1. Write a program to solve the 0/1 Knapsack problem using greedy Algorithm.

```
import java.util.Arrays;

import java.util.Comparator;

class Item {
    int value, weight;
  Public Item(int value, int weight){
      this. value = value;
      this. weight = weight;
  }
}
Public class KnapSack {
   Public static void main(String[] args){
     int capacity = 50;
     item[] items ={new Item(60, 10),
                    new Item(100, 20),
                    new Item(120, 30),};

   double value = KnapSack(items, capacity);
   System.out.println("Maximum KnapSack value"+maxValue);
 }
Public Static ~~void~~ double KnapSack(Item[] items, int capacity){
  Arrays.Sort(items, Comparator<Item>(){
     @override
   Public int compare(Item item1, item 2){
     double ratio1 = (double) item1 . value / item1.weight;
  double ratio2 =(double) item2, value/ item2 . weight;
```

```
    return Double.compare(ratio2, ratio1);
   }
  });
  int currentWeight = 0;
    double totalvalue = 0;
  for (Item item: items){
    if (currentWeight + item.weight <= capacity){
      currentWeight += item.weight;
      total value  += item.value;
    }else{
      double remainingCapacity = capacity - currentWeight;
    totalvalue = (remainingcapacity / item.weight) * item.value;
      break;
    }
  }
  return total value;
 }
}
```

Output :-

Maximum value in Knapsack = 240.0

2. Write a program to solve Fractional Knapsack Problem using Greedy approach.

```
import Java.util.Arrays;
import Java.util.Comparator;
Public class Fractional knapsack {.
  Static class Item{.
      int profit, weight;
    Public Item(int profit, int weight){
      this.profit = profit;
      this.weight = weight;

    }
  }
  Public static double getMaxValue(Item[] items, int capacity){.
   Arrays.sort (items, (item1, item2)-> Double.compare((double)
    item2.profit / item2.weight, (double) item1.profit/item1.weight));
   double totalvalue =0;
   for (Item item : items){
     int currentweight = item.weight;
     int currentvalue = item.profit;
     if ( capacity - currentweight >=0){
       capacity -= currentweight;
       totalvalue += currentvalue;
     }else{
     double fraction = (double) capacity/currentweight;
     totalvalue += (currentvalue * fraction);
     capacity = 0;
      break;
    }
   }
  return totalvalue.
  }
Public static void main(String[] args){.
    Item[] items = {new Item(60, 10),
                    new Item(10, 20),
                    new Item (20, 30)};
    int capacity =50)
  double max value = get.Maxvalue (items; capacity));

  System.out.println("Maximom value in Knapsack ="+ maxvalue);
  }
}
```

output:-

```
  Maximum value in Knapsack = 85.0
```

3. Write a program to solve job Sequencing deadline using greedy Algorithm.

```
import java.util.*;
class Job{
    char id;
    int deadline, profit;
  Public Job() {}
 Public Job (char id, int deadline, int profit) {
   this. id = id;
   this.deadline = deadline;
   this. profit = profit;
  }
void printJobScheduling (ArrayList <Job> arr, int t) {
  int n=arr.size();
  Colections.Sort (arr, (a,b) -> b.Profit - a.profit);
  boolean result[] = new boolean[t];
  char job[] = new char[t];
 for( int i=0; i<n ; i++ ){
    for( int j = Math.min(t-1, arr.get(i).deadine-1);
      j>=0; j--){
        if (result [j] == false) {
            result [j] = true;
            Job [j] = arr.get(i).id;
            break;
        }
    }
  }
for (char jb : job)
        SOP ( jb + " ");
  SOP ();  }
Public static void main (String args[]) {
   ArrayList<Job> arr = new ArrayList<Job>();
   arr.add (new Job ('a', 2, 100));
   arr.add ( new Job ('b', 1, 19));
   arr.add ( new Job ('c', 2, 27));
   arr.add ( new Job ('d', 1, 25));
   arr.add ( new Job ('e', 3, 15));
   arr.add
  SOP (" Following is Maximum profit Sequence of Jobs");
  Job job = new Job();
  job. Print JobScheduling (arr, 3);
 }
}
```

output :-

Following is Maximum profit Sequence of Jobs

c a e.

4. write a program to find Minimum Spaning tree using prims Algorithm.

```
import. Java.IO.*;
import.Java.lang.*;
import.Java.util.*;
class MST {.
  Private static final int V=5; //No-of vertices
 int minKey (int key[], Boolean mstSet[]){
 int min = Integer.MAX_VALUE, min_idex = -1;
 for (int v=0; v<V; v++)
     if (mstSet[v] == false && key[v] < min) {
         min = key[v];
         min_index = v;
 return min_index;
}
void printMST (int parent[], int graph[][]){
  SOP (" Edge \t weight");
 For (int i=1; i<V; i++)
     SOP (Parent[i]+" - "+i+" \t "+graph[i][Parent[i]]);
 }
void primMST(int graph[][]){
   int parent[] = new int[V];
   int key[] = new int[V];
  Boolean mstSet[] = new Boolean[V];
  for (int i=0; i<V; i++){
     Key[i] = Integer.MAX_VALUE;
     mstSet[i] = False;
  }
     Key[0] = 0;
     parent[0] = -1;
 For (int count = 0; count<V-1; count++){.
```

```
        int u = minKey( Key, mstSet);
        mstSet [u] = true;
      for( int v = 0; v < V ; v++)
         if (graph[u][v] != 0 && mstSet[v] == false && graph[u][v]
               < key [v]) {
           Parent [v] = u;
           key [v] = graph[u][v];
         }
      Print MST ( parent, graph);
   }
   Public static void main (String [] args) {
      MST t = new MST ();
      int graph[] [] = new int[][] { {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}};

      t. primMST (graph);
   }
 }.
```

output :-

```
            weight
  Edge
               2
  0 - 1
               3
  1 - 2
               6
  0 - 3
               5
  1 - 4
```