组装最大可靠性设备

题目描述：一个设备由N种类型元器件组成（每种类型元器件只需要一个，类型type编号从0~N-1），每个元器件均有可靠性属性reliability，可靠性越高的器件其价格price越贵。而设备的可靠性由组成设备的所有器件中可靠性最低的器件决定。
给定预算S，购买N种元器件（每种类型元器件都需要购买一个），在不超过预算的情况下，请给出能够组成的设备的最大可靠性。

输入描述：S N // S总的预算，N元器件的种类
total // 元器件的总数，每种型号的元器件可以有多种；此后有total行具体器件的数据
type reliability price // type 整数类型，代表元器件的类型编号从0 ~ N-1；reliability 整数类型，代表元器件的可靠性；price 整数类型，代表元器件的价格

输出描述：符合预算的设备的最大可靠性，如果预算无法买齐N种器件，则返回 -1

补充说明：0 <= S,price <= 10000000;
0 <= N <= 100;
0 <= type <= N-1;
0 <= total <= 100000;
0 < reliability <= 100000;

示例 展开

示例1
输入：500 3
6
0 80 100
0 90 200
1 50 50
1 70 210
2 50 100
2 60 150
输出：60
说明：预算500，设备需要3种元件组成，方案 类型0的第一个(可靠性80), 类型1的第二个(可靠性70), 类型2的第二个(可靠性60) 可以使设备的可靠性最大 60

示例2
输入：100 1
1
0 90 200
输出：-1
说明：组成设备需要1个元件，但是元件价格大于预算，因此无法组成设备，返回-1

```java
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    private static class SomeThing {
        private int type;
        private int price;
        private int relia;

        public SomeThing() {
        }

        public SomeThing(int a, int b, int c) {
            this.type = a;
            this.price = b;
            this.relia = c;
        }

        public int getType() {
            return type;
        }

        public int getPrice() {
            return price;
        }

        public int getRelia() {
            return relia;
        }
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int m = in.nextInt();
        int n = in.nextInt();
        int inputSize = in.nextInt();
        List<SomeThing> someThings = new ArrayList<>();
        for (int i = 0; i < inputSize; i++) {
```

```
            int a = in.nextInt();
            int b = in.nextInt();
            int c = in.nextInt();
            if (c > m) continue;
            SomeThing temp = new SomeThing(a, c, b);
            someThings.add(temp);
        }
        List<SomeThing>[] bests = new ArrayList[n];
        for (int i = 0; i < n; i++) {
            bests[i] = checkInput(someThings, i);
        }
        List<SomeThing> result = bests[0];
        if (result.size() == 0) {
            System.out.println(-1);
            return;
        }
        if (n > 1) {
            for (int i = 1; i < n; i++) {
                if (result.size() == 0) {
                    System.out.println(-1);
                    return;
                }
                List<SomeThing> temp = new ArrayList<>();
                for (SomeThing s : result) {
                    for (SomeThing s2 : bests[i]) {
                        if (s.getPrice() + s2.getPrice() <= m) {
                            SomeThing someThing = new SomeThing(i, s.getPrice() + s2.getPrice(), Math.min(s.getRelia(), s2.getRelia()));
                            temp.add(someThing);
                        }
                    }
                }
                result = checkResult(temp);
            }
        }
        System.out.println(getBest(result, n - 1));
    }

    private static int getBest(List<SomeThing> someThings, int type) {
        int max = 0;
        for (SomeThing s : someThings) {
            if (s.type == type) {
                if (s.relia > max) {
                    max = s.relia;
                }
            }
        }
        return max;
    }

    private static List<SomeThing> checkInput(List<SomeThing> someThings, int n) {
        List<SomeThing> result = new ArrayList<>();
        for (int i = 0; i < someThings.size(); i++) {
            if(someThings.get(i).getType() == n){
                result.add(someThings.get(i));
            }
        }
        return result;
    }

    private static List<SomeThing> checkResult(List<SomeThing> someThings) {
        List<SomeThing> result = new ArrayList<>();
        List<SomeThing> temp = someThings;
        for (int i = 0; i < someThings.size(); i++) {
            boolean flag = true;
            for (int j = 0; j < temp.size() && i != j; j++) {
                if (someThings.get(i).getPrice() >= temp.get(j).getPrice() && someThings.get(i).getRelia() <= temp.get(j).getRelia()) {
                    flag = false;
                    break;
                }

            }
            if(flag){
                result.add(someThings.get(i));
            }
        }
        return result;
    }
}
```