CS29003 ALGORITHMS LABORATORY
Assignment 10
Last Date of Submission: 27 – September – 2018

General Instruction

1. Please do not use any global variable unless you are explicitly instructed so.
2. Please use proper indentation in your code.
3. Please name your file as `<roll_no>_<assignment_no>`. For example, if your roll number is 14CS10001 and you are submitting assignment 3, then name your file as `14CS10001_3.c or 14CS10001_3.cpp` as applicable.
4. Please write your name and roll number at the beginning of your program.

The post office at IIT Kharagpur offers various services to the users, and each instance of the service may take different time. Additionally, the post office has some internal tasks independent of offering services. The post office runs from Monday to Saturday. While all the employees are available on Monday to Friday, post office can only afford to have one employee on Saturday. However, they should satisfy all the requests of the users on each day. Suppose that on every day, details about the services come online before the start of the day, and they need your help in designing schedules, that can be sent to the users apriori so that they do not have to wait in queue at all. Also, assume that on Monday to Friday, they have enough employees to satisfy all the requests and on Saturday, the total service time for all the requests is less than the working hours, thus all the requests can be served by a single counter.

On Monday to Friday, the post office wants to employ minimum number of employees to be able to satisfy all the requests. The remaining employees will be employed for the internal tasks of the post office. Each of the users knows the time Δt required for a given service, and while putting the request, they specify the exact time when they will arrive in the post office and they need to leave exactly Δt time after that. Note that Monday-Friday are quite busy and they cannot afford delay in processing. Given all the requests for a given day, you need to generate a schedule that minimizes the number of employees used for serving the requests. Each employee is assigned to a different counter Id, and a schedule will be of this format.

```
counter Id
request Id, start time, finish time.
```

On Saturday, on the other hand, the scenario is different. Users only specify the service time δt they require and the target time by which the service needs to be completed. However, they understand that there is only one counter and their requests might be served a bit late. Suppose for a request i, if the target time is t_i, and the counter can only process it at a time $p_i > t_i$, the delay for the request i is denoted as $p_i - t_i$. Post office wants to schedule the requests so as to keep the maximum amount of delay observed for any of the requests to a minimum. For instance, suppose there are 10 requests. Post office will prefer a solution where each request has a delay of 2 units than a solution where only one request is delayed by 10 units, while others are served on time.

Finally, it brings a schedule of the following form:

```
request Id, start time, finish time.
```

Part 1. Scheduling on Weekdays

In this part, you will first read n from the user to denote the total number of requests. The following structure can be used to denote a request:

```
typedef struct _rwd {
  int start_time;
  int service_time;
} reqWD;
```

You will dynamically allocate an array of size n for the weekday requests, and read these from the user. The index in this array will correspond to the request Id.

Now, design a greedy algorithm to prepare a schedule that uses the minimum number of employees (or counters).

Once you have been able to schedule the requests, print the schedule for each of the counters as per the sample output. Write the function $\mathtt{schedule1}()$ that creates the schedule and prints it.

Part 2. Scheduling on Saturday

In this part, you will read m from the user to denote the total number of requests on Saturday. Note that for Saturday, you will only need the service time of the request, as well as the target time as set by the user. Thus, the following structure can be used:

```
typedef struct _rsd {
  int service_time;
  int target_time;
} reqSD;
```

You will dynamically allocate an array of size m for the Saturday requests, and read these from the user. The index in this array will correspond to the request Id.

Now, design a greedy algorithm to prepare a schedule such that the maximum delay observed for any of the requests is minimum. Finally, you should print the schedule as per the sample output. Write the function $\mathtt{schedule2}()$ that creates the schedule and prints it.

Main Function

main() function does the following:

1. Reads n from the user.
2. Reads the start time and length of these n requests from the user. Uses $\mathtt{schedule1}()$ and prints the optimal schedule.
3. Reads m from the user.
4. Reads the length and deadline for each of the m requests. Now, uses $\mathtt{schedule2}()$ to print the optimal schedule.

Sample Output

```
Enter the number of requests over weekday: 5
Enter the start time and length for each of the requests
request 0:: 3 10
request 1:: 7 8
request 2:: 12 2
request 3:: 13 12
request 4:: 21 2
Print the schedule as per various counters in the following format:
counter Id
request Id, start time, finish time
Note: Not the optimal schedule, just the format
Counter 0
0 3 13
4 21 23
Counter 1
1 7 15
Counter 2
2 12 14
Counter 3
3 13 25


Enter the number of requests over Saturday: 5
Enter the length and deadline for each of the requests
request 0:: 6 32
request 1:: 8 12
request 2:: 5 30
request 3:: 20 28
request 4:: 2 21
Print the schedule in the following format:
request Id, start time, finish time
Note: Need not be optimal. Only use for format.
4 0 2
2 2 7
0 7 13
1 13 21
3 21 49
```