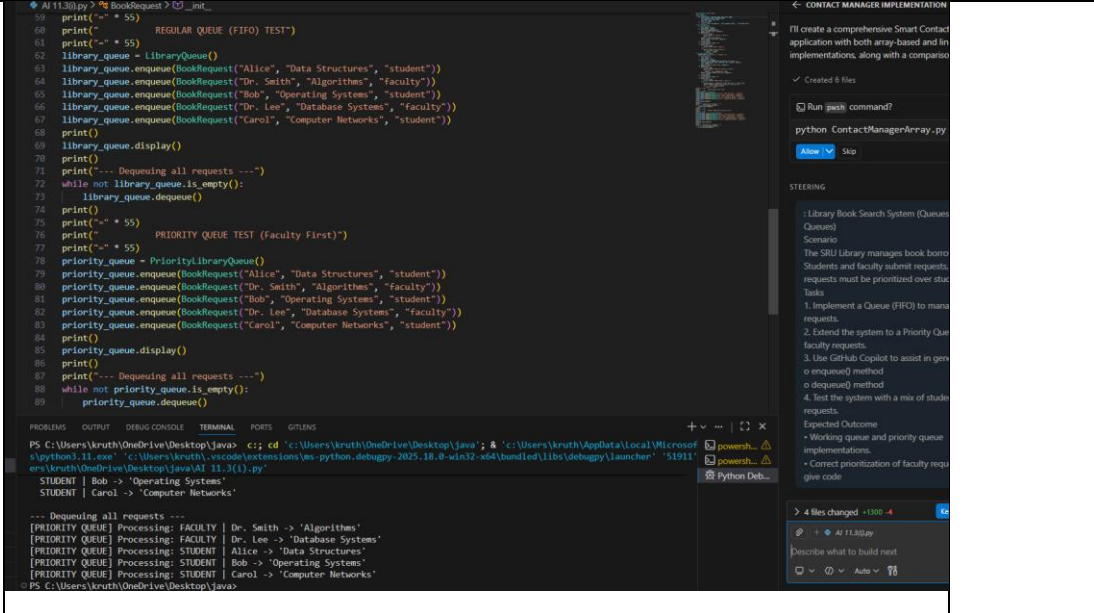
## Tasks

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.

```
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(f"[QUEUE] Current requests:")
30             for req in self._queue:
31                 print(f"    {req}")
32
33 --- Dequeuing all requests ---
34 [PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
35 [PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
36 [PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
37 [PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
38 [PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
39
40 class PriorityLibraryQueue:
41     PRIORITY_MAP = {"Faculty": 0, "Student": 1}
42     def __init__(self):
43         self._heap = []
44         self._counter = 0
45     def enqueue(self, request):
46         priority = self.PRIORITY_MAP.get(request.role.lower(), 1)
47         heapq.heappush(self._heap, (priority, self._counter, request))
48         self._counter += 1
49         print(f"[PRIORITY QUEUE] Added: {request}")
50     def dequeue(self):
51         if not self._heap:
52             print("[PRIORITY QUEUE] No requests in queue.")
53             return None
54         request = heapq.heappop(self._heap)
55         print(f"[PRIORITY QUEUE] Processing: {request}")
56         return request
57     def is_empty(self):
58         return len(self._heap) == 0
59     def display(self):
60         if not self._heap:
61             print("[PRIORITY QUEUE] Queue is empty.")
62         else:
63             sorted_requests = sorted(self._heap, key=lambda x: (x[0], x[1]))
64             print(f"[PRIORITY QUEUE] Current requests (by priority):")
65             for _, _, req in sorted_requests:
66                 print(f"    {req}")
67
68 print("-- * 55)
69 print("
70 print("
71 print("-- * 55)
72
73 --- Dequeuing all requests ---
74 [PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
75 [PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
76 [PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
77 [PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
78 [PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
79
80 PS C:\Users\kruth\OneDrive\Desktop\java>
```

	 <pre> AI 11.30.py &gt; BookRequest &gt; _init_ 59 print("-" * 55) 60 print("    REGULAR QUEUE (FIFO) TEST") 61 print("-" * 55) 62 library_queue = LibraryQueue() 63 library_queue.enqueue(BookRequest("Alice", "Data Structures", "student")) 64 library_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty")) 65 library_queue.enqueue(BookRequest("Bob", "Operating Systems", "student")) 66 library_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty")) 67 library_queue.enqueue(BookRequest("Carol", "Computer Networks", "student")) 68 print() 69 library_queue.display() 70 print() 71 print("--- Dequeuing all requests ---") 72 while not library_queue.is_empty(): 73     library_queue.dequeue() 74 print() 75 print("-" * 55) 76 print("    PRIORITY QUEUE TEST (Faculty First)") 77 print("-" * 55) 78 priority_queue = PriorityLibraryQueue() 79 priority_queue.enqueue(BookRequest("Alice", "Data Structures", "student")) 80 priority_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty")) 81 priority_queue.enqueue(BookRequest("Bob", "Operating Systems", "student")) 82 priority_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty")) 83 priority_queue.enqueue(BookRequest("Carol", "Computer Networks", "student")) 84 print() 85 priority_queue.display() 86 print() 87 print("--- Dequeuing all requests ---") 88 while not priority_queue.is_empty(): 89     priority_queue.dequeue() 90 PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS PS C:\Users\kruth\OneDrive\Desktop\java&gt; c:\; cd 'c:\Users\kruth\OneDrive\Desktop\java'; &amp; 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python11.exe' 'c:\Users\kruth\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '51911' STUDENT   Bob -&gt; 'Operating Systems' STUDENT   Carol -&gt; 'Computer Networks'  --- Dequeuing all requests --- [PRIORITY QUEUE] Processing: FACULTY   Dr. Smith -&gt; 'Algorithms' [PRIORITY QUEUE] Processing: FACULTY   Dr. Lee -&gt; 'Database Systems' [PRIORITY QUEUE] Processing: STUDENT   Alice -&gt; 'Data Structures' [PRIORITY QUEUE] Processing: STUDENT   Bob -&gt; 'Operating Systems' [PRIORITY QUEUE] Processing: STUDENT   Carol -&gt; 'Computer Networks' PS C:\Users\kruth\OneDrive\Desktop\java&gt; </pre> <p><b>Task 3: Emergency Help Desk (Stack Implementation)</b></p> <p><b>Scenario</b></p> <p>SR University's IT Help Desk receives technical support tickets from students and staff. While tickets are received sequentially, issue escalation follows a <b>Last-In, First-Out (LIFO)</b> approach.</p> <p><b>Tasks</b></p> <ol style="list-style-type: none"> <li>1. Implement a <b>Stack</b> to manage support tickets.</li> <li>2. Provide the following operations:       <ul style="list-style-type: none"> <li>o push(ticket)</li> <li>o pop()</li> <li>o peek()</li> </ul> </li> <li>3. Simulate at least <b>five tickets</b> being raised and resolved.</li> <li>4. Use <b>GitHub Copilot</b> to suggest additional stack operations such as:       <ul style="list-style-type: none"> <li>o Checking whether the stack is empty</li> <li>o Checking whether the stack is full (if applicable)</li> </ul> </li> </ol> <p><b>Expected Outcome</b></p> <ul style="list-style-type: none"> <li>• Functional stack-based ticket management system.</li> <li>• Clear demonstration of LIFO behavior.</li> </ul>
--	---



```
Welcome AAC 11.4.py X
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\bundle\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.peak()
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



WelcomeAAC 11.4.py X

C:\Users> shash > Downloads > AAC 11.4.py > ...

```
1 class HashTable:
20
21     def search(self, key):
22         bucket = self.buckets[index]
23         for k, v in bucket:
24             if k == key:
25                 return v
26         return None
27
28     def delete(self, key):
29         index = self._hash(key)
30         bucket = self.buckets[index]
31         for i, (k, v) in enumerate(bucket):
32             if k == key:
33                 del bucket[i]
34                 self.size -= 1
35                 return True
36         return False
37
38     def load_factor(self):
39         return self.size / self.capacity
40
41     def display(self):
42         print(f"[HASH TABLE] capacity={self.capacity}, entries={self.size}, load_factor={self.load_factor()}")
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\Users\shash\Downloads> cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\envs\shashidhar\python.exe' -c 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '60955' '-' 'c:\Users\shash\Downloads\AAC 11.4.py'

[HASH TABLE] capacity=7, entries=8, load\_factor=1.14  
Bucket[ 0]: (frank: frank@ru.edu)  
Bucket[ 2]: (alice: alice.mw@ru.edu) -> (carol: carol@ru.edu) -> (eve: eve@ru.edu)  
Bucket[ 3]: (dave: dave@ru.edu) -> (heidi: heidi@ru.edu)  
Bucket[ 4]: (judy: judy@ru.edu)  
Bucket[ 6]: (ivan: ivan@ru.edu)

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

Created AAC 11.4.py

Created AAC 11.6.py

Tried to run command but was blocked

Run pylanceRunCodeSnippet - pylance mcp server (MCP Server)

Reviewed output for the user

Code saved to [AAC 11.6.py]c:\Users\shash\Downloads\AAC 11.6.py and verified working. Here's a summary:

Methods implemented:

Method	Description
insert(key, value)	Inserts or updates a key-value pair
search(key)	Returns value for key, or None if missing
delete(key)	Removes entry, returns True/False
load_factor()	Returns size / capacity ratio
display()	Prints all non-empty buckets with chains

Collision handling — chaining: With capacity 7 and 10 entries, collisions are visible in the output:

- Bucket[4]: alice -> grace -> heidi (4-way chain)
- Bucket[5]: bob -> frank -> ivan (3-way chain)
- Bucket[3]: eve -> judy (2-way chain)

AAC 11.4.py

Describe what to build next

Claude Sonnet 4.5

WelcomeAAC 11.4.py X

C:\Users> shash > Downloads > AAC 11.4.py > ...

```
41     def display(self):
42         print(f"[HASH TABLE] capacity={self.capacity}, entries={self.size}, load_factor={self.load_factor()}")
43         for i, bucket in enumerate(self.buckets):
44             if bucket:
45                 chain = " -> ".join(f"({k}: {v})" for k, v in bucket)
46                 print(f"Bucket[{i:2}] {chain}")
47
48 print("-" * 55)
49 print("      [HASH TABLE WITH CHAINING DEMO]")
50 print("-" * 55)
51
52 ht = HashTable(capacity=7)
53
54 print("\n--- Inserting entries ---")
55 entries = [
56     ("alice", "alice@ru.edu"),
57     ("bob", "bob@ru.edu"),
58     ("carol", "carol@ru.edu"),
59     ("dave", "dave@ru.edu"),
60     ("eve", "eve@ru.edu"),
61     ("frank", "frank@ru.edu"),
62 ]
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\Users\shash\Downloads> cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\envs\shashidhar\python.exe' -c 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '60955' '-' 'c:\Users\shash\Downloads\AAC 11.4.py'

[HASH TABLE] capacity=7, entries=8, load\_factor=1.14  
Bucket[ 0]: (frank: frank@ru.edu)  
Bucket[ 2]: (alice: alice.mw@ru.edu) -> (carol: carol@ru.edu) -> (eve: eve@ru.edu)  
Bucket[ 3]: (dave: dave@ru.edu) -> (heidi: heidi@ru.edu)  
Bucket[ 4]: (judy: judy@ru.edu)  
Bucket[ 6]: (ivan: ivan@ru.edu)

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

Run pylanceRunCodeSnippet - pylance mcp server (MCP Server)

Reviewed output for the user

Code saved to [AAC 11.6.py]c:\Users\shash\Downloads\AAC 11.6.py and verified working. Here's a summary:

Methods implemented:

Method	Description
insert(key, value)	Inserts or updates a key-value pair
search(key)	Returns value for key, or None if missing
delete(key)	Removes entry, returns True/False
load_factor()	Returns size / capacity ratio
display()	Prints all non-empty buckets with chains

Collision handling — chaining: With capacity 7 and 10 entries, collisions are visible in the output:

- Bucket[4]: alice -> carol -> grace -> heidi (4-way chain)
- Bucket[5]: bob -> frank -> ivan (3-way chain)
- Bucket[3]: eve -> judy (2-way chain)

AAC 11.4.py

Describe what to build next

Claude Sonnet 4.5

Each bucket holds a list of (key, value) pairs; on collision, new entries are simply appended to the chain. On search/delete, the chain is walked linearly until the key matches.

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\anaconda3\Scripts\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\debugpy\_server.py' 'c:\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 > ...
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\anacond
'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\deb
:\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:

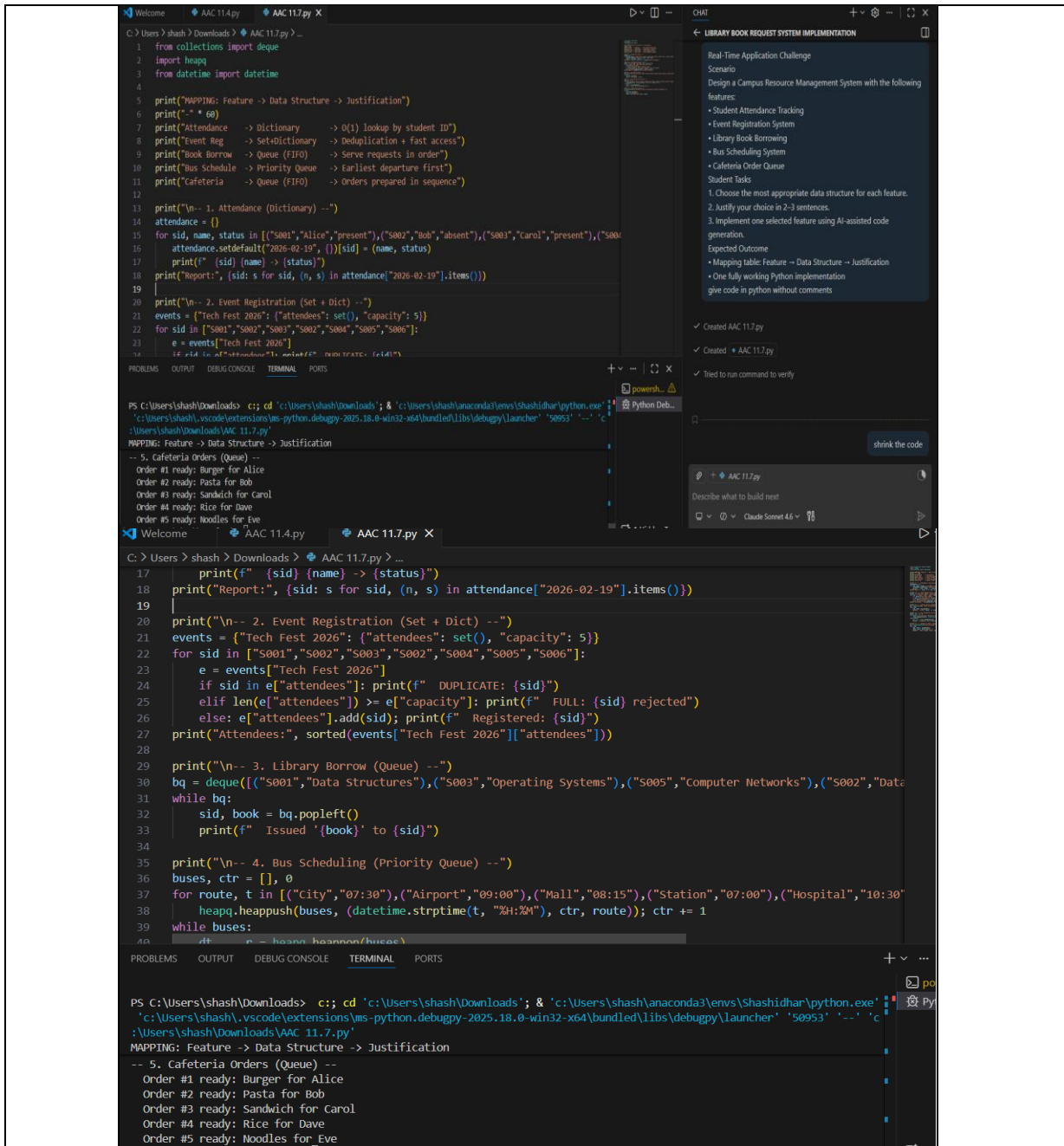
- Student Attendance Tracking
- Event Registration System
- Library Book Borrowing
- Bus Scheduling System
- Cafeteria Order Queue

### 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



```

Welcome  AAC 11.4.py  AAC 11.7.py X
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(enumerate(["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
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\bundle\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.