Programming Assignment 2

Introduction

The fastfood chain Q has L service counters open to serve the store's M menu items to a total of N customers for any particular time period. Each counter is assigned a number $i \in [1, L]$, based on its relative distance from the entrance; station 1 being the nearest and station L being the farthest. Each kth customer, $k \in [1, N]$, always goes to the shortest line in the store. If there are multiple lines with the same length, the customer queues to the nearest line from the entrance.

The fastfood chain's jth menu item is numbered 2^{j-1}, $j \in [1, M]$. Each menu item has a corresponding cost c_j (in Php) and time to prepare t_j (in seconds). Each kth customer orders a meal (containing multiple items in the menu) by giving an order number o_k, which is the sum of the item number of the individual items in the menu. For example, if a customer wants to order the 1st and 4th item in the menu, he will simply give an order number of 9 ($2^{1-1} + 2^{4-1} = 1 + 8 = 9$).

Problem Description

The owner wants to predict how much worth of food his store actually provides and how fast his store could serve them for a given set of customers. In particular, the owner wants to determine the following: (1) how much the store earns, segregated by the amount for each counter; (2) how fast the store can finish all orders; and (3) how crowded the store gets, which is the maximum number of people lined up in all counters.

For simplicity, we assume the following: (1) no customer will order a menu item multiple times; (2) the cost and preparation time for each menu item is independent of each other; (3) an order's preparation time encompasses all other factors such as payment and ordering; (4) each customer arrives every T seconds, with the first customer arriving at time $t = 0$; and (5) there is no delay when a customer queues in one of the counters.

Input Description

The first line of the input contains $1 \leq L \leq 20$, $1 \leq M \leq 50$, $1 \leq N \leq 1000$ and $0 \leq T \leq 300$ separated by a whitespace. The next M lines contains $c_j > 0$ and $t_j > 0$ separated by a whitespace. The last N lines contains $1 \leq o_k \leq 2^M - 1$. To simplify the problem, you can assume that any given line is always open and that there is no maximum line length.

Output Description

The first line of the output contains the amount earned by each service counter. The second line of the output contains the total time taken to complete all orders and the maximum total customer queue size separated by whitespace.

Grading Milestones

10%	able to read/write formatted input/output
30%	able to determine the total cost of all orders
40%	able to produce the required output assuming $T = 0$ for at least 60% of the provided test cases
60%	able to produce the required output for at least 60% of the provided test cases
80%	able to process all provided test cases
100%	able to handle all test cases; time complexity of O(N) when L,M are held constant
100%+bonus	able to toggle customer queueing behaviour

Sample Input/Output 1

```
2 4 6 0
25 30
75 60
100 90
50 45
4
9
11
6
13
5
```

```
425 375
390 6
```

Sample Input/Output 2

```
2 4 6 45
25 30
75 60
100 90
50 45
4
9
11
6
13
5
```

```
550 250
510 3
```

Bonus

In an attempt to improve the store's service time (and how crowded the store gets sometimes), an additional worker is assigned to guide the incoming customer to a line for one of the service counters. Instead of letting the next customer go to the shortest line, s/he is directed to the line for the service counter that is currently expected to finish first; if there are multiple lines that will finish at the same time, the customer is directed to the farthest line from the entrance.

Notes

1. If you have a question involving source code, please do post the code snippet and/or error messages. Always resolve compile-time errors in the order they appear in your code. If you think that the code snippet might be too revealing of your work, post it as a private question first and I'll decide if said code may be made public to your classmates.
2. Make sure that you are using a g++-8 compiler which is either from GCC 8.1 or GCC 8.2 so that you will not have issues during checking. If you want to use a new C++ specification kindly comment what compile command you are using as a comment at the beginning of your source code.
3. You may use any library in the C++ STL.
4. All unspecified parts are made open to interpretation. If you really want to make sure, just post a question at Piazza.
5. Test input of varying size (and possibly difficulty) will be provided a week before the soft deadline; however the actual test input for checking will not be given.

Submission

1. Soft deadline is set to 9:00 PM of 19 March 2019.
2. Hard deadline is set to 9:00 PM of 27 March 2019.