Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Subject : Competitive Programmimg Lab

Name : Jaykishan Natwar Varma **Roll No. :** 68

Class : TY. Comp. Engg. **Batch :** T4 **Division:**

Expt. No. : **Date :**

Title : Shoemaker's Problem

Remark

Signature

ASSIGNMENT/EXPERIMENT: _____		
Date of Performance:	**Date of Submission:**	
Marks Split Up	**Maximum Marks**	**Marks Obtained**
Performance/Conduction	**3**	
Report Writing	**3**	
Attendance	**2**	
Viva/Oral	**2**	
Total Marks	**10**	
Signature of Subject Teacher		

Title: Shoemaker's Problem

Aim: Shoemaker has N jobs (orders from customers) which he must make. Shoemaker can work on only one job in each day. For each i-th job, it is known the integer Ti ($1 \leq Ti \leq 1000$), the time in days it takes the shoemaker to finish the job. For each day of delay before starting to work for the i-th job, shoemaker must pay a fine of Si ($1 \leq Si \leq 10000$) cents. Your task is to help the shoemaker, writing a programm to find the sequence of jobs with minimal total fine.

Language used: Python

Platform Used: Pycharm, VS code etc.

Sample Input: The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs. First line of input contains an integer N ($1 \leq N \leq 1000$). The next N lines each contain two numbers: the time and fine of each task in order.

Sample Output: For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line. You programm should print the sequence of jobs with minimal fine. Each job should be represented by its number in input. All integers should be placed on only one output line and separated by one space. If multiple solutions are possible, print the first lexicographically.

Example:

Sample Input :

1

4

3 4

1 1000

2 2

5 5

Sample Output:

2 1 3 4

Algorithm/Flowchart:

function schedule_jobs(jobs):

// Sort jobs based on the ratio (processing_time / deadline) in ascending order

```
    sort(jobs, key=lambda job: job.processing_time / job.deadline)

    total_penalty = 0
    current_time = 0  // Current time on the schedule

    // Iterate over each job in sorted order
    for job in jobs:
        // Calculate the finish time of the current job
        finish_time = current_time + job.processing_time

        // Calculate the lateness (delay) of the job
        lateness = max(0, finish_time - job.deadline)

        // Update the total penalty with the lateness of the current job
        total_penalty += lateness

        // Update the current time to the finish time of the current job
        current_time = finish_time

    return total_penalty

// Example usage:
class Job:
    def __init__(self, processing_time, deadline):
        self.processing_time = processing_time
        self.deadline = deadline

jobs = [
```

```
    Job(2, 4),
    Job(5, 2),
    Job(1, 5),
    Job(3, 7)
]

result = schedule_jobs(jobs)
print("Minimum total penalty:", result)
```

Code:

[2]:
```
num_jobs = int(input("Enter the number of jobs: "))
jobs = []
for i in range(num_jobs):
    time = int(input("Enter the processing time for job {}:".format(i + 1)))
    fine = int(input("Enter the fine per day for job {}:".format(i + 1)))
    jobs.append((i + 1, fine, time))
sequence = total_fine(jobs)
print("Sequence of jobs with minimum total fine:", sequence)
```

Output:-

```
Enter the number of jobs: 4
Enter the processing time for job 1: 3
Enter the fine per day for job 1: 4
Enter the processing time for job 2: 1
Enter the fine per day for job 2: 1000
Enter the processing time for job 3: 2
Enter the fine per day for job 3: 2
Enter the processing time for job 4: 5
Enter the fine per day for job 4: 5
Sequence of jobs with minimum total fine: [3, 1, 4, 2]
```

Conclusion: In this way we implement The Shoemaker's Problem using loops and conditional statements.