```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java selection_sort
****SELECTION SORT ALGORITHM****
Enter the number of elements to be sorted
5
Array elements to be sorted are:
60 52 44 81 0
The sorted elements are:
0 44 52 60 81
The time taken to sort is: 6100ns
********************************
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java quicksort
****QUICK SORT ALGORITHM****
Enter the number of elements to be sorted
4
Array elements to be sorted are:
47 26 20 88
The sorted elements are:
20 26 47 88
The time taken to sort is 4100ns
********************************
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java mergesort
****MERGE SORT ALGORITHM****
Enter the number of elements to be sorted
5
Array elements to be sorted are:
30 63 95 92 91
The sorted elements are:
30 63 91 92 95
The time taken to sort is 307000 ns
********************************
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java AllPairShortestPath
The following matrix shows the shortest distances between every pair of vertices
0 5 8 9
INF 0 3 4
INF INF 0 1
INF INF INF 0
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java GFG
220
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java SubsetSum
Subsets with sum equal to 9:
[1, 2, 6]
[1, 8]
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java fractional_knapsack
*** KNAPSACK PROBLEM-GREEDY METHOD***
Enter the number of items in the store:
7
Enter the (weight and profit) of items:
2       10
3       5
5       15
7       7
1       6
4       18
1       3
Enter the capacity of the knapsack:
15

Items Selected Fraction Selected(0/1/Partial)
*********************************************
        1               1.0
        2               0.6666666666666666
        3               1.0
        4               0.0
        5               1.0
        6               1.0
        7               1.0

Max Profit = 55.333333333333336, Max Weight = 15.0
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java Dijkstra
********* DIJKSTRA'S ALGORITHM *********
Enter no. of nodes :
5
Enter cost adjacency matrix :
0       3       999     7       999
3       0       4       2       999
999     4       0       5       6
7       2       5       0       4
999     999     6       4       0
Enter source vertex :
1
The shortest path and distance is shown below:
DEST VERTEX<-(Intermediate vertices)<-SOURCE=DISTANCE
1<-1=0
2<-1=3
3<-2<-1=7
4<-2<-1=5
5<-4<-2<-1=9
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java TSE
The cost of most efficient tour = 80
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java Kruskal_algo
********* KRUSKAL'S ALGORITHM *********
Enter number of nodes
6
Enter the cost adjacency matrix
0       5       10      30      10      999
5       0       999     999     30      8
10      999     0       20      999     999
30      999     20      0       20      999
10      30      999     20      0       2
999     8       999     999     2       0
The min cost spanning tree with edges is
*******************
Edge    Weight
*******************
5->6    2
1->2    5
2->6    8
1->3    10
3->4    20
Cost of the Spanning tree=45
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java Prims_algo
********* PRIMS ALGORITHM *********
Enter number of nodes
6
Enter the cost adacency matrix
0       5       10      30      10      999
5       0       999     999     30      8
10      999     0       20      999     999
30      999     20      0       20      999
10      30      999     20      0       2
999     8       999     999     2       0
The min cost spanning tree with edges is
*******************
Edge    Weight
*******************
1->2    5
2->6    8
6->5    2
1->3    10
3->4    20
************************
cost of spanning tree=45
************************
```

```
C:\Users\Admin\Desktop\1JS21AI018\DAA_LAB>java HamiltonianCycles
Hamiltonian Cycles:
[0, 1, 2, 4, 3]
[0, 3, 4, 2, 1]
```