

# Assignment 11.3

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		Week6 - Wednesday

## Lab Objectives:

The screenshot shows a Microsoft Visual Studio Code (VS Code) interface. On the left, the code editor displays `AAC 11.4.py` containing Python code for a contact manager. The code defines two classes: `ArrayContactManager` and `Node`. The `ArrayContactManager` class has methods for adding, searching, and deleting contacts from an array. The `Node` class represents a node in a linked list. The terminal at the bottom shows the command `conda activate Shashidhar` and some test outputs for both array-based and linked-list implementations.

CHAT

CONTACT MANAGER IMPLEMENTATION WITH ARRAYS AND LINKED LI...

- 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

Expected Outcome

- Two working implementations (array-based and linked-list-based).
- A brief comparison explaining performance differences.

give code in python without comments

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.debugger-2025.18.0-win32-x64\bundled\libs\debug\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: 709  
Linked delete Dave: True  
Linked search Dave after delete: None

PS C:\Users\shash\Downloads>

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%

**VS Code Terminal Screenshot**

The screenshot shows two terminal sessions in VS Code. Both sessions are running Python code related to contact management.

**Top Terminal Session:**

```

Welcome AAC 11.4.py
C: > Users > shash > Downloads > AAC 11.4.py > ArrayContactManager
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
41
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\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
○ PS C:\Users\shash\Downloads>

```

**Bottom Terminal Session:**

```

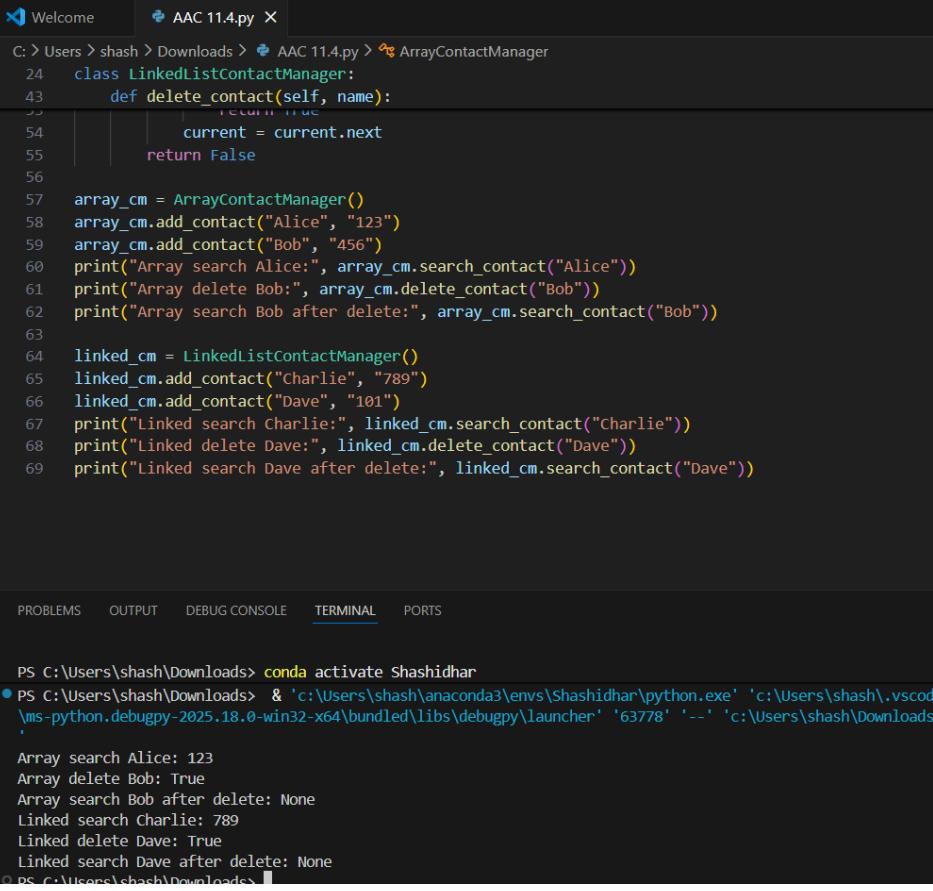
Welcome AAC 11.4.py
C: > Users > shash > Downloads > AAC 11.4.py > ArrayContactManager
24     class LinkedListContactManager:
36         def search_contact(self, name):
37             current = self.head
38             if current:
39                 if current.name == name:
40                     return 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>

```

	 <pre> 24 class LinkedListContactManager: 25     def delete_contact(self, name): 26         if self.head is None: 27             return None 28         current = self.head 29         while current.name != name: 30             current = current.next 31         if current == self.head: 32             self.head = current.next 33         else: 34             previous.current = current.next 35         return True 36 37     current = current.next 38 39     return False 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 </pre> <p>PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS</p> <pre> PS C:\Users\shash\Downloads&gt; conda activate Shashidhar PS C:\Users\shash\Downloads&gt; &amp; 'c:\Users\shash\anaconda3\envs\Shashidhar\python.exe' 'c:\Users\shash\vscode\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63778' '--' 'c:\Users\shash\Downloads\AAC 11.4.py' </pre> <p>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</p>	
	<p><b>Task 2: Library Book Search System (Queues &amp; Priority Queues)</b></p> <p><b>Scenario</b></p> <p>The SRU Library manages book borrow requests. Students and faculty submit requests, but <b>faculty requests must be prioritized</b> over student requests.</p> <p><b>Tasks</b></p> <ol style="list-style-type: none"> <li>1. Implement a <b>Queue (FIFO)</b> to manage book requests.</li> <li>2. Extend the system to a <b>Priority Queue</b>, prioritizing faculty requests.</li> <li>3. Use <b>GitHub Copilot</b> to assist in generating:       <ul style="list-style-type: none"> <li>o enqueue() method</li> <li>o dequeue() method</li> </ul> </li> <li>4. Test the system with a mix of student and faculty requests.</li> </ol> <p><b>Expected Outcome</b></p> <ul style="list-style-type: none"> <li>• Working queue and priority queue implementations.</li> <li>• Correct prioritization of faculty requests.</li> </ul>	

The screenshot shows a developer's workspace with two code editors and a terminal window.

**Code Editor 1 (Top):** The code is for a `LibraryQueue` class. It uses a `deque` from the `collections` module to manage requests. The `enqueue` method adds a request to the queue, and the `dequeue` method removes it. The `is_empty` method checks if the queue is empty. The `__repr__` method returns a string representing the queue's state.

```
1 from collections import deque
2 import heapq
3
4 class BookRequest:
5     def __init__(self, requester_name, book_title, role):
6         self.requester_name = requester_name
7         self.book_title = book_title
8         self.role = role
9
10    def __repr__(self):
11        return f'{self.role.upper()} | {self.requester_name} -> {self.book_title}'
12
13 class LibraryQueue:
14     def __init__(self):
15         self._queue = deque()
16
17     def enqueue(self, request):
18         self._queue.append(request)
19         print(f'[QUEUE] Added: {request}')
20
21     def dequeue(self):
22         if not self._queue:
23             print('[QUEUE] No requests in queue.')
24             return None
25
26         request = self._queue.popleft()
27         print(f'[QUEUE] Processing: {request}')
28
29         return request
30
31     def is_empty(self):
32         return len(self._queue) == 0
```

**Code Editor 2 (Bottom):** The code is for a `PriorityLibraryQueue` class. It uses a min-heap from the `heapq` module to manage requests. The `enqueue` method pushes a request onto the heap, and the `dequeue` method removes the top request from the heap. The `is_empty` method checks if the heap is empty. The `PRIORITY_MAP` dictionary maps roles to priority levels (0 for faculty, 1 for student).

```
15 class LibraryQueue:
16
17     def is_empty(self):
18         return len(self._queue) == 0
19
20     def display(self):
21         if not self._queue:
22             print('[QUEUE] Queue is empty.')
23         else:
24             print('[QUEUE] Current requests:')
25             for req in self._queue:
26                 print(f'  {req}')
27
28 class PriorityLibraryQueue:
29     PRIORITY_MAP = {'faculty': 0, 'student': 1}
30
31     def __init__(self):
32         self._heap = []
33         self._counter = 0
34
35     def enqueue(self, request):
36         priority = self.PRIORITY_MAP.get(request.role.lower(), 2)
37         heapq.heappush(self._heap, (priority, self._counter, request))
38         self._counter += 1
39
40     def dequeue(self):
41         if not self._heap:
42             print('[PRIORITY QUEUE] No requests in queue.')
43
44         request = heapq.heappop(self._heap)[2]
45         print(f'[PRIORITY QUEUE] Added: {request}')
```

**Terminal:** The terminal shows the execution of the code in a Python environment. It prints the processing of various requests (Faculty and Student) from the queue.

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

The screenshot shows a Visual Studio Code interface with a dark theme. The top bar has tabs for 'Welcome' and 'AAC 11.4.py'. The code editor displays a Python file named 'AAC 11.4.py' with the following content:

```
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
41         def is_empty(self):
42             return len(self._heap) == 0
43
44         def display(self):
45             if not self._heap:
46                 print("[PRIORITY QUEUE] Queue is empty.")
47             else:
48                 sorted_requests = sorted(self._heap, key=lambda x: (x[0], x[1]))
49                 print("[PRIORITY QUEUE] Current requests (by priority):")
50                 for _, _, req in sorted_requests:
51                     print(f" {req}")
52
53             print("=" * 55)
54
55             print("      REGULAR QUEUE (FIFO) TEST")
56             print("=" * 55)
57
58             library_queue = LibraryQueue()
59             library_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
60             library_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
```

The terminal below the code editor shows the command run and its output:

```
PS C:\Users\shash\Downloads> c;; cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda3\envs\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("--- Dequeueing 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'
```

```
--- 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'
```

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 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.exe' 'c:\users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\client.py' -d :56757 > C:\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 `pylanceRunCodeSelection` - 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 4.6 ✓

---

**CHAT**

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

- ✓ Tried to run command but was blocked
- ✓ Ran `pylanceRunCodeSelection` - 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:

**Operations implemented:**

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

∅ + AAC 11.4.py

Describe what to build next

CLAude Sonnet 4.6 ✓

---

**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
<code>push(ticket)</code>	Adds ticket to top of stack
<code>pop()</code>	Removes and resolves ticket from top (LIFO)
<code>peek()</code>	Views top ticket without removing it
<code>is_empty()</code>	Returns <code>True</code> if no tickets
<code>is_full()</code>	Returns <code>True</code> if at <code>MAX_SIZE</code>
<code>size()</code>	Returns current ticket count
<code>display()</code>	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.

CLAude Sonnet 4.6 ✓

∅ + AAC 11.4.py

Describe what to build next

CLAude Sonnet 4.6 ✓

The screenshot shows a Microsoft Visual Studio Code interface with a terminal window open. The terminal window has a dark background and displays Python code for a ticket system named AAC 11.4.py. The code includes imports, a stack class definition, and logic for raising, displaying, and resolving tickets. Below the code, the terminal shows the command to run the script and its execution output, which lists five ticket entries and their resolution status.

```
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



```
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\envs\shash\python\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\del\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\py37\python.exe' 'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\deb\pydevd.py' 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



The screenshot shows the Visual Studio Code interface with two tabs open: "AAC 11.4.py" and "AAC 11.7.py". The "AAC 11.7.py" tab contains the following Python code:

```
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
```

The terminal window below shows the output of running the script:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS + v

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.