Test Cases

Testing The Graph Class

```
    # Testing Vertex Class
    v1 = _DSAGraphVertex("Rockingham")
    print("Testing vertex A creation: " + str(v1))
    print()

    v2 = _DSAGraphVertex("Perth")

    # Testing Edges Class
    e1 = _DSAGraphEdge(v1, v2, "Read St", 50)
    print("Testing 'Read St' Rockingham to Perth road creation\n" + str(e1))
    print()

    # Testing Graph Vertex
    g1 = DSAGraph()
    g1.addVertex("Mandurah")
    g1.addVertex("Fremantle")
    g1.addVertex("Serpentine")
    g1.addVertex("Rockingham")
    g1.addVertex("Perth")
    g1.addVertex("Midland")
    g1.addVertex("Subiaco")
    g1.addVertex("Z")
    print("Testing Graph Mandurah, Fremantle, Serpentine... creation: ", end= '')
    g1.vertices.printList()

    # Testing Get Vertex
    print("Testing find Mandurah: " + str(g1.getVertex("Mandurah")))
    print()

    # Testing Graph Edges
    print("Testing Road add Mandurah, Fremantle: ")
    g1.addEdge("Mandurah","Fremantle","Stock Rd", 80)
    g1.addEdge("Fremantle","Perth", "Ocean Beach Rd", 40)
    g1.addEdge("Perth","Rockingham", "Southern Fwy", 100)
    g1.addEdge("Rockingham","Serpentine", "Mundijong Rd", 30)
    g1.addEdge("Perth","Midland","Northern Fwy", 30)
    g1.addEdge("Fremantle","Z","Narnia Rd",1000)

    # Testing Graph Display function
    print("Testing Graph Display function")
    g1.displayAsList()

    # Testing Retrieve Neighbours
    print("Testing Retrieve Neighbours")
    g1.getAdjacent("Perth").printList()

    #Testing is_path function
    print("Testing is_path function")
    print(g1.is_path("Mandurah","Fremantle"))
    print(g1.is_path("Fremantle","Mandurah"))
    print(g1.is_path("Subiaco","Fremantle"))
    print(g1.is_path("Fremantle","Subiaco"))
    print(g1.is_path("Mandurah","Perth"))
    print(g1.is_path("Z","Fremantle"))
```

Output

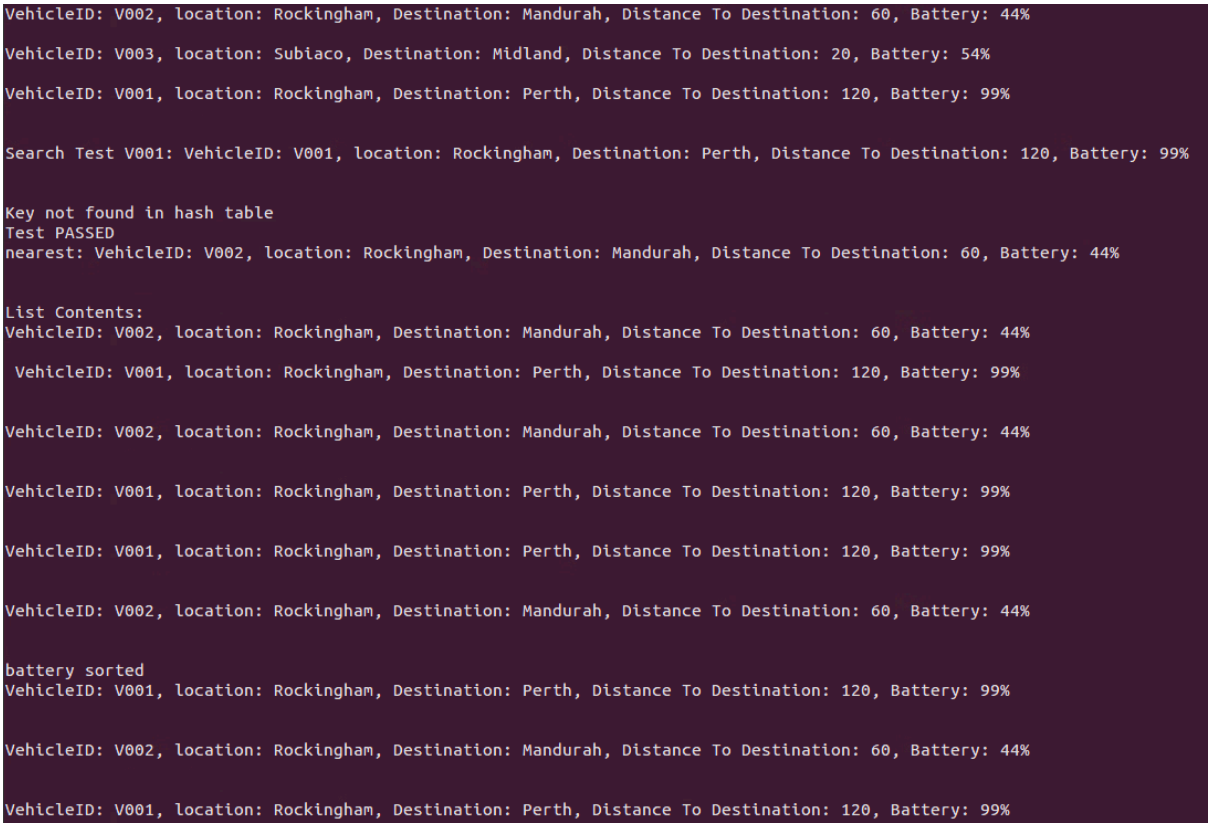
```
Testing vertex A creation: Label: Rockingham, Vertex Visited State: False

Testing 'Read St' Rockingham to Perth road creation
Location 1: Label: Rockingham, Vertex Visited State: False
Location 2: Label: Perth, Vertex Visited State: False
Edge Visited State: False
Road Name: Read St
Road Distance: 50

Testing Graph Mandurah, Fremantle, Serpentine... creation: List Contents:
Label: Mandurah, Vertex Visited State: False Label: Fremantle, Vertex Visited State: False
 Label: Serpentine, Vertex Visited State: False Label: Rockingham, Vertex Visited State: F
alse Label: Perth, Vertex Visited State: False Label: Midland, Vertex Visited State: False
 Label: Subiaco, Vertex Visited State: False Label: Z, Vertex Visited State: False
Testing find Mandurah: Label: Mandurah, Vertex Visited State: False

Testing Road add Mandurah, Fremantle:
Testing Graph Display function

List of Locations and there neighboring places:

Location 1
Fremantle:
  Adjacent City: Mandurah
    Connecting Road: Stock Rd
    Distance: 80

  Adjacent City: Perth
    Connecting Road: Ocean Beach Rd
    Distance: 40

  Adjacent City: Z
    Connecting Road: Narnia Rd
    Distance: 1000


Location 2
Mandurah:
  Adjacent City: Fremantle
    Connecting Road: Stock Rd
    Distance: 80


Location 3
Midland:
  Adjacent City: Perth
    Connecting Road: Northern Fwy
    Distance: 30
```

```
Location 4
Perth:
  Adjacent City: Fremantle
    Connecting Road: Ocean Beach Rd
    Distance: 40

  Adjacent City: Midland
    Connecting Road: Northern Fwy
    Distance: 30

  Adjacent City: Rockingham
    Connecting Road: Southern Fwy
    Distance: 100


Location 5
Rockingham:
  Adjacent City: Perth
    Connecting Road: Southern Fwy
    Distance: 100

  Adjacent City: Serpentine
    Connecting Road: Mundijong Rd
    Distance: 30


Location 6
Serpentine:
  Adjacent City: Rockingham
    Connecting Road: Mundijong Rd
    Distance: 30


Location 7
Subiaco:
No adjacent Towns

Location 8
Z:
  Adjacent City: Fremantle
    Connecting Road: Narnia Rd
    Distance: 1000


Testing Retrieve Neighbours
List Contents:
Label: Fremantle, Vertex Visited State: False Label: Midland, Vertex Visited State: False
Label: Rockingham, Vertex Visited State: False
Testing is_path function
True
True
False
False
True
True
```

Testing the Vehicle Class, Hash Table and sorting

```
g1 = DSAGraph()
g1.addVertex("Mandurah")
g1.addVertex("Fremantle")
g1.addVertex("Serpentine")
g1.addVertex("Rockingham")
g1.addVertex("Perth")
g1.addVertex("Midland")
g1.addVertex("Subiaco")
g1.addEdge("Mandurah","Fremantle","Stock Rd", 80)
g1.addEdge("Fremantle","Perth", "Ocean Beach Rd", 40)
g1.addEdge("Perth","Rockingham", "Southern Fwy", 100)
g1.addEdge("Rockingham","Serpentine", "Mundijong Rd", 30)
g1.addEdge("Perth","Midland","Northern Fwy", 30)

car1 = Vehicle("V001","Rockingham","Perth",120,99,g1)
car2 = Vehicle("V002","Rockingham","Mandurah",60,44,g1)
car3 = Vehicle("V003","Subiaco","Midland",20,54,g1)

vTable = VehicleHashTable(5)
vTable.insert(car1)
vTable.insert(car2)
vTable.insert(car3)
vTable.printVehicles()
print("\nSearch Test V001: " + str(vTable.search("V001")))
vTable.delete("V003")
try:
    print("\nSearch Test V003: " + str(vTable.search("V003")))
except KeyNotFoundError as err:
    print(err)
    print("Test PASSED")

vehicleList = vTable.getVehicleList()

nearest = find_nearest_vehicle(vehicleList)
print("nearest: " + str(nearest))
vehicleList.printList()
sortedList = heapSortDistanceTo(vehicleList)
for vehicle in sortedList:
    print(vehicle)
ascSortedList = heapSortDistanceAsc(vehicleList)
for vehicle in ascSortedList:
    print(vehicle)

batterySorted = quickSortBattery(vehicleList)
print("battery sorted")
for vehicle in batterySorted:
    print(vehicle)

highestBattery = find_vehicle_with_highest_battery(vehicleList)
print(highestBattery)
```

Output:



Other tests (Manual through Menu)

Add Location	PASSED
Delete Location	PASSED
Deleting non existing location notification	PASSED
Add Road	PASSED
Notified- Add roads between non existing locations	PASSED
Notified - Add roads with non integer lengths	PASSED
Notfied - add roads that already exists between two points	PASSED
Delete road	PASSED
Notified- delete roads that dont exist	PASSED
Display Network	PASSED
Check Path between connected	PASSED
Check Path between unconnected	PASSED
Add Vehicles	PASSED

Notitfied - Add vehicles between non existent locations	PASSED
Notified - Add vehicles with higher than 100 battery	PASSED
Notified - Add vehicles with higher than 0 battery	PASSED
Find Nearest Vehicle	PASSED
Find Highest Battery	PASSED
Display Vehicles sorted by distance To	PASSED
Dispaly Vehicles by battery level	PASSED