

Assignment 11.3

| | | | |
|---|--|--|-------------------------|
| 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 | |

Lab Objectives:

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

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-2025.18.8-x64\bundle\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>

CONTACT MANAGER IMPLEMENTATION WITH ARRAYS AND LINKED L...

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

✓ pwd
✓ Created array_contact_manager.py
✓ Created linked_list_contact_manager.py
✓ Created comparison.txt
I've created three files in your current directory (C:\shash) with the implementations:

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

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

AAC 11.4.py
Describe what to build next
GroK Code Fast 1

```

Welcome AAC 11.4.py X
C: > Users > shash > Downloads > AAC 11.4.py > ArrayContactManager
18 class Node:
19     def __init__(self, name, phone):
20
21         self.phone = phone
22         self.next = None
23
24     class LinkedListContactManager:
25     def __init__(self):
26         self.head = None
27     def add_contact(self, name, phone):
28         new_node = Node(name, phone)
29         if not self.head:
30             self.head = new_node
31         else:
32             current = self.head
33             while current.next:
34                 current = current.next
35             current.next = new_node
36     def search_contact(self, name):
37         current = self.head
38         while current:
39             if current.name == name:
40                 return current.phone
41             current = current.next

```

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\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>

```

```

Welcome AAC 11.4.py X
C: > Users > shash > Downloads > AAC 11.4.py > ArrayContactManager
24 class LinkedListContactManager:
36     def search_contact(self, name):
37
38         return current.phone
39         current = current.next
40     return None
41
42     def delete_contact(self, name):
43         if not self.head:
44             return False
45         if self.head.name == name:
46             self.head = self.head.next
47             return True
48         current = self.head
49         while current.next:
50             if current.next.name == name:
51                 current.next = current.next.next
52                 return True
53             current = current.next
54         return False
55
56     array_cm = ArrayContactManager()
57     array_cm.add_contact("Alice", "123")
58     array_cm.add_contact("Bob", "456")
59     print("Array search Alice:", array_cm.search_contact("Alice"))
60

```

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\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: 43 def delete_contact(self, name): 44 return True 45 def search_contact(self, name): 46 current = self.head 47 while current: 48 if current.name == name: 49 return current 50 current = current.next 51 return False 52 53 array_cm = ArrayContactManager() 54 array_cm.add_contact("Alice", "123") 55 array_cm.add_contact("Bob", "456") 56 print("Array search Alice:", array_cm.search_contact("Alice")) 57 print("Array delete Bob:", array_cm.delete_contact("Bob")) 58 print("Array search Bob after delete:", array_cm.search_contact("Bob")) 59 60 linked_cm = LinkedListContactManager() 61 linked_cm.add_contact("Charlie", "789") 62 linked_cm.add_contact("Dave", "101") 63 print("Linked search Charlie:", linked_cm.search_contact("Charlie")) 64 print("Linked delete Dave:", linked_cm.delete_contact("Dave")) 65 print("Linked search Dave after delete:", linked_cm.search_contact("Dave")) </pre> <pre> 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\ms-python-debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63778' '--' 'c:\Users\shash\Downloads\ </pre> <pre> 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> | |
| | <p>Task 2: Library Book Search System (Queues & Priority Queues)</p> <p>Scenario The SRU Library manages book borrow requests. Students and faculty submit requests, but faculty requests must be prioritized over student requests.</p> <p>Tasks</p> <ol style="list-style-type: none"> 1. Implement a Queue (FIFO) to manage book requests. 2. Extend the system to a Priority Queue, prioritizing faculty requests. 3. Use GitHub Copilot to assist in generating: <ul style="list-style-type: none"> o enqueue() method o dequeue() method 4. Test the system with a mix of student and faculty requests. <p>Expected Outcome</p> <ul style="list-style-type: none"> • Working queue and priority queue implementations. • Correct prioritization of faculty requests. | |

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

from collections import deque
import heapq
class BookRequest:
 def __init__(self, requester_name, book_title, role):
 self.requester_name = requester_name
 self.book_title = book_title
 self.role = role
 def __repr__(self):
 return f"({self.role.upper()}) | ({self.requester_name}) -> '{self.book_title}'"
class LibraryQueue:
 def __init__(self):
 self._queue = deque()
 def enqueue(self, request):
 self._queue.append(request)
 print(f"[QUEUE] Added: {request}")
 def dequeue(self):
 if not self._queue:
 print("[QUEUE] No requests in queue.")
 return None
 request = self._queue.popleft()
 print(f"[QUEUE] Processing: {request}")
 return request
 def is_empty(self):
 return len(self._queue) == 0

PS C:\Users\shash\Downloads> cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\envs\Shashidhar\python.exe' -i
'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '59531' '-...'
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'

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

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.
give code in python without comments just paste in the file

✓ Optimizing tool selection...
✓ Reviewed and created AAC_11.4.py
✓ Created + AAC_11.4.py
✓ Configured a Python environment
✓ Ran pylance-lsp-mcp-server - pylance mcp server (MCP Server)
✓ Reviewed code and provided a brief summary

Describe what to build next

Claude Sonnet 4.5

21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44

class LibraryQueue:
 def is_empty(self):
 return len(self._queue) == 0
 def display(self):
 if not self._queue:
 print("[QUEUE] Queue is empty.")
 else:
 print("[QUEUE] Current requests:")
 for req in self._queue:
 print(f"({req.role}) {req.requester_name} -> '{req.book_title}'")
class PriorityLibraryQueue:
 PRIORITY_MAP = {"faculty": 0, "student": 1}
 def __init__(self):
 self._heap = []
 self._counter = 0
 def enqueue(self, request):
 priority = self.PRIORITY_MAP.get(request.role.lower(), 2)
 heapq.heappush(self._heap, (priority, self._counter, request))
 self._counter += 1
 print(f"[PRIORITY QUEUE] Added: {request}")
 def dequeue(self):
 if not self._heap:

PS C:\Users\shash\Downloads> cd 'c:\Users\shash\Downloads'; & 'c:\Users\shash\anaconda\envs\Shashidhar\python.exe' -i
'c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '59531' '-...'
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'

CHAT

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

✓ Optimizing tool selection...
✓ Reviewed and created AAC_11.4.py
✓ Created + AAC_11.4.py
✓ Configured a Python environment
✓ Ran pylance-lsp-mcp-server - pylance mcp server (MCP Server)
✓ Reviewed code and provided a brief summary

Code written to AAC_11.4.py(c:\Users\shash\Downloads\AAC_11.4.py) and verified working. Here's what the output shows:

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

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

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

Describe what to build next

Claude Sonnet 4.5

```

Welcome
AAC 11.4.py
C: > Users > shash > Downloads > AAC 11.4.py > ...
32 class PriorityLibraryQueue:
43     def dequeue(self):
45         print("[PRIORITY QUEUE] No requests in queue.")
46         return None
47         _, _ , request = heapq.heappop(self._heap)
48         print(f"[PRIORITY QUEUE] Processing: {request}")
49         return request
50     def is_empty(self):
51         return len(self._heap) == 0
52     def display(self):
53         if not self._heap:
54             print("[PRIORITY QUEUE] Queue is empty.")
55         else:
56             sorted_requests = sorted(self._heap, key=lambda x: (x[0], x[1]))
57             print("[PRIORITY QUEUE] Current requests (by priority):")
58             for _, _, req in sorted_requests:
59                 print(f"    {req}")
60 print("=" * 55)
61 print("          REGULAR QUEUE (FIFO) TEST")
62 print("=" * 55)
63 library_queue = LibraryQueue()
64 library_queue.enqueue(BookRequest("Alice", "Data Structures", "student"))
65 library_queue.enqueue(BookRequest("Dr. Smith", "Algorithms", "faculty"))

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

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

```

```

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

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

```

```

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

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'

--- 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.

WelcomeAAC 11.4py X

C:\Users> shash > Downloads > AAC 11.4py > ...
1 class SupportTicket:
2 def __init__(self, ticket_id, requester, issue, priority="normal"):
3 self.ticket_id = ticket_id
4 self.requester = requester
5 self.issue = issue
6 self.priority = priority
7
8 def __repr__(self):
9 return f"[#{self.ticket_id}] {self.requester} | {self.issue} ({self.priority.upper()})"
10
11
12 class HelpDeskStack:
13 MAX_SIZE = 10
14
15 def __init__(self):
16 self._stack = []
17
18 def push(self, ticket):
19 if self.is_full():
20 print(f"[STACK FULL] Cannot add: {ticket}")
21 return
22 self._stack.append(ticket)
23 print(f"[PUSHED] {ticket}")
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
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
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

PS C:\Users\shash\Downloads> cd "c:\Users\shash\Downloads"; & "c:\Users\shash\anaconda\envs\shash\python.exe" "c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher" "51957" "-..." "c:\Users\shash\Downloads\AAC 11.4.py"
SR UNIVERSITY IT HELP DESK - TICKET SYSTEM

--- Raising Support Tickets ---
[PUSHED] #101 Alice (Student) | Cannot access Wi-Fi (NORMAL)
[PUSHED] #102 Bob (Staff) | Projector not working in Lab3 (HIGH)
[PUSHED] #103 Carol (Student) | Email login failure (NORMAL)
[PUSHED] #104 Dr. Smith (Staff) | VPN disconnecting repeatedly (CRITICAL)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

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 v1

WelcomeAAC 11.4py X

C:\Users> shash > Downloads > AAC 11.4py > ...
12 class HelpDeskStack:
23 print(f"[PUSHED] {ticket}")
24
25 def pop(self):
26 if self.is_empty():
27 print(f"[STACK EMPTY] No tickets to resolve.")
28 return None
29 ticket = self._stack.pop()
30 print(f"[RESOLVED] {ticket}")
31 return ticket
32
33 def peek(self):
34 if self.is_empty():
35 print(f"[STACK EMPTY] No tickets in stack.")
36 return None
37 ticket = self._stack[-1]
38 print(f"[PEEK] Next to resolve: {ticket}")
39 return ticket
40
41 def is_empty(self):
42 return len(self._stack) == 0
43
44 def is_full(self):
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
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

PS C:\Users\shash\Downloads> cd "c:\Users\shash\Downloads"; & "c:\Users\shash\anaconda\envs\shash\python.exe" "c:\Users\shash\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher" "51957" "-..." "c:\Users\shash\Downloads\AAC 11.4.py"
[PUSHED] #105 Eve (Student) | Printer offline in Library (NORMAL)

[STACK] Current tickets (top = most recent, total=5):
#105 Eve (Student) | Printer offline in Library (NORMAL)
#104 Dr. Smith (Staff) | VPN disconnecting repeatedly (CRITICAL)
#103 Carol (Student) | Email login failure (NORMAL)
#102 Bob (Staff) | Projector not working in Lab3 (HIGH)
#101 Alice (Student) | Cannot access Wi-Fi (NORMAL)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Python Debug Console

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 v1

WelcomeAAC 11.4py X

C:\Users> shash > Downloads > AAC 11.4py > ...
12 class HelpDeskStack:
40
41 def is_empty(self):
42 return len(self._stack) == 0
43
44 def is_full(self):
45 return len(self._stack) >= self.MAX_SIZE
46
47 def size(self):
48 return len(self._stack)
49
50 def display(self):
51 if self.is_empty():
52 print(f"[STACK] Stack is empty.")
53 else:
54 print(f"[STACK] Current tickets (top = most recent, total={self.size}):")
55 for ticket in reversed(self._stack):
56 print(f" {ticket}")
57
58
59 print(f"--- * 60
60 print(f"SR UNIVERSITY IT HELP DESK - TICKET SYSTEM")
61 print(f"--- * 60
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274

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:

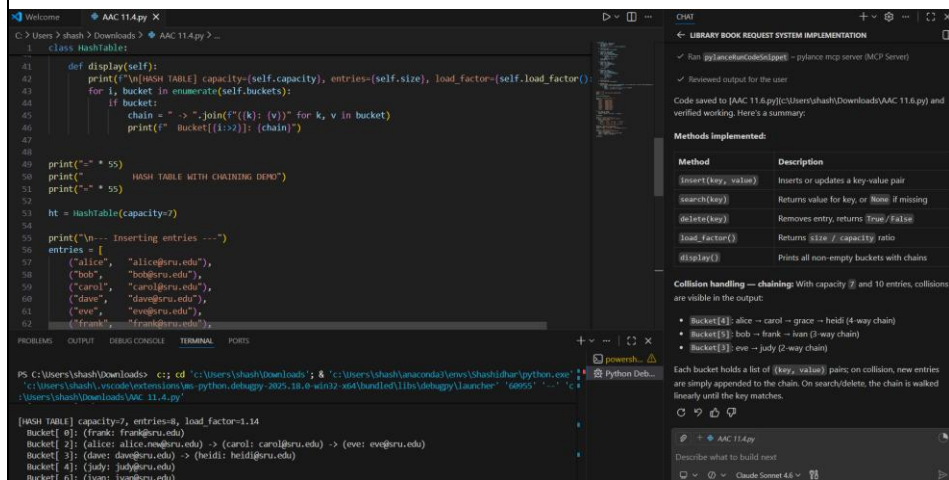
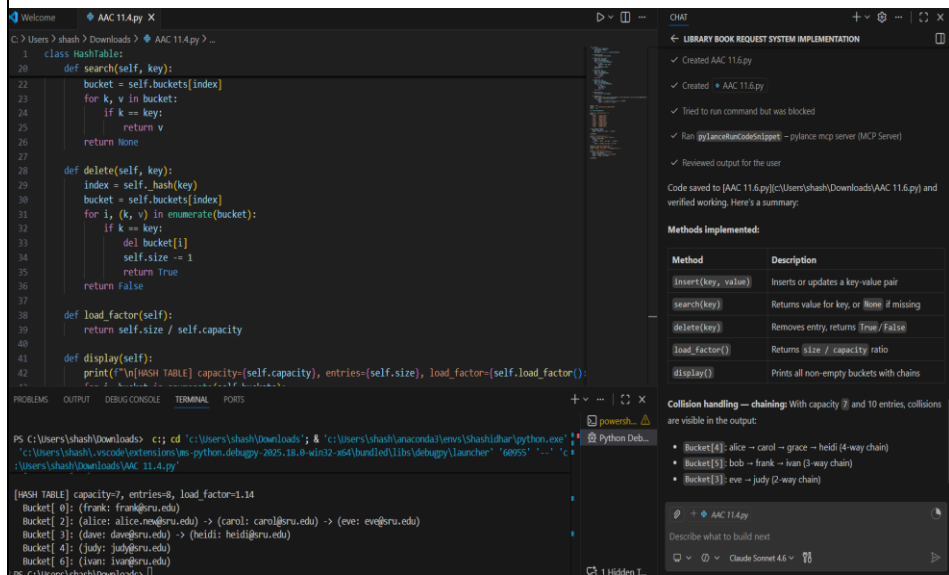
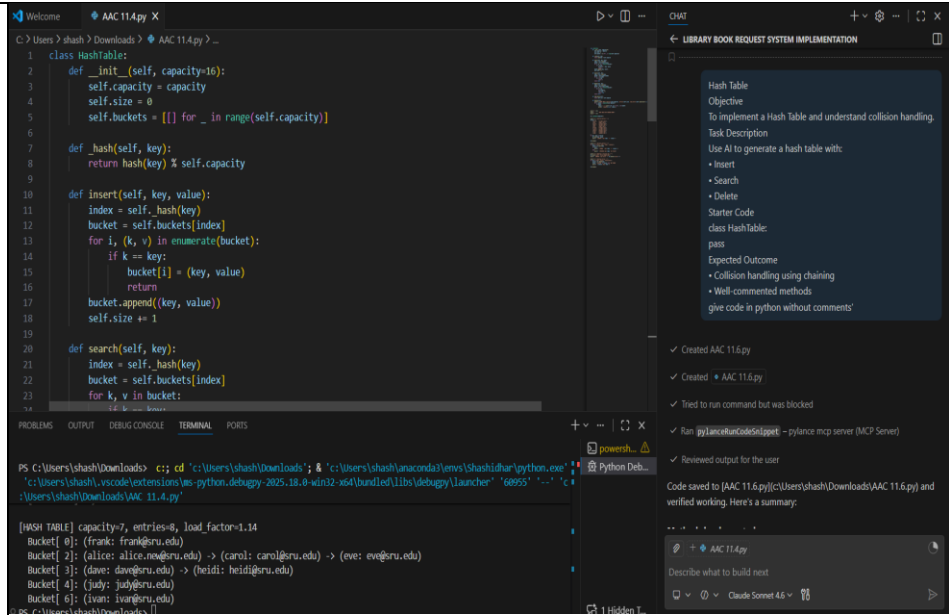
- Insert
- Search
- Delete

Starter Code

```
class HashTable:
    pass
```

Expected Outcome

- Collision handling using chaining
- Well-commented methods



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_launcher.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 > ...
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", "absent"), ("S005", "Eve", "present")]:
16     attendance.setdefault("2026-02-19", {})[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", "S002", "S004", "S005", "S006"]:
23     e = events["Tech Fest 2026"]
24     if sid in e["attendees"] && len(e["attendees"]) >= e["capacity"]:
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); print(f" Registered: {sid}")
30 print("Attendees:", sorted(events["Tech Fest 2026"]["attendees"]))
31
32 print("\n-- 3. Library Borrow (Queue) --")
33 bq = deque([("S001", "Data Structures"), ("S003", "Operating Systems"), ("S005", "Computer Networks"), ("S002", "Data Structures")])
34 while bq:
35     sid, book = bq.popleft()
36     print(f" Issued '{book}' to {sid}")
37
38 print("\n-- 4. Bus Scheduling (Priority Queue) --")
39 buses, ctr = [], 0
40 for route, t in [("City", "07:30"), ("Airport", "09:00"), ("Mall", "08:15"), ("Station", "07:00"), ("Hospital", "10:30")]:
41     heapq.heappush(buses, (datetime.strptime(t, "%H:%M"), ctr, route)); ctr += 1
42 while buses:
43     dt, c, route = heapq.heappop(buses)
44     print(f" Next bus: {route} at {dt.strftime('%H:%M')} (ID: {c})")
45
46 print("\n-- 5. Cafeteria Orders (Queue) --")
47 orders = deque()
48 for i, order in enumerate(["Burger for Alice", "Pasta for Bob", "Sandwich for Carol", "Rice for Dave", "Noodles for Eve"]):
49     orders.append(i, order)
50 while orders:
51     i, order = orders.popleft()
52     print(f" Order #{i+1} ready: {order}")
```

LIBRARY BOOK REQUEST SYSTEM IMPLEMENTATION

Real-Time Application Challenge

Scenario

Design a Campus Resource Management System with the following features:

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

Student Tasks

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

Expected Outcome

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

give code in python without comments

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

shrink the code

```
Welcome AAC 11.4.py AAC 11.7.py X
C:\Users\shash>Downloads> AAC 11.7.py > ...
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", "S002", "S004", "S005", "S006"]:
23     e = events["Tech Fest 2026"]
24     if sid in e["attendees"] && len(e["attendees"]) >= e["capacity"]:
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); print(f" Registered: {sid}")
30 print("Attendees:", sorted(events["Tech Fest 2026"]["attendees"]))
31
32 print("\n-- 3. Library Borrow (Queue) --")
33 bq = deque([("S001", "Data Structures"), ("S003", "Operating Systems"), ("S005", "Computer Networks"), ("S002", "Data Structures")])
34 while bq:
35     sid, book = bq.popleft()
36     print(f" Issued '{book}' to {sid}")
37
38 print("\n-- 4. Bus Scheduling (Priority Queue) --")
39 buses, ctr = [], 0
40 for route, t in [("City", "07:30"), ("Airport", "09:00"), ("Mall", "08:15"), ("Station", "07:00"), ("Hospital", "10:30")]:
41     heapq.heappush(buses, (datetime.strptime(t, "%H:%M"), ctr, route)); ctr += 1
42 while buses:
43     dt, c, route = heapq.heappop(buses)
44     print(f" Next bus: {route} at {dt.strftime('%H:%M')} (ID: {c})")
45
46 print("\n-- 5. Cafeteria Orders (Queue) --")
47 orders = deque()
48 for i, order in enumerate(["Burger for Alice", "Pasta for Bob", "Sandwich for Carol", "Rice for Dave", "Noodles for Eve"]):
49     orders.append(i, order)
50 while orders:
51     i, order = orders.popleft()
52     print(f" Order #{i+1} ready: {order}")
```

```
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
```

```

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.