Aim:

Using greedy approach for algorithm design, write a program to implement Subset Cover Problem.

import java.util.*;

Code:

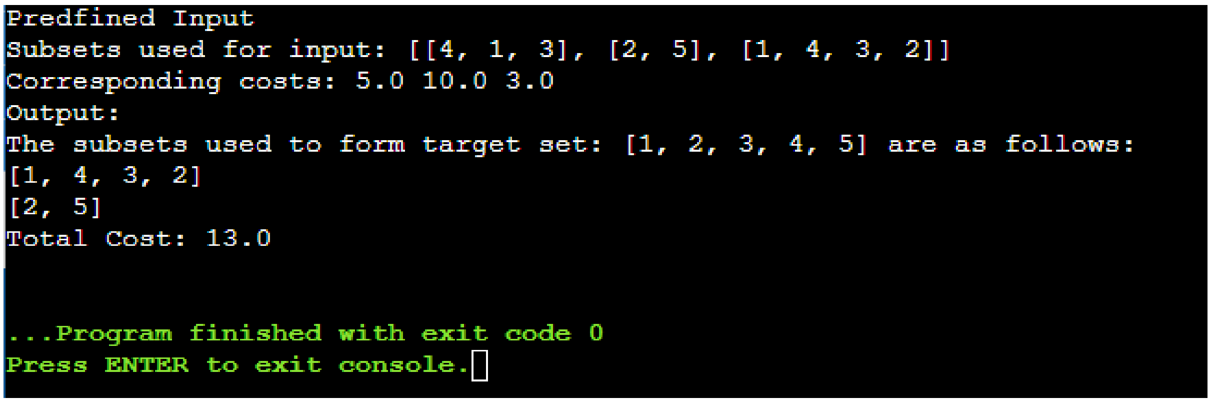
```
public class Main {
    public static double totalcost = 0;
    public static Object[] solarray = new Object[5];
    static int k = 0;
    public static List<Set<Object>> sets1 = new ArrayList<Set<Object>>();

    public static int minIndex(Double[] array) {
        if (array.length == 0)
        {
            return -1;
        }
        int index = 0;
        double min = array[index];
        for (int i = 1; i < array.length; i++) {
            if (array[i] <= min) {
                min = array[i];
                index = i;
            }
        }
        return index;
    }

    public static Set<Object> check(List<Set<Object>> set1, Double[] cost1, Set<Object> initial) {
        Double[] store = new Double[set1.size()];
        int i = 0;
        for (Set<Object> temp : set1) {
            temp.removeAll(initial);
            store[i] = cost1[i] / temp.size();
            i++;
            temp.addAll(initial);
        }
        int idx = minIndex(store);
        totalcost += cost1[idx];
        solarray[k++] = sets1.toArray()[idx];
        System.out.println(solarray[k - 1]);
        for (Set<Object> temp : set1) {
            if (set1.toArray()[idx] == temp) {
                for (Object o : temp) {
                    initial.add(o);
                }
            }
        }
        return initial;
    }

    public static void main(String[] args) {
        Object[][] arrayOfSets = { { 4, 1, 3 }, { 2, 5 }, { 1, 4, 3, 2 }, };
        Double[] costs = { 5.0, 10.0, 3.0 };
        Object[] target = { 1, 2, 3, 4, 5 };
        Object[][] init = {};
        System.out.print("Predfined Input\n");
        List<Set<Object>> sets = new ArrayList<Set<Object>>();
        for (Object[] element : arrayOfSets) {
            sets.add(new LinkedHashSet<Object>(Arrays.asList(element)));
        }
        for (Object[] element : arrayOfSets) {
            sets1.add(new LinkedHashSet<Object>(Arrays.asList(element)));
        }
        final Set<Object> solution = new LinkedHashSet<Object>(Arrays.asList(target));
        Set<Object> initial1 = new HashSet<Object>(Arrays.asList(init));
        System.out.println("Subsets used for input: "+ sets);
        System.out.print("Corresponding costs:");
        for(int i=0;i<costs.length;i++){
            System.out.print(" "+costs[i]);
        }
        System.out.println("\nOutput:");
        System.out.println("The subsets used to form target set: "+solution+" are as follows: ");
        while (!initial1.equals(solution)) {
            initial1 = check(sets, costs, initial1);
        }
        System.out.println("Total Cost: "+totalcost);
    }
}
```

Output:



```
Predfined Input
Subsets used for input: [[4, 1, 3], [2, 5], [1, 4, 3, 2]]
Corresponding costs: 5.0 10.0 3.0
Output:
The subsets used to form target set: [1, 2, 3, 4, 5] are as follows:
[1, 4, 3, 2]
[2, 5]
Total Cost: 13.0


...Program finished with exit code 0
Press ENTER to exit console.
```