Convert all hhmm inputs into minutes such that 5am = 0 minutes.
Make nodes which are basically flights using all the inputs you take. A node must contain 6 details - flight number (4 character string), departure and arrival time (int -> minutes following 5 am). flight duration (int -> minutes), source city (int) and destination city (int).
Add an edge if there is a connecting flight with the weight denoting the waiting time between the two flights.

DSA Lab 12 | TheFinalTask

Time Limit: 3 sec
Memory Limit : 1024 MB

The task is to implement a flight trip planner. For every pair of cities, denoted by distinct integers, you will be provided with flight number, departure time and arrival time of that flight. Note that there could be multiple flights between a pair of cities, each with a distinct flight number.
“city1 city2 flight-no. depart arrival " means that there is a flight called "flight-no" (which is a string of four characters eg. ABCD) which leaves city1 at the time "depart" and arrives city2 at the time "arrival". The format for the time is *hhmm* hours.

For simplicity, assume that all flights depart after 5 am and arrive before 11 pm. Further, assume that we would always want to travel for one day only (so no overnight staying at any of the cities).

Now given two cities *c1* and *c2*, and times "t1", "t2" you have to find the *best trip* which leaves city *c1* after time "t1" and arrives at city *c2* before time "t2". There could be multiple possible paths covering different cities to complete the trip. You also need to take into account the waiting time for the next flight once you reach an intermediate city. The *best trip* would be the one which takes overall least amount of time, after one boards the first flight and reaches the final destination. Note that the trip could involve multiple flights.

Note you should check that the arrival time of a flight in this trip is before the departure time of the next flight in the trip. You can assume that the departure time would be distinct for all flights for a given pair of cities.

Input
The first line contains 1 number denoting the number of cities X.
The next line contains 4 elements(single space separated) src des t1 t2, where src and des are integer ids representing the source and destination cities respectively (1 <= src,des <= X) and t1 and t2 give the time interval in which a flight should cover the journey as described before. t1 and t2 are time in format hhmm
The next line contains an integer N, representing the number of flights.
Next, N lines contain input in the following format in each line
c1 c2 f d a , means there is a flight with flight no. f from c1 to c2 departing at time d and arriving at the time a (d < a , ‘<’ means bigger time). c1 and c2 are integers (1 <= c1,c2 <= X), d and a are the time in format hhmm and f is a 4 character string of alphabets (both uppercase and lowercase).

Constraints

Normal:
2 <= X <= 100
1 <= N <= X*(X-1)
No multiple flights between two city pairs
Bonus:
2 <= X <= 500
1 <= N <= 4 * X * (X -1)
There can be multiple flights between city-pairs.

Output
Print the number of flights (T) in the trip. If no trip is feasible, print -1.
Give the sequence of flight numbers, (a sequence of T space-separated strings), in which the journey would be covered as per the constraints described.

Sample Input 1
```
5
1 4 1000 1500
8
1 2 ABCE 1030 1150
1 2 BCDE 1045 1215
1 5 CDEE 1315 1415
2 4 EFGE 1230 1430
3 1 MNOE 0940 1100
5 3 AAAE 1305 1335
5 3 BBBE 1400 1440
5 4 PQRE 1630 1840
```

Sample Output 1
```
2
BCDE EFGE
```

Explanation
Route possible 1 2 4
Along route 1 to 2, flight BCDE is preferred over flight ABCE as the total time spent (in flight+waiting for connecting flight to city 4) is lesser in case of BCDE, even though the flight duration was lesser for ABCE

Sample Input 2
```
5
4 1 1000 1600
10
1 2 ABCX 1030 1150
```

```
1 2 BCDX 1045 1215
1 5 CDEX 1315 1415
2 4 EFGX 1230 1430
3 1 MNOX 1500 1620
4 5 GGGX 1015 1205
5 3 AAAX 1305 1335
5 3 BBBX 1400 1440
5 4 PQRX 1630 1840
5 1 DDDX 1150 1350
```

Sample Output 2

```
-1
```

Explanation

DDDX would have departed by the time GGGX reached city 5, so the connecting sequence DDDX - GGGX is not possible.
Other connections GGGX - AAAX - MNOX or GGGX - BBBX - MNOX are not possible as the arrival time lies outside the time range specified (it's later than t2 value)
No other possibility exists.

Cases with no multiple flights between a pair of citiies

Sample Input 3

```
5
1 5 1000 1600
6
1 2 ABCD 1100 1130
1 3 AAAA 1215 1235
1 5 BBBB 900 1030
2 4  BCDA 1140 1200
3 5 MNOP 1325 1425
4 5 PQRS 1215 1255
```

Sample Output 3

```
3
ABCD BCDA PQRS
```

Explanation

Computing the total time travelling over route 1-2-4-5 is 30+10+20+15+40=115 minutes, where 30, 20, 40(in minutes) are in-flight duration and 10, 15(in minutes) is time spent waiting for connecting flight.
Over route 1-5 the departure time is out of the preferred time interval, hence not taken.
Over route 1-3-5, total time spent is 20+50+60 =130 minutes, where 20 and 60 min. are time spent in-flight and 50 minutes spent waiting for connecting flight.

Hence, route 1-2-4-5 takes least time.

Sample Input 4
```
5
1 5 1000 1400
6
1 2 ABCD 1100 1130
1 3 AAAA 1215 1235
1 5 BBBB 900 1030
2 4  BCDA 1125 1200
3 5 MNOP 1325 1425
4 5 PQRS 1215 1255
```

Sample Output 4
```
-1
```

Explanation
No route possible as
For 1-2-4-5, BCDE would have departed before ABCD arrived at city 2.
For 1-5, departure time is out of specified time interval.
For 1-3-5, MNOP would arrive out of the specified time interval.
Hence, no route is possible.

Note
Some ideas for the modelling graph for this problem are discussed below. You are free to come up with your own model as well.

Option 1: Each city by itself is not a vertex on the graph, but each flight constitutes an edge. If there are m flights leaving a city, and n flights reaching it, there will be m+n vertices for that city. Each flight will be an edge. Further there will be edges for all the waiting times at a city. Thus if a flight lands at 10 in city XX, and there are flights leaving XX at 12, 14, and 15 hours, then there will be 3 edges between these landings and take off times. If another flight lands at 12:30 in city XX then there will be two more edges connecting to flights leaving at 14 and 15 hours.
In other words each flight is an edge and each connection time at an airport is an edge.

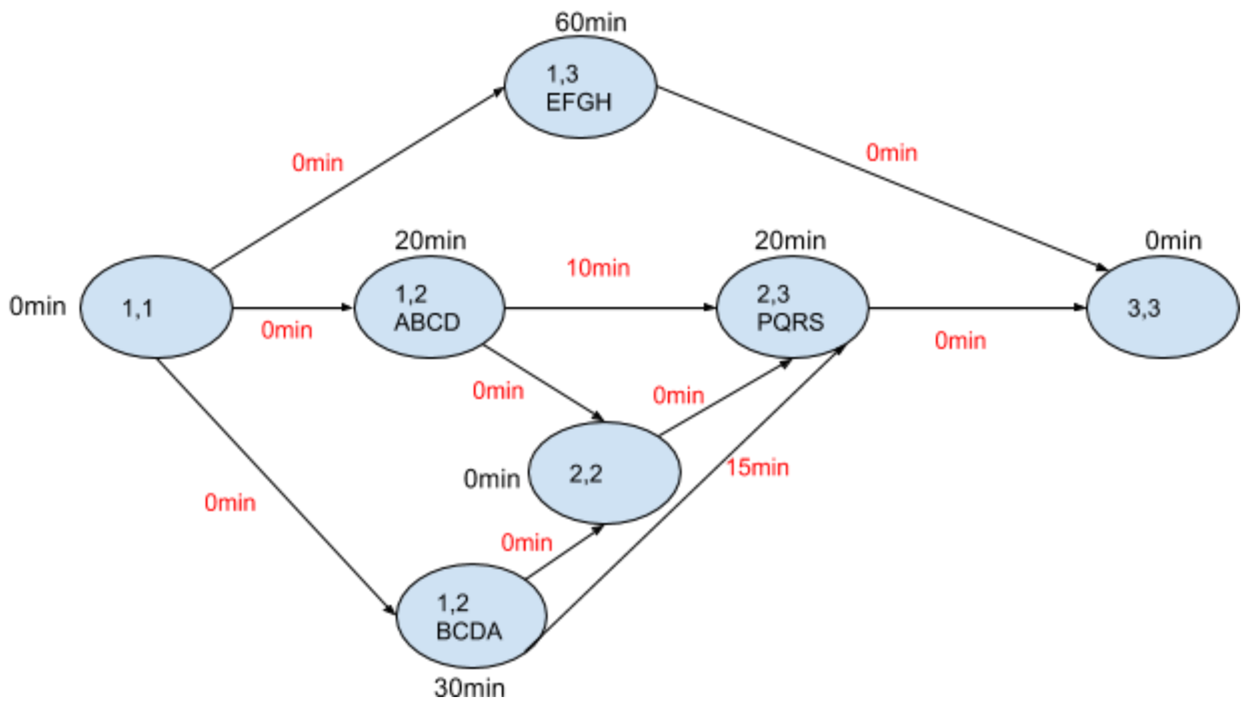
Option 2: One can model every flight as a node that maintains the departure and arrival time, and add edges between each node if there is a connecting flight, with edge wait denoting the waiting time between the two flights **(add this edge provided waiting time is non-negative)**. Each node also carries a weight denoting the flight-duration of the flight the node represents. For instance, consider the following input:

```
3
1 3 1000 1500
4
1 2 ABCD 1100 1120
1 2 BCDA 1115 1145
1 3 EFGH 1200 1300
2 3 PQRS 1130 1150
```

Expected Output

```
2
ABCD PQRS
```

The corresponding graph is shown below where weight is denoted in minutes. Note that the complete node structure has not been included for brevity but the nodes would also contain fields of arrival and departure time to check for the constraints.



There is an edge between ((1,2),(2,3)) indicating there is a connecting flight at city 2 to go from city 1 to city 3. Edge weights (in red) denote waiting time between connecting flights. Node weight in black denote the flight duration. Note that nodes of the form (x,x) eg. (1,1), (2,2), (3,3) are not flights but denote the city from where flights takeoff or land. Such nodes would be useful to find optimal route from city A to city B- for such nodes the node weight is 0 as there is no flight duration and edges from or to such nodes would also be 0 as no waiting time is involved here.