```
import java.util.*;
import java.util.Scanner;
// A class to store a Job
class Job
{
   String start, finish;
   int profit;

   Job(String start, String finish, int profit)
   {
      this.start = start;
      this.finish = finish;
      this.profit = profit;
   }

   @Override
   public String toString() {
      return "(" + this.start + ", " + this.finish + ", " + this.profit + ") ";
   }
}

class Main
{
   // Function to perform a binary search on the given jobs, which are sorted
   // by finish time. The function returns the index of the last job, which
   // doesn't conflict with the given job, i.e., whose finish time is
   // less than or equal to the given job's start time.
   public static int findLastNonConflictingJob(List<Job> jobs, int n)
   {
```

```
    // search space
    int low = 0;
    int high = n;

    // iterate till the search space is exhausted
    while (low <= high)
    {
        int mid = (low + high) / 2;
        int cmp = jobs.get(mid).finish.compareTo(jobs.get(n).start);

        if (cmp <= 0)
        {
            int cmp1 = jobs.get(mid+1).finish.compareTo(jobs.get(n).start);
            if (cmp1 <= 0) {
                low = mid + 1;
            }
            else {
                return mid;
            }
        }
        else {
            high = mid - 1;
        }
    }

    return -1;
}

// Function to print remaining jobs and calculate remaining earnings
```

```
//excluding the non-overlapping max profits tasks picked by lokesh
public static void findRemainingEarningAndJobs(List<Job> jobs)
{
   // sort jobs in increasing order of their finish times
   Collections.sort(jobs, Comparator.comparing(x -> x.finish));

   // get the number of jobs
   int n = jobs.size();

   // base case
   if (n == 0) {
      return;
   }

   int[] remainingEarnings = new int[n];
   int totalProfit = 0;
   List<List<Integer>> tasks = new ArrayList<>();
   for (int i = 0; i < n; i++) {
      tasks.add(new ArrayList<>());
      totalProfit = totalProfit + jobs.get(i).profit;
   }

   // initialize `remainingEarnings[0]` and `tasks[0]` with the first job
   remainingEarnings[0] = jobs.get(0).profit;
   tasks.get(0).add(0);


   for (int i = 1; i < n; i++)
   {
```

```
        // find the index of the last non-conflicting job with the current job
        int index = findLastNonConflictingJob(jobs, i);

        // include the current job with its non-conflicting jobs
        int currentProfit = jobs.get(i).profit;

        if (index != -1) {
            currentProfit += remainingEarnings[index];
        }

        // if including the current job leads to the maximum profit so far
        if (remainingEarnings[i-1] < currentProfit)
        {
            remainingEarnings[i] = currentProfit;

            if (index != -1) {
                tasks.set(i, new ArrayList<>(tasks.get(index)));
            }
            //tasks.set(i, new ArrayList<>(tasks.get(i-1)));
tasks.get(i).add(i);
        }

        // excluding the current job leads to the maximum profit so far
        else {
            //tasks.get(i).add(i);
tasks.set(i, new ArrayList<>(tasks.get(i-1)));
            remainingEarnings[i] = remainingEarnings[i-1];
        }
    }
```

```
    var remainingJobs = n - tasks.get(n-1).size();
    var remainingProfit = totalProfit - remainingEarnings[n-1];
    System.out.println("The number of tasks and earnings available for others");

    System.out.println("Task " + remainingJobs);
    System.out.println("Earnings " + remainingProfit);

}

public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);
    System.out.println("Enter the number of Jobs ");
    int jobCount = sc.nextInt();

    if(jobCount>100)
    {
        System.out.println("The number of jobs in the day should be less than 100");
        return;
    }

    List<Job> jobs =  new ArrayList<>();
    System.out.println("Enter job start time, end time, and earnings");


    for (int i = 0; i<jobCount;i++) {

        List<String> inputs = new ArrayList<>();
```

```
    int a = 0;

    while(a<3)
    {
        inputs.add(sc.next());
        a++;
    }


    String start = inputs.get(0);
    String finish = inputs.get(1);
    int profit = Integer.parseInt(inputs.get(2));

    int compareInputTime = start.compareTo(finish);
    if(compareInputTime>=0)
    {
        System.out.println("Finish time cannot be earlier or same as Start time");
        return;
    }
    System.out.println("Enter next job start time, end time and earnings");
    jobs.add(new Job( start, finish, profit ));
    sc.nextLine();

}
 sc.close();
// List<Job> jobs = Arrays.asList(

//          new Job( "0900", "1030", 100 ),
//        new Job( "1000", "1200", 500 ),
```

```
        //        new Job( "1100", "1200", 300 ));


        findRemainingEarningAndJobs(jobs);
    }
}
```