- Section 1 -

1

- Section 2 -

2

☆ Manhattan Distance

This is just a simple into test to get you used to the HackerRank platform.

You need to find the distance between two points using the Manhattan distance method (you have been supplied with an implementation of this method, so no need to research this!)

Your input will come via standard input in the form of 2 lines of text, each of the two lines will contain the x and y coordinates of a point.
You need to work out the distance between the supplied points and print your answer to standard output.

Example:

```
1 1
2 2
```

so the above output is two coordinates, first coordinate is (1,1), second coordinate is (2,2)

Expected output

```
2
```

Manhattan distance between (1,1) and (2,2) is 2, so we would expect "2" to be output

YOUR ANSWER

☆ Getting Tickets to Fans V3

Introduction

One of viagogo's missions is to get every fan the ticket they want. We want to help them find those tickets based on the following requirements:

- They want one ticket for the closest event
- They want the cheapest ticket for the event (if two events are equally close they always prefer the cheapest)
- If the tickets for two events are the same price and the same distance, they'll always prefer the event with the smallest Id (don't ask us why)
- All distance should be measured using Manhattan Distance (the method to get this has been supplied in the code below)

We're going to provide you with a list of all the events and tickets, as well as the customers who want tickets. We want you to tell us which event the customer will buy a ticket for and the price they pay.

Input and output details

Input Specification

- World size - int 1..100
- Number of events - int 0..1000
- One line per event, each line containing space separated ints for:
  - EventId
  - X Coordinate [0 - based]
  - Y Coordinate
  - 0 or more ticket prices
- Number of buyers - int 0..1000
- One line per buyer, each line containing space separated ints for:
  - X coordinate
  - Y coordinate

For Example

Note: *the input files provided will not contain any comments*

```
5             // World size
2             // Number of events
1 1 1 40 60   // Event 1, located at 1,1, with two tickets at 40 and 60
2 1 4 50      // Event 2, located at 1,4, with one ticket at 50
3             // 3 buyers
3 3           // First buyer at 3,3
3 2           // Second buyer at 3,2
4 3           // Third buyer at 4,3
```

Output specification

- You should print one line for each buyer, each line containing space separated ints for:
  - Id of event
  - Price of ticket

For Example

Note: *the output you print create should not contain any comments*

```
2 50     // First buyer purchases a ticket for 50 from event 2
1 40     // Second buyer purchases a ticket for 40 from event 1
1 60     // Third buyer purchases a ticker for 60 from event 1
```

Examples

Example 1

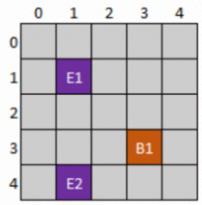
Input

```
5
2
1 1 1 40
2 1 4 50
1
3 3
```

Expected Output

```
2 50
```

Explanation



There are two events and one buyer. The events have tickets as follows:

- Event One (E1): 1 ticket for 40
- Event Two (E2): 1 ticket for 50

The buyer is closest to event 2 (E2) so he will purchase a ticket for that event, There is only one ticket at 50 for event 2, to the price they will pay is 50.
Therefore you will print out "2 50" (2 for Event 2, and 50 for the price they will pay)

Example 2

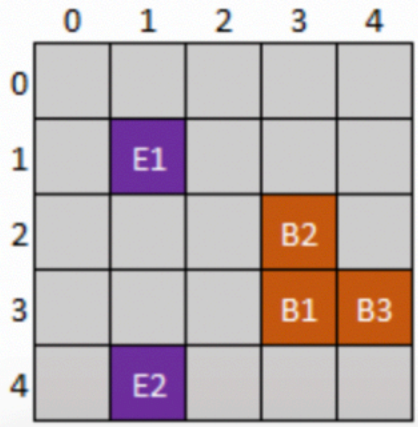
Input

```
5
2
1 1 1 40 60
2 1 4 50
3
3 3
3 2
4 3
```

Expected Output

```
2 50
1 40
1 60
```

Explanation



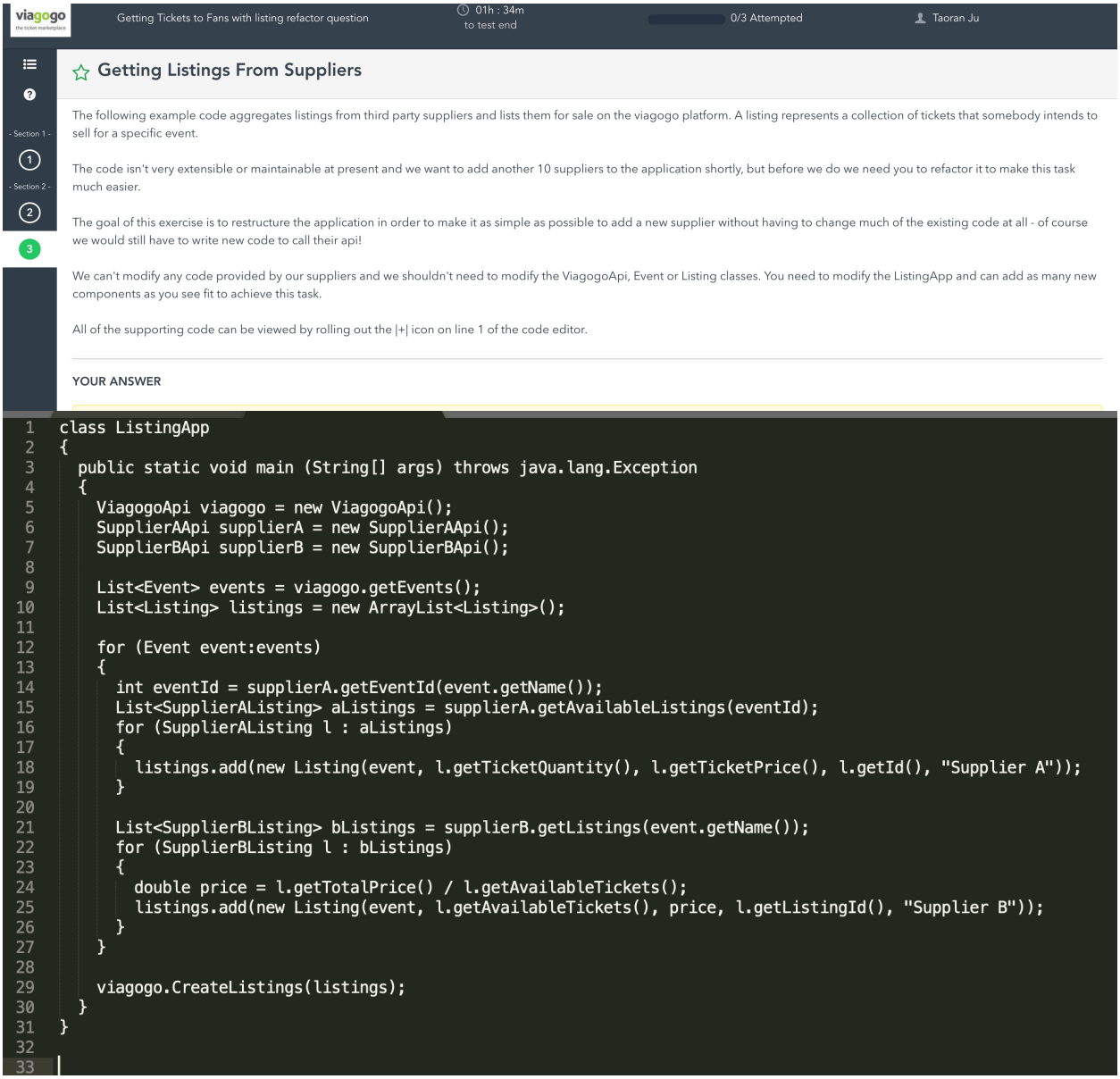
There are two events and three buyers. The events have tickets as follows:

- Event One (E1): 2 tickets, one for 40 and another for 60
- Event Two (E2): 1 ticket for 50

The first buyer (B1) is closest to event 2 (E2), so he purchases the cheapest ticket for 50.
The second buyer (B2) is closest to event 1 (E1), so he purchases the cheapest ticket for 40.
The third buyer (B3) is closest to event 2 (E2), but there are no tickets left (because the first buyer though the only ticket). The next closest event is event 2, so he purchases the last remaining ticket for 60.

Notes

- You can submit your test as often as you want, so ***submit often and early*** so that you have your attempts stored
- Distance from buyer to event should be computed as the Manhattan distance (method to get Manhattan distance is supplied to you in the code below)
- If there are no tickets available for a buyer, return event id -1 and price 0 (So output "`-1 0`")
- If two events are the same distance from the buyer, then the buyer prefers the cheapest ticket, if the tickets for both event are the same price, then the buyer prefers the event with the smallest event Id
- Once a ticket is allocated to a buyer it cannot be given to another buyer
- No two events will share the same location
- Buyers may share the same location




```
class ListingApp
{
  public static void main (String[] args) throws java.lang.Exception
  {
    ViagogoApi viagogo = new ViagogoApi();
    SupplierAApi supplierA = new SupplierAApi();
    SupplierBApi supplierB = new SupplierBApi();

    List<Event> events = viagogo.getEvents();
    List<Listing> listings = new ArrayList<Listing>();

    for (Event event:events)
    {
      int eventId = supplierA.getEventId(event.getName());
      List<SupplierAListing> aListings = supplierA.getAvailableListings(eventId);
      for (SupplierAListing l : aListings)
      {
        listings.add(new Listing(event, l.getTicketQuantity(), l.getTicketPrice(), l.getId(), "Supplier A"));
      }

      List<SupplierBListing> bListings = supplierB.getListings(event.getName());
      for (SupplierBListing l : bListings)
      {
        double price = l.getTotalPrice() / l.getAvailableTickets();
        listings.add(new Listing(event, l.getAvailableTickets(), price, l.getListingId(), "Supplier B"));
      }
    }

    viagogo.CreateListings(listings);
  }
}

```