United International University (UIU)
Dept. of Computer Science & Engineering (CSE)
Course Code: CSI 228 Section:
Course Title: Algorithms Lab

Maximize your silver coins

Suppose, you are working as an apprentice magician who makes potions. You work **6 hours a day** from 12:00 PM to 6 PM. You make **five** types of potions: their names, prices (in silver coins) and times to make them (in hour) is given bellow:

Potion	Time needed (hour)	Price (#silver-coins)
r	2	2000
s	1	1000
t	1	4000
u	2	3000
v	1	3000

You receive orders from your classmates to make potions; each order provides you three information. **One**, the name of the classmate. **Two**, the name of the potion. **Three**, within which time you must make this potion (**last time**). If you fail to make the potion within that time, then you are not paid. You cannot make multiple potions at a time. **Write the following greedy algorithm to choose the orders to maximize your silver coins and cancel the others. Print the name of classmates you have chosen and the #silver coins you are going to earn tomorrow.**
[You do not need to work continuously to make a potion. For example, if you want you can make the potion **r,** which needs 2 hours to make, from 7PM-8PM then 9PM-10PM.]

1. Sort the orders by descending order of *#silver-coins per hour*.
2. Choose the order with the **highest** *#silver-coins per hour* and do it in the last possible hours. Make those hours occupied.
3. Choose the **next highest** paid (*#silver-coins per hour*) order, find the last possible hours that are not occupied, do it at those hours and make those hours occupied. If enough hour is not free to make this potion, you cancel this order.
4. Repeat step 3 until no order in left unscheduled.

Sample Input

classmate	Potion name	Last time (PM)
a	r	4
b	s	1
c	t	1
d	u	4
e	v	3

Sample Output

```
c, e, d
10000

Explanation:
c:12PM   ,4000sc
e:2PM    ,3000sc
d:1PM,3PM,3000sc
b: not possible
a: not possible
```

classmate	Potion name	Last time (PM)
a	r	4
b	s	5
c	t	1
d	u	3

```
c, d, b
8000

Explanation:
c:12PM   ,4000sc
d:1PM,2PM,3000sc
b:4PM    ,1000sc
a: not possible
```

United International University (UIU)
Dept. of Computer Science & Engineering (CSE)
Course Code: CSI 228 Section:
Course Title: Algorithms Lab

Maximize your payment

Suppose, you are temporarily working as a photoshop editor. You work **10 hours a day** from 12:00 PM to 10 PM. You have a unique set of customers; each customer provides you three information for the next day tasks. **One is**, what he is going to pay you (**payment**); **two**, **how many hours** will be needed to do that task; and **three**, within which time you must complete this task (**last time**). If you fail to do the tasks within that time, then that customer does not give the task. You cannot serve multiple customers at a time. **Write the following greedy algorithm to choose the tasks to maximize your payment.** [You can do a task partially at one time and finish it at another time.]

1. Sort the tasks by descending order of *payment per hour*.
2. Choose the task with the maximum *payment per hour* and do it in the last possible hours. Make that hour occupied.
3. Choose the next maximum paid (*payment per hour*) task, find the last possible time that is not occupied, do it at that time and make that hours occupied. If no such time exists, you cannot do this task.
4. Move to the next maximum profitable task (based on *payment per hour*) and repeat step 3.

Sample Input — **Sample Output**

Customer	Last time (PM)	Time needed (hour)	Payment (Tk)
a	4	2	2000
b	1	1	1000
c	1	1	4000
d	2	2	3000

```
c, a
6000
```

Customer	Last time (PM)	Time needed (hour)	Payment (Tk)
a	2	1	1000
b	1	1	190
c	2	1	270
d	1	1	250
e	3	1	150

```
c, a, e
1420
```

You are working as a part-time programmer where you have to prioritize and choose tasks for your boss Mr. Tom so that your boss's earns the maximum possible amount each day. Each day in the morning Mr. Tom informs you

how many free hours he has that day which is C.

Then he informs that he received N tasks from clients.

Then for each task t_i, he tells you the client's name's first letter (L_i), how many hours it will take to complete (H_i) and what is the payment (P_i).

The client will pay him only if he can complete the task that day.

You have to write a code **using tabulation method** to choose tasks for that day so that your boss can earn the maximum possible amount that day. **You have to print the maximum amount Mr. Tom can earn that day.**

Bonus: Print the tasks you have chosen – 2 Marks

Sample Input $\boldsymbol{C}$ $\boldsymbol{N}$ $\boldsymbol{L_1, H_1, P_1}$ $\boldsymbol{L_2, H_2, P_2}$ **…**	**Sample Output**
5 5 A 2 6000 B 2 1000 C 1 1200 D 2 2000 E 1 3000	11000 **E** **D** **A**
5 4 B 2 1000 C 1 1200 D 2 2000 E 1 3000	6200 **E** **D** **C**

United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Course Code: CSI 228 Section:

Course Title: Algorithms Lab

Activity Selection Problem

[Hint]:

First sort by finish

```
GREEDY-ACTIVITY-SELECTOR(s, f)
1  $n \leftarrow length[s]$
2  $A \leftarrow \{a_1\}$
3  $i \leftarrow 1$
4  for $m \leftarrow 2$ to $n$
5      do if $s_m \geq f_i$
6          then $A \leftarrow A \cup \{a_m\}$
7              $i \leftarrow m$
8  return $A$
```

Sample Input							Sample Output
i	1	2	3	4	5	6	2
s_i	1	3	0	5	3	5	
f_i	4	5	6	7	8	9	

United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Course Code: CSI 228 Section:

Course Title: Algorithms Lab

Maximize your payment

Suppose, you are temporarily working as a photoshop editor. You work 8 hours a day from 12:01 PM to 8 PM and each customer takes 1 hour from you to edit their photos. You have a unique set of customers; each customer provides you two information for the next day tasks. One is what he is going to pay you (payment); and two, within which time you must complete this task (last time). If you fail to do the tasks within that time, then that customer does not give the task. You cannot serve multiple customers at a time. How you can choose to do the tasks to maximize your payment?

[Hint]:

1. Sort the tasks by descending order of payment.
2. Choose the task with the maximum payment and do it in the last possible hour. Make that hour occupied.
3. Choose the next maximum paid task, find the last possible time that is not occupied, do it at that time and make that hour occupied. If no such time exists, you can not do this task.
4. Move to the next maximum profitable task and repeat step 3.

Sample Input			Sample Output
Customer	Last time (PM)	Payment (Tk)	c, a 6000
a	4	2000	
b	1	1000	
c	1	4000	
d	1	3000	
Customer	Last time (PM)	Payment (Tk)	c, a, e 1420
a	2	1000	
b	1	190	
c	2	270	
d	1	250	
e	3	150	

Minimize your travel

A family goes for picnic along a straight road. There are k hotels in their way situated at distances $d_1, d_2 \ldots d_k$ from the start. The last hotel is their destination. They can travel at most L length in day. However, they need to stay at a hotel at night. Find a strategy for the family so that they can reach hotel k in minimum number of days.

[Hint]:
travel on a particular day as long as possible, i.e., do not stop at a hotel if the next hotel can be reached on the same day.

Sample Input L k $d_1, d_2 \ldots d_k$	**Sample Output**

5
3
4 9 14
	3

5 4 **4** 6 **9 14**	3
5 5 **4** 6 **9 14 14**	3
1 5 **4** 6 **9 14 14**	-1

United International University (UIU)
Dept. of Computer Science & Engineering (CSE)
Course Code: CSI 228 Section:
Course Title: Algorithms Lab

Maximum Tables Occupied

Your friend has opened a new café. You wanted to know how occupied his café was yesterday. Instead of giving a direct answer, he let you know the arrival and departure times of different groups at his café [This is your input]. Each group occupies one table. You must find the maximum number of tables that were occupied at a time there.

[Hint]:
1. Sort the arrival array
2. Sort the departure array
3. Try to merge the sorted arrays
4. Keep track of the number of groups inside the café during each iteration
5. And also keep track of the maximum number at any time

Sample Input	Sample Output

arrival = {2.00, 2.10, 3.00, 3.20, 3.50, 5.00}
departure = {2.30, 3.40, 3.20, 4.30, 4.00, 5.20}
	2

arrival = {9.00, 9.40, 9.50, 11.00, 15.00, 18.00}
departure = {9.10, 12.00, 11.20, 11.30, 19.00, 20.00}
	3

Explanation for the first sample input:
1. Sort the arrival array
   a. arrival = {2.00, 2.10, 3.00, 3.20, 3.50, 5.00}
2. Sort the departure array
   a. departure = {2.30, 3.20, 3.40, 4.00, 4.30, 5.20}
3. Merge the two sorted arrays into a new array so that the merged array is also sorted. Also, keep track of which time is arrival and which is departure.
   a. Merged = {(2.00,a), (2.10,a), **(2.30,d)**, (3.00,a), (3.20,a), **(3.20,d)**, **(3.40,d)**, (3.50,a), **(4.00,d)**, **(4.30,d)**, (5.00,a), **(5.20,d)**} // a= arrival, d=departure
4. Keep track of the number of tables occupied at the café during each iteration
   #tables occupied
   - (2.00,a) -> 1
   - (2.10,a) -> 2
   - **(2.30,d)** -> 1
   - (3.00,a) -> 2
   - (3.20,a) -> 3
   - **(3.20,d)** -> 2
   - **(3.40,d)** -> 1
   - (3.50,a) **-> 2**
   - **(4.00,d)** -> 1

- **(4.30,d)** -> 0
- (5.00,a) -> 1
- **(5.20,d)** -> 0

5. **Answer 2**; you may think it is 3 but it is 2 because at 3.20 even though 1 group arrives, one leaves immediately at 3.20 too