Name: *Krithika Swaminathan*
Reg. no.: *205001057*

DAA Lab – Assignment 6

Greedy technique

1. To Implement Knapsack Algorithm.

Code:

```
# Python program to implement the Knapsack problem

def knapSack(W, wt, val, n):
      K = [[0 for x in range(W + 1)] for x in range(n + 1)]

      for i in range(n + 1):
            for w in range(W + 1):
                  if i == 0 or w == 0:
                        K[i][w] = 0
                  elif wt[i-1] <= w:
                        K[i][w] = max(val[i-1]
                                    + K[i-1][w-wt[i-1]],
                                          K[i-1][w])
                  else:
                        K[i][w] = K[i-1][w]

      return K[n][W]

val = list(map(int,input("Enter values of items: ").split()))
wt = list(map(int,input("Enter weights of items: ").split()))
W = int(input("Enter max weight: "))
print("Max value:",knapSack(W, wt, val, len(val)))
```

Output:

```
~/DAA-Exercise6$ python3 knapsack.py
Enter values of items: 60 100 120
Enter weights of items: 10 20 30
Enter max weight: 50
Max value: 220
~/DAA-Exercise6$ python3 knapsack.py
Enter values of items: 30 90 70
Enter weights of items: 5 10 8
Enter max weight: 20
Max value: 160
~/DAA-Exercise6$
```

Name: *Krithika Swaminathan*
Reg. no.: *205001057*

2. To Implement Prim's Algorithm for MST.

Code:

```
# Python program to implement Prim's algorithm for MST

import sys

class Graph():
      def __init__(self, vertices):
            self.V = vertices
            self.graph = [[0 for column in range(vertices)]
                              for row in range(vertices)]

      def printMST(self, parent):
            print("\tMST:\nEdge \tWeight")
            cost = 0
            for i in range(1, self.V):
                  print (parent[i], "-", i, "\t", self.graph[i][parent[i]])
                  cost += self.graph[i][parent[i]]
            print("\nTotal Cost:",cost)
      def minKey(self, key, mstSet):
            min = sys.maxsize

            for v in range(self.V):
                  if key[v] < min and mstSet[v] == False:
                        min = key[v]
                        min_index = v

            return min_index

      def primMST(self):
            key = [sys.maxsize] * self.V
            parent = [None] * self.V
            key[0] = 0
            mstSet = [False] * self.V

            parent[0] = -1

            for vertex in range(self.V):
                  u = self.minKey(key, mstSet)
                  mstSet[u] = True

                  for v in range(self.V):
                        if self.graph[u][v] > 0 and mstSet[v] == False and
key[v] > self.graph[u][v]:
                                    key[v] = self.graph[u][v]
                                    parent[v] = u

            self.printMST(parent)

g = Graph(5)
g.graph = [ [0, 2, 0, 6, 0],
                  [2, 0, 3, 8, 5],
```

```
                  [0, 3, 0, 0, 7],
                  [6, 8, 0, 0, 9],
                  [0, 5, 7, 9, 0]]
g.primMST();
```

Output:

```
~/DAA-Exercise6$ python3 prims.py
    MST:
Edge    Weight
0 - 1    2
1 - 2    3
0 - 3    6
1 - 4    5

Total Cost: 16
~/DAA-Exercise6$
```