Traveling Salesman Problem (TSP) is an important problem in computer science. The TSP aims at finding a route passed all cities exactly once with minimum traveling distance. There are two representations for TSP problems: using the coordinates of all cities or using the distance matrix for the distance between all pairs of the cities.

Your goal is to write a program to store the TSP problem represented by coordinates of all cities, and to translate (store and output) the representation to distance (cost) matrix. The TSP class object should contain an array of class city object, which records the x and y coordinates value of a city. Overload the stream extraction and insertion operators for the two classes to input cities' coordinates and output TSP instances; that is, make the following statements possible (Assume tspObject and cityObject are the two objects of class TSP and class City, respectively):

```
cin >> tspObject;
cout << tspObject;
cin >> cityObject;
cout << cityObject;
```

Also provide a default constructor, copy constructor, assignment operator, and destructor for the class. Remember to use dynamic memory allocation in the class according to the input of number of cities.

Requirement: Provide a class for storing the position of each city and a class for storing TSP. Prepare appropriate constructors and overloaded operators for your classes, and encapsulate the methods. Separate your program in files of three kinds: the class header file (.h), the class source code file (.cpp), and the file containing main function (.cpp).

Prohibited: Use C-style input/output.

Input

Each case contains an integer n, representing the number of cities, and 2n integers, denoting the coordinates of the n cities. The input ends with -1.

Output

For each case, output the distance matrix, where the element $m_{i,j}$ at i^{th} row and j^{th} column is the Euclidean distance between i^{th} city and j^{th} city according to the input order. For each distance value, please truncate it to an integer and set the field width to 4. Any two adjacent columns are separated by a space, and any two adjacent rows are separated by an end line stream manipulator. Two consecutive cases are also separated by an end line stream manipulator.

Sample Input

```
5
-1 2
3 1
4 3
5 -4
-5 0
-1
```

Sample Output

```
   0   4   5   8   4
   4   0   2   5   8
   5   2   0   7   9
   8   5   7   0  10
   4   8   9  10   0
```