

3.

```

control_Structure.py
control_Structure.py
35 #
36 # Sample employee data (replace with database queries in a real system)
37 employees = {
38     "employee1": "employee1pass",
39     "employee2": "employee2pass",
40     # Add more employees as needed
41 }
42
43 def authenticate_user(username, password):
44     if username in employees and employees[username] == password:
45         return f"Authentication successful. Welcome, {username} (employee).",
46     else:
47         return "Authentication failed. Invalid username or password."
48
49 # Sample usage
50 username_input = input("Enter your username: ")
51 password_input = input("Enter your password: ")
52 authentication_result = authenticate_user(username_input, password_input)
53 print(authentication_result)
54

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL RUN AND TEST

```

PS D:\workspace\Assignment\assignment_4\python> & "C:\Program Files\Python\python.exe" "d:\workspace\Assignment\assignment_4\python\control_Structure.py"
d:\workspace\Assignment\assignment_4\python\control_Structure.py:39: SyntaxWarning: invalid escape sequence '\d'
...
Enter your username: employee1
Enter your password: employee1pass
Authentication successful. Welcome, employee1 (employee).
PS D:\workspace\Assignment\assignment_4\python>

```

4.

```

control_Structure.py X
control_Structure.py >...
36 ad
37 couriers = {
38     {"CourierID": 1, "Capacity": 10, "location": "A"},
39     {"CourierID": 2, "Capacity": 13, "location": "B"},
40 }
41
42 shipments = {
43     {"ShipmentID": 101, "Weight": 8, "Destination": "A"},
44     {"ShipmentID": 102, "Weight": 12, "Destination": "B"},
45 }
46
47 def assign_courier(shipment):
48     for courier in couriers:
49         if courier["Capacity"] >= shipment["Weight"] and courier["location"] == shipment["Destination"]:
50             return f"Assigned CourierID {courier['CourierID']} to ShipmentID {shipment['ShipmentID']}."
51     return "no suitable courier found for the shipment."
52
53 # Example usage
54 for shipment in shipments:
55     assignment_result = assign_courier(shipment)
56     print(assignment_result)
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
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1
```

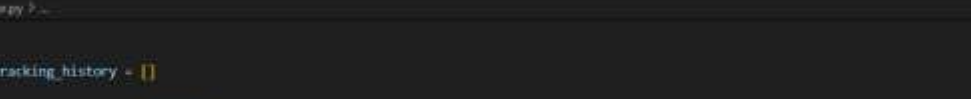
5.

```
control_Structure.py X
control_Structure.py > display_customer_orders
80
81 orders = [
82     ["OrderID": 1, "CustomerName": "customer1", "Status": "Delivered"],
83     ["OrderID": 2, "CustomerName": "customer1", "Status": "Processing"],
84     ["OrderID": 3, "CustomerName": "customer2", "Status": "Delivered"],
85 ]
86
87 def display_customer_orders(customer_name):
88     print(f"Orders for {customer_name}:")
89     for order in orders:
90         if order["CustomerName"] == customer_name:
91             print(f"OrderID: {order['OrderID']}, Status: {order['Status']}")
92
93 # Example usage
94 customer_name_input = input("Enter customer name: ")
95 display_customer_orders(customer_name_input)
96
97
PS D:\Hexaware\Assignment\assignment 4\Python> "C:/Program Files/Python311/python.exe" "d:\Hexaware\Assignment\assignment 4\python/control_structure.py"
Enter customer name: customer1
Orders for customer1:
OrderID: 1, Status: Delivered
OrderID: 2, Status: Processing
PS D:\Hexaware\Assignment\assignment 4\Python>
```

6.

```
control_Structure.py X
control_Structure.py >
98
99
100 courier = {"CourierID": 1, "CurrentLocation": "A", "Destination": "B"}
101
102 def track_courier_location(courier):
103     print(f"Courier {courier['CourierID']} is currently at {courier['CurrentLocation']}")
104     while courier['CurrentLocation'] != courier['Destination']:
105         new_location = input("Enter the new location of the courier: ")
106         courier['CurrentLocation'] = new_location
107         print(f"Courier {courier['CourierID']} is now at {courier['CurrentLocation']}")
108
109     print(f"Courier {courier['CourierID']} has reached its destination {courier['Destination']}")
110
111 # Example usage
112 track_courier_location(courier)
113
114
115
PS D:\Hexaware\Assignment\assignment 4\Python> "C:/Program Files/Python311/python.exe" "d:\Hexaware\Assignment\assignment 4\python/control_structure.py"
Courier 1 is currently at A.
Enter the new location of the courier: B
Courier 1 is now at B.
Courier 1 has reached its destination B.
PS D:\Hexaware\Assignment\assignment 4\Python>
```

7.



```

115 """
116 #?
117 parcel_tracking_history = []
118
119 def update_parcel_location(location):
120     parcel_tracking_history.append(location)
121     print("Parcel tracking history updated. Current location: (location)")
122
123 # Sample usage
124 update_parcel_location("Warehouse A")
125 update_parcel_location("In Transit")
126 update_parcel_location("Destination B")
127

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Hexaware\Assignment\assignment 4\Python & "C:\Program Files\Python12\python.exe" "d:\Hexaware\Assignment\assignment 4\Python\control_structure.py"

Parcel tracking history updated. Current location: Warehouse A

Parcel tracking history updated. Current location: In Transit

Parcel tracking history updated. Current location: Destination B

PS D:\Hexaware\Assignment\assignment 4\Python

8.

```

131 ##
132
133 couriers = [
134     ("CourierID": 1, "CurrentLocation": "Warehouse A"),
135     ("CourierID": 2, "CurrentLocation": "Warehouse B"),
136     ("CourierID": 3, "CurrentLocation": "Warehouse C"),
137 ]
138
139
140 def find_nearest_courier(destination):
141     nearest_courier = None
142     min_distance = float('inf')
143
144     for courier in couriers:
145         distance = abs(ord(courier['CurrentLocation'][0]) - ord(destination[0]))
146
147         if distance < min_distance:
148             min_distance = distance
149             nearest_courier = courier
150
151     return nearest_courier
152
153
154 # Example usage
155 order_destination_input = input("Enter the destination for the new order: ")
156 nearest_courier = find_nearest_courier(order_destination_input)
157 print(f"The nearest available courier is CourierID {nearest_courier['CourierID']} at {nearest_courier['CurrentLocation']}")
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

```

9.

```

control_Structure.py ✕
control_Structure.py > ...
101 #
102 parcel_tracking_data = [
103     ["345678", "In transit"],
104     ["789012", "out for Delivery"],
105     ["345678", "Delivered"],
106 ]
107
108 def track_parcel(parcel_number):
109     for entry in parcel_tracking_data:
110         if entry[0] == parcel_number:
111             status = entry[1]
112             if status == "In transit":
113                 print("Parcel in transit.")
114             elif status == "out for Delivery":
115                 print("Parcel out for delivery.")
116             elif status == "Delivered":
117                 print("Parcel delivered.")
118             else:
119                 print("Invalid status.")
120     return
121     print("Parcel not found.")
122
123 # sample usage
124 parcel_number_input = input("Enter the parcel tracking number: ")
125 track_parcel(parcel_number_input)
126
127
128 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
129 Python + - [ ] ... ^ x
130 PS D:\Vexware\Assignment\assignment 4\Python > "C:\Program Files\Python112\python.exe" "D:\Vexware\Assignment\assignment 4\Python\control_structure.py"
131 Enter the parcel tracking number: 789012
132 Parcel out for delivery.
133 PS D:\Vexware\Assignment\assignment 4\Python >

```

10.

The screenshot shows a code editor with a file named `control_structure.py`. The code defines a function `validate_customer_info(data, detail)` that checks if input data matches a specific detail type (name, address, or phone number) using regular expressions. Below the function, there are three example usages where user input is taken and validated.

```
187 #10
188 import re
189
190 def validate_customer_info(data, detail):
191     if detail == "name":
192         return data.isalpha() and data.istitle()
193     elif detail == "address":
194         return data.isalnum() and not any(char.isdigit() for char in data)
195     elif detail == "phone number":
196         return re.match(r'^\d{3}-\d{3}-\d{4}$', data) is not None
197
198     else:
199         return False
200
201 # Example usage
202 name_input = input("Enter customer name: ")
203 print(validate_customer_info(name_input, "name"))
204
205 address_input = input("Enter customer address: ")
206 print(validate_customer_info(address_input, "address"))
207
208 phone_number_input = input("Enter customer phone number (format: ###-###-####): ")
209 print(validate_customer_info(phone_number_input, "phone number"))
210
211
```

Below the code editor, a terminal window displays the output of running the script:

```
PS D:\Hexaware\Assignment\assignment_4\Python > .\control_structure.py
Enter customer name: Rohan
True
Enter customer address: Jalgaon
True
Enter customer phone number (format: ###-###-####): 123-456-7894
True
PS D:\Hexaware\Assignment\assignment_4\Python >
```


11.

[illegible]

12.

The image shows a VS Code editor with a Python file named `control_Structure.py`. The script defines a function `generate_order_confirmation_email` that takes four arguments: `customer_name`, `order_number`, `delivery_address`, and `delivery_date`. It constructs an email content string with placeholders for these values. Below the function, there is an example usage section that prompts the user for each input and then calls the function to generate and print the confirmation email.

```

206 #12
207 def generate_order_confirmation_email(customer_name, order_number, delivery_address, delivery_date):
208     email_content = {
209         "Dear %s,\n"
210         "Order Confirmation:\n"
211         "Order Number: %s\n"
212         "Delivery Address: %s\n"
213         "Expected Delivery Date: %s\n"
214         "Thank you for choosing our courier service!"
215         % (customer_name, order_number, delivery_address, delivery_date)
216     }
217     return email_content
218
219 # Example usage
220 customer_name_input = input("Enter customer name: ")
221 order_number_input = input("Enter order number: ")
222 delivery_address_input = input("Enter delivery address: ")
223 delivery_date_input = input("Enter expected delivery date: ")
224 confirmation_email = generate_order_confirmation_email(customer_name_input, order_number_input, delivery_address_input, delivery_date_input)
225 print(confirmation_email)
226

```

The terminal window at the bottom shows the execution of the script. It prompts for the customer name (Rohan), order number (1), delivery address (Jalgaon), and expected delivery date (12). The output shows the formatted email content, including the order confirmation details and a thank-you message.

```

PS D:\vscode\Assignment\assignment 4\Python & "C:/Program Files/Python12/python.exe" "d:/vscode/Assignment/assignment 4/Python/control_Structure.py"
Enter customer name: Rohan
Enter order number: 1
Enter delivery address: Jalgaon
Enter expected delivery date: 12
Dear Rohan,

Order Confirmation:
Order Number: 1
Delivery Address: Jalgaon
Expected Delivery date: 12

Thank you for choosing our courier service!
PS D:\vscode\Assignment\assignment 4\Python

```


The screenshot displays a Visual Studio Code window with a Python file named `control_structure.py`. The code defines a function `calculate_shipping_cost` which takes source address, destination address, and parcel weight as inputs. It calculates the distance by comparing the first character of each address and multiplies it by the parcel weight and a constant factor of 0.1. An example usage section prompts the user for input values.

```
def calculate_shipping_cost(source_address, destination_address, parcel_weight):
    # Replace with your actual distance and cost calculation logic
    distance = abs(ord(source_address[0]) - ord(destination_address[0]))
    shipping_cost = distance * parcel_weight * 0.1
    return shipping_cost

# Example usage
source_address_input = input("Enter source address: ")
destination_address_input = input("Enter destination address: ")
parcel_weight_input = float(input("Enter parcel weight: "))
shipping_cost = calculate_shipping_cost(source_address_input, destination_address_input, parcel_weight_input)
print("Shipping Cost:", shipping_cost)
```

The bottom panel shows the terminal output after running the script:

```
PS D:\Hexaware\Assignment\assignment 4\Python> "C:\Program Files\Python111\python.exe" "D:\hexaware\assignment\assignment 4\python/control_structure.py"
Enter source address: A
Enter destination address: C
Enter parcel weight: 5
Shipping Cost: 1.0
PS D:\Hexaware\Assignment\assignment 4\Python>
```

[illegible]

```
control_structure.py X
control_structure.py > ...
257
258
259 def find_similar_addresses(address, addresses_list):
260     similar_addresses = [addr for addr in addresses_list if addr==address]
261     return similar_addresses
262
263 # Example usage
264 addresses_list = ["123 Main St, City1", "456 First Ave, City2", "123 Main St, City1", "789 Second St, City4"]
265 address_input = input("Enter an address: ")
266 similar_addresses = find_similar_addresses(address_input, addresses_list)
267 print("Similar Addresses:", similar_addresses)
268
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Hexaware\Assignment\Assignment 4\Python & > "C:\Program Files\Python312\python.exe" "D:\Hexaware\Assignment\Assignment 4\Python\control_structure.py"

Enter an address: 123 Main St, City1

Similar Addresses: ['123 Main St, City1']

PS D:\Hexaware\Assignment\Assignment 4\Python >

Task(5,6,7,8):

```
class.py X
class User:
    def __init__(self, userID, userName, email, password, contactNumber, address):
        self.__userID = userID
        self.__userName = userName
        self.__email = email
        self.__password = password
        self.__contactNumber = contactNumber
        self.__address = address

class Courier:
    def __init__(self, courierID, senderName, senderAddress, receiverName, receiverAddress, weight, status, trackingNumber, deliveryDate,
        self.__courierID = courierID
        self.__senderName = senderName
        self.__senderAddress = senderAddress
        self.__receiverName = receiverName
        self.__receiverAddress = receiverAddress
        self.__weight = weight
        self.__status = status
        self.__trackingNumber = trackingNumber
        self.__deliveryDate = deliveryDate
        self.__userId = userID

class Employee:
    def __init__(self, employeeID, employeeName, email, contactNumber, role, salary):
        self.__employeeID = employeeID
        self.__employeeName = employeeName
        self.__email = email
        self.__contactNumber = contactNumber
        self.__role = role
        self.__salary = salary

class Location:
    def __init__(self, locationID, locationName, address):
        self.__locationID = locationID
        self.__locationName = locationName
        self.__address = address
```

```
class.py
class CourierCompany:
    def __init__(self, companyName):
        self.__companyName = companyName
        self.__courierDetails = [] # Collection of Courier objects
        self.__employeeDetails = [] # Collection of Employee objects
        self.__locationDetails = [] # Collection of Location objects

class Payment:
    def __init__(self, paymentID, courierID, amount, paymentDate):
        self.__paymentID = paymentID
        self.__courierID = courierID
        self.__amount = amount
        self.__paymentDate = paymentDate

#
from abc import ABC, abstractmethod
import random

class ICourierUserService(ABC):
    @abstractmethod
    def placeOrder(self, courierObj):
        pass

    @abstractmethod
    def getOrderStatus(self, trackingNumber):
        pass

    @abstractmethod
    def cancelOrder(self, trackingNumber):
        pass

    @abstractmethod
    def getAssignedOrder(self, courierStaffID):
```



```

class.py
class.py
72 @abstractmethod
73 def getAssignedOrder(self, courierStaffId):
74     pass
75
76
77 class Courier:
78     unique_tracking_number = random.randint(1000, 9999)
79
80     def __init__(self, senderName, senderAddress, receiverName, receiverAddress, weight, userId):
81         self.trackingNumber = Courier.unique_tracking_number
82         Courier.unique_tracking_number += 1
83
84
85 # Example of usage
86 class CourierUserService(ICourierUserService):
87
88     def placeOrder(self, courierObj):
89         return courierObj.trackingNumber
90
91     def getOrderStatus(self, trackingNumber):
92         return "In Transit"
93
94     def cancelOrder(self, trackingNumber):
95         return True
96
97     def getAssignedOrder(self, courierStaffId):
98         return [1234, 5678, 91011]
99
100
101 # Example of usage
102 courierObj = Courier("John Doe", "123 Main St", "Jane Doe", "456 Second St", 2.5, 1)
103 courierUserService = CourierUserService()
104 tracking_number = courierUserService.placeOrder(courierObj)
105 print(f"Tracking Number: {tracking_number}")
106 status = courierUserService.getOrderStatus(tracking_number)
107 print(f"Order Status: {status}")
108 canceled = courierUserService.cancelOrder(tracking_number)
109 print(f"Order Canceled: {canceled}")

```

Implementation of methods in classes:

```

class.py
class.py
111 from abc import ABC, abstractmethod
112 from datetime import datetime
113 import random
114
115
116 class Employee:
117     def __init__(self, employeeID, employeeName, contactNumber):
118         self.employeeID = employeeID
119         self.employeeName = employeeName
120         self.contactNumber = contactNumber
121
122
123 class Courier:
124     tracking_number_counter = random.randint(1000, 9999)
125
126     def __init__(self, senderName, senderAddress, receiverName, receiverAddress, weight, userId):
127         self.trackingNumber = Courier.tracking_number_counter
128         Courier.tracking_number_counter += 1
129         self.senderName = senderName
130         self.senderAddress = senderAddress
131         self.receiverName = receiverName
132         self.receiverAddress = receiverAddress
133         self.weight = weight
134         self.status = "yetToTransit"
135         self.userId = userId
136
137
138 class ICourierUserService(ABC):
139     @abstractmethod
140     def placeOrder(self, courierObj):
141         pass
142
143     @abstractmethod
144     def getOrderStatus(self, trackingNumber):
145         pass
146
147     @abstractmethod
148     def cancelOrder(self, trackingNumber):
149         pass

```

```

class.py X
class.py > CourierUserService > placeOrder
144
145     @abstractmethod
146     def getAssignedOrder(self, courierStaffId):
147         pass
148
149
150 class ICourierAdminService(ABC):
151     @abstractmethod
152     def addCourierStaff(self, name, contactNumber):
153         pass
154
155
156 # Custom Exceptions
157 class TrackingNumberNotFoundException(Exception):
158     pass
159
160 class InvalidEmployeeIdException(Exception):
161     pass
162
163
164 class Courier:
165     tracking_number_counter = random.randint(1000, 9999)
166
167     def __init__(self, senderName, senderAddress, receiverName, receiverAddress, weight, userId):
168         self.trackingNumber = Courier.tracking_number_counter
169         Courier.tracking_number_counter += 1
170         self.senderName = senderName
171         self.senderAddress = senderAddress
172         self.receiverName = receiverName
173         self.receiverAddress = receiverAddress
174         self.weight = weight
175         self.status = "yetToTransit"
176         self.userId = userId
177
178
179 class CourierService(ICourierUserService, ICourierAdminService):
180     courier_orders = []
181     courier_staff = []
182
183     def placeOrder(self, courierObj):
184         CourierService.courier_orders.append(courierObj)

```

```

class.py X
class.py > CourierUserService > placeOrder
185         return courierObj.trackingNumber
186
187     def getOrderStatus(self, trackingNumber):
188         for order in CourierService.courier_orders:
189             if order.trackingNumber == trackingNumber:
190                 return order.status
191         raise TrackingNumberNotFoundException("tracking number not found.")
192
193     def cancelOrder(self, trackingNumber):
194         for order in CourierService.courier_orders:
195             if order.trackingNumber == trackingNumber:
196                 if order.status == "yetToTransit":
197                     CourierService.courier_orders.remove(order)
198                     return True
199                 else:
200                     raise TrackingNumberNotFoundException("cannot cancel an order that is already in transit or delivered.")
201         raise TrackingNumberNotFoundException("tracking number not found.")
202
203     def getAssignedOrder(self, courierStaffId):
204         try:
205             staff_orders = [order for order in CourierService.courier_orders if order.userId == courierStaffId]
206             if not staff_orders:
207                 raise TrackingNumberNotFoundException("No orders assigned to the specified courier staff.")
208             return staff_orders
209         except TrackingNumberNotFoundException as e:
210             print(f"Exception: {e}")
211
212     def addCourierStaff(self, name, contactNumber):
213         new_staff = Employee(employeeID=len(CourierService.courier_staff) + 1, employeeName=name, contactNumber=contactNumber)
214         CourierService.courier_staff.append(new_staff)
215         return new_staff.employeeID
216
217     def getEmployeeNameById(self, employeeID):
218         for employee in CourierService.courier_staff:
219             if employee.employeeID == employeeID:
220                 return employee.employeeName
221         raise InvalidEmployeeIdException("Invalid Employee ID.")

```

```

class.py X
class.py > CourierService > placeOrder
222
223 courier_service = CourierService()
224
225 try:
226     # Place Order
227     order1 = Courier("John Doe", "123 Main St", "Jane Doe", "456 Second St", 10, userId=1)
228     tracking_number = courier_service.placeOrder(order1)
229     print(f"Order placed. Tracking Number: {tracking_number}")
230
231     # Get Order Status
232     status = courier_service.getOrderStatus(tracking_number)
233     print(f"Order Status: {status}")
234
235     # Cancel Order
236     cancellation_result = courier_service.cancelOrder(tracking_number)
237     print(f"Order Cancellation Result: {cancellation_result}")
238
239     # Get Assigned Orders
240     assigned_orders = courier_service.getAssignedOrder(courierStaffId=1)
241     print(f"Assigned Orders: {assigned_orders}")
242
243     # Add Courier Staff
244     staff_id = courier_service.addCourierStaff(name="Courier Staff 1", contactNumber="9876543210")
245     print(f"New Courier Staff added. Staff ID: {staff_id}")
246
247     # Get Employee Name by ID
248     employee_name = courier_service.getEmployeeNameById(employeeId=1)
249     print(f"Employee Name: {employee_name}")
250
251 except (TrackingNumberNotFoundException, InvalidEmployeeIdException) as e:
252     print(f"Exception: {e}")
253 except Exception as e:
254     print(f"Unexpected Exception: {e}")
255 finally:
256     print("Execution completed.")
257

```

```

class.py X
class.py > CourierService > placeOrder
256
257     print("Execution completed.")
258
259
260 #110
261 import re
262
263 def validate_customer_info(data, detail):
264     if detail == "name":
265         return data.isalpha() and data.istitle()
266     elif detail == "address":
267         return data.isalnum() and not any(char.isdigit() for char in data)
268     elif detail == "phone number":
269         return re.match(r"^\d{3}-\d{3}-\d{4}$", data) is not None
270
271     else:
272         return False
273
274 # Example usage
275 name_input = input("Enter customer name: ")
276 print(validate_customer_info(name_input, "name"))
277
278 address_input = input("Enter customer address: ")
279 print(validate_customer_info(address_input, "address"))
280
281 phone_number_input = input("Enter customer phone number (format: xxx-xxx-xxxx): ")
282 print(validate_customer_info(phone_number_input, "phone number"))

```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Hexaware\Assignment\Assignment 4\Python & "C:/Program Files/Python312/python.exe" "d:/Hexaware/Assignment/Assignment 4/Python/classes.py"
Order placed, Tracking Number: 1574
Order Status: yetToTransit
Order Cancellation Result: True
Exception: No orders assigned to the specified courier staff.
Assigned Orders: none
New Courier Staff added, Staff ID: 1
Employee Name: Courier Staff 1
Execution completed.
Enter customer name: █
```

New inputs:

```
124 courier_service = CourierService()
125
126 try:
127     # Place Order
128     order1 = Courier("Rohan ", "123 Jalgaon", "Rohan Chaudhari", "456 Second St", 11, userId=2)
129     tracking_number = courier_service.placeOrder(order1)
130     print(f"Order placed, Tracking Number: {tracking_number}")
131
132     # Get Order Status
133     status = courier_service.getOrderStatus(tracking_number)
134     print(f"Order Status: {status}")
135
136     # Cancel Order
137     cancellation_result = courier_service.cancelOrder(tracking_number)
138     print(f"Order Cancellation Result: {cancellation_result}")
139
140     # Get Assigned Orders
141     assigned_orders = courier_service.getAssignedOrder(courierStaffId=1)
142     print(f"Assigned Orders: {assigned_orders}")
143
144     # Add Courier Staff
145     staff_id = courier_service.addCourierStaff(name="Courier Staff 2", contactNumber="9876543210")
146     print(f"New Courier Staff added, Staff ID: {staff_id}")
147
148     # Get Employee Name by ID
149     employee_name = courier_service.getEmployeeNameById(employeeID=2)
150     print(f"Employee Name: {employee_name}")
151
152 except (TrackingNumberNotFoundException, InvalidEmployeeIdException) as e:
153     print(f"Exception: {e}")
154 except Exception as e:
155     print(f"Unexpected Exception: {e}")
156 finally:
157     print("Execution completed.")
158
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Hexaware\Assignment\Assignment 4\Python & "C:/Program Files/Python312/python.exe" "d:/Hexaware/Assignment/Assignment 4/Python/classes.py"
Order placed, Tracking Number: 3798
Order Status: yetToTransit
Order Cancellation Result: True
Exception: No orders assigned to the specified courier staff.
Assigned Orders: none
New Courier Staff added, Staff ID: 1
Exception: Invalid Employee ID.
Execution completed.
Enter customer name: Rohit
True
Enter customer address: Jalgaon
True
Enter customer phone number (format: ###-###-####): 456-564-1234
True
PS D:\Hexaware\Assignment\Assignment 4\Python █
```


Task9:

```
database.py X
database.py > ...
1 import mysql.connector
2 from mysql.connector import Error
3
4 class DBConnection:
5     connection = None
6
7     @staticmethod
8     def getConnection():
9         if DBConnection.connection is None:
10             try:
11                 DBConnection.connection = mysql.connector.connect(
12                     host='localhost',
13                     user='root',
14                     password='root',
15                     database='couriermanagementsystem'
16                 )
17                 if DBConnection.connection.is_connected():
18                     print("Connected to the database")
19
20             except mysql.connector.Error as e:
21                 print(f"Error: {e}")
22
23         return DBConnection.connection
24
25
26 class CourierServiceDB:
27     connection = DBConnection.getConnection()
28
29
30     def insertOrder(courierId, senderName, senderAddress, receiverName, receiverAddress, weight, status, trackingNumber, deliveryDate):
31         try:
32             cursor = CourierServiceDB.connection.cursor()
33             query = "INSERT INTO Couriers (courierID, SenderName, SenderAddress, ReceiverName, ReceiverAddress, Weight, Status, TrackingNumber, DeliveryDate)"
34             query = "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)"
35             data = (courierId, senderName, senderAddress, receiverName, receiverAddress, weight, status, trackingNumber, deliveryDate)
36             cursor.execute(query, data)
```

```
database.py X
database.py > ...
37             cursor.execute(query, data)
38             CourierServiceDB.connection.commit()
39             print("Order inserted successfully.")
40
41         except Error as e:
42             print(f"Error: {e}")
43         finally:
44             cursor.close()
45
46
47     def updateCourierStatus(trackingNumber, newStatus):
48         try:
49             cursor = CourierServiceDB.connection.cursor()
50             query = "UPDATE Couriers SET Status = %s WHERE TrackingNumber = %s"
51             data = (newStatus, trackingNumber)
52
53             cursor.execute(query, data)
54             CourierServiceDB.connection.commit()
55             print("Courier status updated successfully.")
56
57         except Error as e:
58             print(f"Error: {e}")
59         finally:
60             cursor.close()
61
62
63     def getDeliveryHistory(trackingNumber):
64         try:
65             cursor = CourierServiceDB.connection.cursor()
66             query = "SELECT * FROM Couriers WHERE TrackingNumber = %s"
67             data = (trackingNumber,)
68
69             cursor.execute(query, data)
70             result = cursor.fetchall()
71             return result
72
```


Inputs:

```

66         cursor = CourierServiceDb.connection.cursor()
67         query = "SELECT * FROM Couriers WHERE TrackingNumber = %s"
68         data = (trackingNumber,)
69
70         cursor.execute(query, data)
71         result = cursor.fetchall()
72         return result
73
74     except Error as e:
75         print(f"Error: {e}")
76     finally:
77         cursor.close()
78
79
80
81 #Insert Order
82 CourierServiceDb.insertOrder(10,"Rohan","Jalgaon","Rohit","mumbai",5,'delivered',"9887656","2023-04-05")
83
84 #Update Courier Status
85 CourierServiceDb.updateCourierStatus("78789802", newStatus="In Transit")
86
87 # Get Delivery History
88 delivery_history = CourierServiceDb.getDeliveryHistory("78789802")
89 print("Delivery History:", delivery_history)

```

Output:

```
PS D:\Hexaware\Assignment\assignment_4\python> & "C:/Program Files/Python32/python.exe" "d:/hexaware/Assignment/assignment_4/python/database.py"
Connected to the database
Order inserted successfully.

Inserted
Courier status updated successfully.

updated

Delivery History: {(5, 'Kavita Patel', '777 Oak Street, Delhi', 'Vinod Iyer', '888 Maple Avenue, Mumbai', Decimal('0.90'), 'In transit', 'IN090802', datetime.date(2023, 4, 6))}

PS D:\Hexaware\Assignment\assignment_4\python>
```

Database changes:

SQL File 7

```
1. use couriermanagementsystem;
2. show tables;
3. select * from couriers;
4.
```

Result Grid

CourierID	SenderName	SenderAddress	ReceiverName	ReceiverAddress	Weight	Status	TrackingNumber	DeliveryDate
1	Raj Sharma	789 Pine Street, Delhi	Suresh Kumar	367 Elm Street, Mumbai	8.50	In Transit	TM123456	2023-04-01
2	Priya Patel	523 Oak Street, Kolkata	Anita Gupta	456 Maple Street, Bangalore	6.00	Delivered	TM789012	2023-03-28
3	Jyoti Singh	112 Pine Street, Hyderabad	Alok Gupta	322 Oak Street, Chennai	7.00	In Transit	TM654321	2023-04-05
4	Arun Verma	805 Maple Avenue, Pune	Shweta Reddy	606 Elm Street, Ahmedabad	4.50	Delivered	TM987654	2023-04-02
5	Kavita Yadav	777 Oak Street, Delhi	Vinod Iyer	888 Maple Avenue, Mumbai	9.20	In Transit	TM290002	2023-04-08
6	Rohit Deshmukh	999 Pine Street, Kolkata	Sneha Kapoor	111 Elm Street, Bangalore	6.80	Scheduled	TM345678	2023-04-30
7	Raj Sharma	303 Oak Street, Hyderabad	Suman Mehta	404 Maple Avenue, Pune	5.30	Delivered	TM012345	2023-03-30
8	Sangeeta Iyer	202 Elm Street, Chennai	Vinod Desai	303 Pine Street, Ahmedabad	8.00	In Transit	TM678901	2023-04-12
10	Rohan	Jalgaon	Rohit	Mumbai	5.30	delivered	TM087656	2023-04-03

SQL File 7

```
1. use couriermanagementsystem;
2. show tables;
3. select * from courierservices;
4.
```

Result Grid

ServiceID	ServiceName	Cost
1	Express	15.00
2	Next Day Delivery	30.00
3	Two-Day Delivery	18.00
4	Economy	12.00
5	Same Day Delivery	25.00
6	Standard	10.00
7	Express	22.00
11	Delivery Driver	30.00

SQL File 7

```
1. use couriermanagementsystem;
2. show tables;
3. select * from employees;
4.
```

Result Grid

EmployeeID	Name	Email	ContactNumber	Role	Salary
1	Rohan Gupta	rohan.gupta@gmail.com	111-222-3333	Delivery Driver	50000.00
2	Chitra Iyer	chitra.iyer@gmail.com	444-333-5555	Customer Service	45000.00
3	Vikram Singh	vikram.singh@gmail.com	777-888-1111	Warehouse Manager	55000.00
4	Pooja Desai	pooja.desai@gmail.com	444-333-5555	Dispatcher	48000.00
5	Kunal Kapoor	kunal.kapoor@gmail.com	111-222-3333	Delivery Driver	52000.00
6	Aarti Singh	aarti.singh@gmail.com	888-999-7777	Customer Service	47000.00
7	Rajesh Mehta	rajesh.mehta@gmail.com	888-999-1111	Warehouse Staff	49000.00
8	Ananya Iyer	ananya.iyer@gmail.com	222-333-4444	Dispatcher	46000.00

SQL File 1

```

1* use couriermanagementsystem;
2* show tables;
3* select * from locations;
4

```

Result Grid

LocationID	LocationName	Address
1	Main Office	123 Main Street, Delhi
2	Branch Office	456 Oak Avenue, Mumbai
3	Warehouse A	777 Pine Street, Hyderabad
4	Warehouse B	888 Oak Street, Pune
5	Branch Office 2	999 Elm Street, Bangalore
6	Branch Office 3	331 Maple Avenue, Chennai
7	Warehouse C	222 Oak Street, Kolkata
8	Branch Office 4	333 Elm Street, Ahmedabad

SQL File 2

```

1* use couriermanagementsystem;
2* show tables;
3* select * from payments;
4

```

Result Grid

PaymentID	CourierID	LocationID	Amount	PaymentDate
1	1	1	20.00	2023-03-25
2	2	2	30.00	2023-04-02
3	3	3	15.00	2023-04-01
4	4	4	25.00	2023-04-02
5	5	5	18.00	2023-04-03
6	6	6	22.00	2023-04-04
7	7	7	30.00	2023-04-05
8	8	8	20.00	2023-04-06

SQL File 3

```

1* use couriermanagementsystem;
2* show tables;
3* select * from users;
4

```

Result Grid

UserID	Name	Email	Password	ContactNumber	Address
1	Raj Sharma	raj.sharma@gmail.com	password123	9976543210	123 Main Street, Delhi
2	Priya Patel	priya.patel@gmail.com	pass456	7890123456	456 Oak Avenue, Mumbai
3	Amit Verma	amit.verma@gmail.com	amit123	4321098765	789 Elm Street, Bangalore
4	Neha Reddy	neha.reddy@gmail.com	neha456	7654321098	101 Pine Avenue, Hyderabad
5	Rahul Kapoor	rahul.kapoor@gmail.com	rahul347	4567890123	202 Oak Street, Kolkata
6	Arjun Desai	arjun.desai@gmail.com	arjun456	1432109876	303 Maple Avenue, Chennai
7	Sanjay Mehta	sanjay.mehta@gmail.com	sanjay789	4321098765	404 Elm Street, Pune
8	Meera Iyer	meera.iyer@gmail.com	meera867	3210987654	505 Pine Avenue, Ahmedabad