

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	
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>Lab Objectives: 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 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 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. 	

AI 113.py >...

```

1 class ArrayContactManager:
2     def __init__(self):
3         self.contacts = []
4
5     def add_contact(self, name, phone):
6         self.contacts.append({'name': name, 'phone': phone})
7
8     def search_contact(self, name):
9         for contact in self.contacts:
10            if contact['name'] == name:
11                return contact['phone']
12        return None
13
14    def delete_contact(self, name):
15        for i, contact in enumerate(self.contacts):
16            if contact['name'] == name:
17                del self.contacts[i]
18                return True
19        return False
20
21
22 class Node:
23     def __init__(self, name, phone):
24         self.name = name
25         self.phone = phone
26         self.next = None
27
28
29 class LinkedListContactManager:
30     def __init__(self):
31         self.head = None

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + v ... x PS C:\Users\kruth\OneDrive\Desktop\java> & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\kruth\AI 113.py' powershell powershell Python Deb...

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\kruth\OneDrive\Desktop\java>

AI 113.py >...

```

29 class LinkedListContactManager:
30     def __init__(self):
31         self.head = None
32
33     def add_contact(self, name, phone):
34         new_node = Node(name, phone)
35         if not self.head:
36             self.head = new_node
37         else:
38             current = self.head
39             while current.next:
40                 current = current.next
41             current.next = new_node
42
43     def search_contact(self, name):
44         current = self.head
45         while current:
46             if current.name == name:
47                 return current.phone
48             current = current.next
49         return None
50
51     def delete_contact(self, name):
52         if not self.head:
53             return False
54         if self.head.name == name:
55             self.head = self.head.next
56             return True
57         current = self.head
58         while current:
59             if current.next.name == name:
60                 current.next = current.next.next
61                 return True
62             current = current.next
63         return False
64
65
66 array_cm = ArrayContactManager()
67 array_cm.add_contact("Alice", "123")
68 array_cm.add_contact("Bob", "456")
69 print("Array search Alice:", array_cm.search_contact("Alice"))
70 print("Array delete Bob:", array_cm.delete_contact("Bob"))
71 print("Array search Bob after delete:", array_cm.search_contact("Bob"))
72
73 linked_cm = LinkedListContactManager()
74 linked_cm.add_contact("Charlie", "789")
75 linked_cm.add_contact("Dave", "101")
76 print("Linked search Charlie:", linked_cm.search_contact("Charlie"))
77 print("Linked delete Dave:", linked_cm.delete_contact("Dave"))
78 print("Linked search Dave after delete:", linked_cm.search_contact("Dave"))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + v ... x PS C:\Users\kruth\OneDrive\Desktop\java> & 'c:\Users\kruth\AppData\Local\Microsoft\WindowsApps\python3.11.exe' 'c:\Users\kruth\AI 113.py' powershell powershell Python Deb...

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\kruth\OneDrive\Desktop\java>

AI 113.py >...

```

2. Implement the same functionality using a linked list for dynamic memory allocation.
3. Implement the following operations in both approaches:
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

```

✓ Optimizing tool selection..
Planned Smart Contact Manager with array and linked list implementations

I'll create a comprehensive Smart Contact Manager application with both array-based and linked list implementations, along with a comparison analysis.

> Working...
Created > ContactManagerArray.py
Creating file
< Preparing...
Python ContactManagerArray.py
Run python command?

4 files changed +1000 -4 Keep Undo

AI 113.py

AI 113.py >...

```

2. Implement the same functionality using a linked list for dynamic memory allocation.
3. Implement the following operations in both approaches:
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

```

✓ Optimizing tool selection..
Planned Smart Contact Manager with array and linked list implementations

I'll create a comprehensive Smart Contact Manager application with both array-based and linked list implementations, along with a comparison analysis.

> Created 6 files
Run python command?
Python ContactManagerArray.py
Allow Skip

4 files changed +1000 -4 Keep Undo

AI 113.py

AI 113.py >...

```

2. Implement the same functionality using a linked list for dynamic memory allocation.
3. Implement the following operations in both approaches:
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

```

✓ Optimizing tool selection..
Planned Smart Contact Manager with array and linked list implementations

I'll create a comprehensive Smart Contact Manager application with both array-based and linked list implementations, along with a comparison analysis.

> Created 6 files
Run python command?
Python ContactManagerArray.py
Allow Skip

4 files changed +1000 -4 Keep Undo

AI 113.py

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.

```
AI113000> BookRequest > __init__
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 __str__(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 len(self._queue) == 0:
23              print('[QUEUE] No requests in queue.')
24          return None
25
26      request = self._queue.popleft()
27      print(f'[QUEUE] Processing: {request}')
28      return request
29
30      def is_empty(self):
31          return len(self._queue) == 0
32
33      def display(self):
34          if not self._queue:
35              print('[QUEUE] Queue is empty.')
36          else:
37              print('[QUEUE] Current requests:')
38              for req in self._queue:
39                  print(f' {req}')
40
41
42  PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + x ... x
43
44 PS C:\Users\kruth\OneDrive\Desktop\java> c:; cd c:\Users\kruth\OneDrive\Desktop\java'; & c:\Users\kruth\AppData\Local\Microsoft\Windows\Start Menu\Programs\Python\Python3.11\Scripts\pythonw AI11_3(i).py
45 STUDENT | Bob -> 'Operating Systems'
46 STUDENT | Carol -> 'Computer Networks'
47
48 ... Dequeuing all requests ...
49 [PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
50 [PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
51 [PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
52 [PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
53 [PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
54
55 print(r' {req}' )
56
57 class PriorityLibraryQueue:
58     PRIORITY_MAP = {'faculty': 0, 'student': 1}
59
60     def __init__(self):
61         self._heap = []
62         self._counter = 0
63
64     def enqueue(self, request):
65         priority = self.PRIORITY_MAP.get(request.role.lower(), 1)
66         heapq.heappush(self._heap, (priority, self._counter, request))
67         self._counter += 1
68         print(f'[PRIORITY QUEUE] Added: {request}')
69
70     def dequeue(self):
71         if not self._heap:
72             print('[PRIORITY QUEUE] No requests in queue.')
73             return None
74
75         request = heapq.heappop(self._heap)
76         print(f'[PRIORITY QUEUE] Processing: {request}')
77
78         return request
79
80     def is_empty(self):
81         return len(self._heap) == 0
82
83     def display(self):
84         if not self._heap:
85             print('[PRIORITY QUEUE] Queue is empty.')
86         else:
87             sorted_requests = sorted(self._heap, key=lambda x: (x[0], x[1]))
88             print(f'[PRIORITY QUEUE] Current requests (by priority):')
89             for _, __, req in sorted_requests:
90                 print(f' {req}')
91
92 print('=-' * 55)
93 print("          REGULAR QUEUE (FIFO) TEST")
94 print('=-' * 55)
95
96
97 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS + x ... x
98
99 PS C:\Users\kruth\OneDrive\Desktop\java> c:; cd c:\Users\kruth\OneDrive\Desktop\java'; & c:\Users\kruth\AppData\Local\Microsoft\Windows\Start Menu\Programs\Python\Python3.11\Scripts\pythonw AI11_3(i).py
100 STUDENT | Bob -> 'Operating Systems'
101 STUDENT | Carol -> 'Computer Networks'
102
103 ... Dequeuing all requests ...
104 [PRIORITY QUEUE] Processing: FACULTY | Dr. Smith -> 'Algorithms'
105 [PRIORITY QUEUE] Processing: FACULTY | Dr. Lee -> 'Database Systems'
106 [PRIORITY QUEUE] Processing: STUDENT | Alice -> 'Data Structures'
107 [PRIORITY QUEUE] Processing: STUDENT | Bob -> 'Operating Systems'
108 [PRIORITY QUEUE] Processing: STUDENT | Carol -> 'Computer Networks'
109
110
111 PS C:\Users\kruth\OneDrive\Desktop\java>
```

```

AI 11.30.py > BookRequest > __init__
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 library_queue.enqueue(BookRequest("Bob", "Operating Systems", "student"))
57 library_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty"))
58 library_queue.enqueue(BookRequest("Carol", "Computer Networks", "student"))
59 print()
60 library_queue.display()
61 print()
62 print("... Dequeueing all requests ...")
63 while not library_queue.is_empty():
64     library_queue.dequeue()
65     print()
66 print("=" * 55)
67 print("= PRIORITY QUEUE TEST (Faculty First)")
68 print("=" * 55)
69 priority_queue = PriorityLibraryQueue()
70 priority_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
71 priority_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))
72 priority_queue.enqueue(BookRequest("Bob", "Operating Systems", "student"))
73 priority_queue.enqueue(BookRequest("Dr. Lee", "Database Systems", "faculty"))
74 priority_queue.enqueue(BookRequest("Carol", "Computer Networks", "student"))
75 print()
76 priority_queue.display()
77 print()
78 print("... Dequeueing all requests ...")
79 while not priority_queue.is_empty():
80     priority_queue.dequeue()
81     print()
82 print("= PRIORITY_QUEUE.is_empty():")

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
PS C:\Users\kruth\OneDrive\Desktop> java -cp 'c:\Users\kruth\OneDrive\Desktop\java'; & 'c:\Users\kruth\AppData\Local\Microsoft\Windows\Start Menu\Programs\Python\Python3.11\Scripts' python debug\bookrequest.py
[AI 11.30.py] PS C:\Users\kruth\OneDrive\Desktop> python AI 11.30.py
STUDENT | Bob -> 'Operating Systems'
STUDENT | Carol -> 'Computer Networks'

--- 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'

PS C:\Users\kruth\OneDrive\Desktop>

```

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

Describe what to build next

Claude Sonnet 4.6 ✓

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

- ✓ Created AAC 11.5.py
- ✓ Tried to run command but was blocked
- ✓ Ran pylanceRunCodeSelection - pylance mcp 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

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
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 resolved in reverse order 105→101. The final pop() on an empty stack is handled gracefully with a message instead of crashing.

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
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

[INFO] Stack empty? False

[INFO] Stack full? False

[INFO] Total tickets in stack: 5

... Peeking at top ticket: ...

[PEEK] Next to resolve: [#105] Eve (Student) | Printer offline in Library (NORMAL)

```
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 displays two side-by-side code editors and their corresponding terminal outputs. Both editors show the same Python script, `AAC 11.4.py`, which defines a `HashTable` class with methods for insertion, search, deletion, and display. The right editor shows a modified version of the script, `AAC 11.6.py`, with increased capacity and entries.

Left Monitor (AAC 11.4.py):

```
#!/usr/bin/python
# AAC 11.4.py

class HashTable:
    def __init__(self):
        self.size = 2
        self.buckets = [None] * self.size
        self.load_factor = 1.4
        self.entries = 0
        self.chains = []

    def insert(self, key, value):
        if self.entries >= self.size * self.load_factor:
            self._rehash()
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            bucket = []
            self.buckets[index] = bucket
        else:
            for i, (k, v) in enumerate(bucket):
                if k == key:
                    bucket[i] = (key, value)
                    return
            bucket.append((key, value))
            self.entries += 1
        self._check_load_factor()

    def search(self, key):
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            return None
        for k, v in bucket:
            if k == key:
                return v
        return None

    def delete(self, key):
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            return False
        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]
                self.entries -= 1
                return True
        return False

    def load_factor(self):
        return self.size / self.capacity

    def display(self):
        print(f"\n{len(self.buckets)} HASH TABLE capacity={self.capacity}, entries={self.entries}, load_factor={self.load_factor}:")
        for i, bucket in enumerate(self.buckets):
            if bucket:
                print(f"\tBucket {i}: {bucket}")

if __name__ == "__main__":
    ht = HashTable(capacity=2)
    ht.insert("alice", "alice@sru.edu")
    ht.insert("bob", "bob@sru.edu")
    ht.insert("carol", "carol@sru.edu")
    ht.insert("dave", "dave@sru.edu")
    ht.insert("eve", "eve@sru.edu")
    ht.insert("frank", "frank@sru.edu")
    ht.insert("grace", "grace@sru.edu")
    ht.insert("heidi", "heidig@sru.edu")
    ht.insert("ivan", "ivan@sru.edu")
    ht.display()
```

Right Monitor (AAC 11.6.py):

```
#!/usr/bin/python
# AAC 11.6.py

class HashTable:
    def __init__(self):
        self.size = 7
        self.buckets = [None] * self.size
        self.load_factor = 1.4
        self.entries = 0
        self.chains = []

    def display(self):
        print(f"\n{len(self.buckets)} HASH TABLE capacity={self.capacity}, entries={self.entries}, load_factor={self.load_factor}:")
        for i, bucket in enumerate(self.buckets):
            if bucket:
                print(f"\tBucket {i}: {bucket}")

    def insert(self, key, value):
        if self.entries >= self.size * self.load_factor:
            self._rehash()
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            bucket = []
            self.buckets[index] = bucket
        else:
            for i, (k, v) in enumerate(bucket):
                if k == key:
                    bucket[i] = (key, value)
                    return
            bucket.append((key, value))
            self.entries += 1
        self._check_load_factor()

    def search(self, key):
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            return None
        for k, v in bucket:
            if k == key:
                return v
        return None

    def delete(self, key):
        index = self._hash(key)
        bucket = self.buckets[index]
        if bucket is None:
            return False
        for i, (k, v) in enumerate(bucket):
            if k == key:
                del bucket[i]
                self.entries -= 1
                return True
        return False

    def load_factor(self):
        return self.size / self.capacity

    def display(self):
        print(f"\n{len(self.buckets)} HASH TABLE capacity={self.capacity}, entries={self.entries}, load_factor={self.load_factor}:")
        for i, bucket in enumerate(self.buckets):
            if bucket:
                print(f"\tBucket {i}: {bucket}")

if __name__ == "__main__":
    ht = HashTable(capacity=7)
    ht.insert("alice", "alice@sru.edu")
    ht.insert("bob", "bob@sru.edu")
    ht.insert("carol", "carol@sru.edu")
    ht.insert("dave", "dave@sru.edu")
    ht.insert("eve", "eve@sru.edu")
    ht.insert("frank", "frank@sru.edu")
    ht.insert("grace", "grace@sru.edu")
    ht.insert("heidi", "heidig@sru.edu")
    ht.insert("ivan", "ivan@sru.edu")
    ht.display()
```

Terminal Output:

```
C:\> Users> bash> Downloads> AAC 11.4.py --> ...
C:\> Users> bash> Downloads> AAC 11.6.py --> ...
```

```

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\envs\shash\python\python.exe' 'c:\Users\shash\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\def:\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\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

LIA 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

CHAT

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' '50953' '--' 'c:\Users\shash\Downloads\AAC 11.7.py'

```
MAPPING: Feature -> Data Structure -> Justification
-- 1. Attendance (Dictionary) --
1 from collections import deque
2 import heapq
3 from datetime import datetime
4
5 print("MAPPING: Feature -> Data Structure -> Justification")
6 print("- * 60")
7 print("Attendance -> Dictionary -> O(1) lookup by student ID")
8 print("Event Reg -> Set+Dictionary -> Deduplication + fast access")
9 print("Book Borrow -> Queue (FIFO) -> Serve requests in order")
10 print("Bus Schedule -> Priority Queue -> Earliest departure first")
11 print("Cafeteria -> Queue (FIFO) -> Orders prepared in sequence")
12
13 print("\n-- 1. Attendance (Dictionary) --")
14 attendance = {}
15 for sid, name, status in [("S001", "Alice", "present"), ("S002", "Bob", "absent"), ("S003", "Carol", "present"), ("S004", "Dave", "present"), ("S005", "Eve", "absent")]:
16     attendance.setdefault(datetime.strptime("2026-02-19", "%Y-%m-%d"), {})[sid] = (name, status)
17     print(f" {sid} {name} -> {status}")
18 print("Report:", {sid: s for sid, (n, s) in attendance["2026-02-19"].items()})
19
20 print("\n-- 2. Event Registration (Set + Dict) --")
21 events = {"Tech Fest 2026": {"attendees": set(), "capacity": 5}}
22 for sid in ["S001", "S002", "S003", "S004", "S005", "S006"]:
23     e = events["Tech Fest 2026"]
24     if sid in e["attendees"]:
25         print(f" DUPLICATE: {sid}")
26     elif len(e["attendees"]) >= e["capacity"]:
27         print(f" FULL: {sid} rejected")
28     else:
29         e["attendees"].add(sid)
30         print(f" Registered: {sid}")
31 print("Attendees:", sorted(events["Tech Fest 2026"]["attendees"]))
32
33 print("\n-- 3. Library Borrow (Queue) --")
34 bq = deque([('S001', 'Data Structures'), ('S003', 'Operating Systems'), ('S005', 'Computer Networks'), ('S002', 'Database Systems'), ('S004', 'Machine Learning')])
35 while bq:
36     sid, book = bq.popleft()
37     print(f" Issued '{book}' to {sid}")
38
39 print("\n-- 4. Bus Scheduling (Priority Queue) --")
40 buses, ctr = [], 0
41 for route, t in [('city', '07:30'), ('Airport', '09:00'), ('Mall', '08:15'), ('Station', '07:00'), ('Hospital', '10:30')]:
42     heapq.heappush(buses, (datetime.strptime(t, "%H:%M"), ctr, route))
43     ctr += 1
44 while buses:
45     dt, r = heapq.heappop(buses)
```

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' '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
```

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' '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
```

The screenshot shows a VS Code interface with two terminal tabs open. The left tab displays Python code for bus scheduling and cafeteria orders, while the right tab shows the execution of the code and its output.

Terminal Left (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
```

Terminal Right (Output):

```
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.