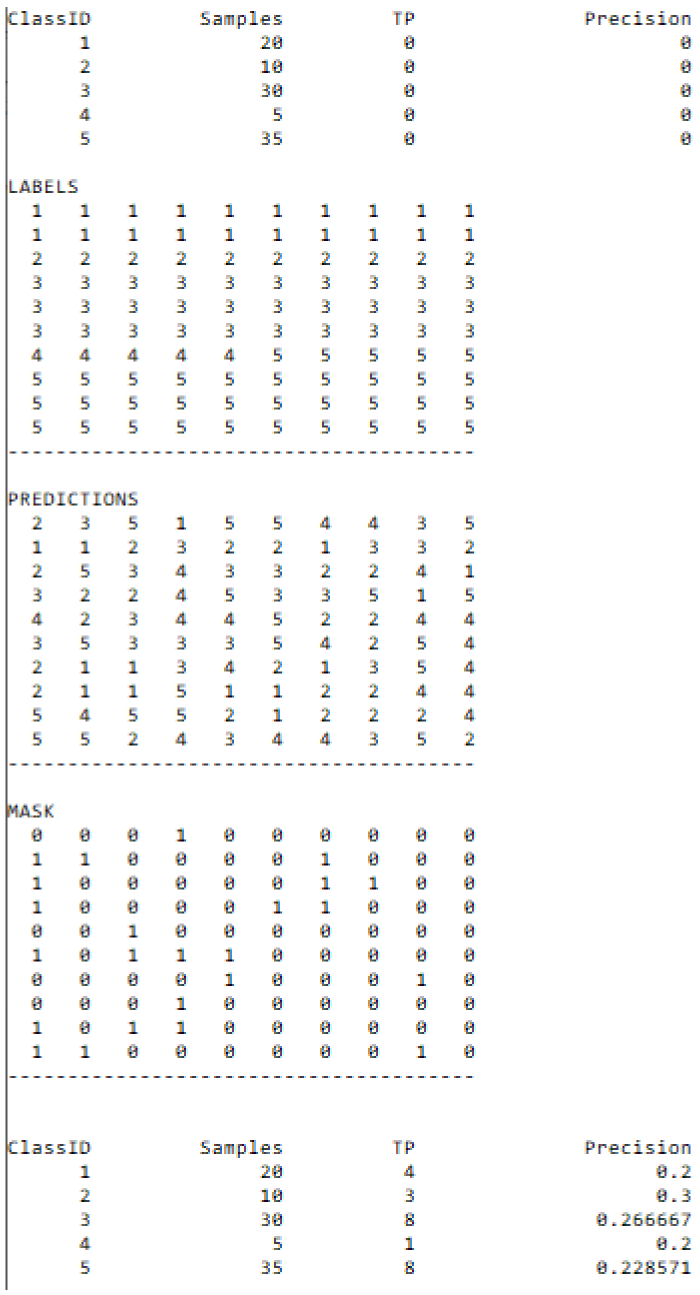


```
ClassID         Samples         TP            Precision
      1              20          0                      0
      2              10          0                      0
      3              30          0                      0
      4               5          0                      0
      5              35          0                      0

LABELS
   1   1   1   1   1   1   1   1   1   1
   1   1   1   1   1   1   1   1   1   1
   2   2   2   2   2   2   2   2   2   2
   3   3   3   3   3   3   3   3   3   3
   3   3   3   3   3   3   3   3   3   3
   3   3   3   3   3   3   3   3   3   3
   4   4   4   4   4   5   5   5   5   5
   5   5   5   5   5   5   5   5   5   5
   5   5   5   5   5   5   5   5   5   5
   5   5   5   5   5   5   5   5   5   5
---------------------------------------

PREDICTIONS
   2   3   5   1   5   5   4   4   3   5
   1   1   2   3   2   2   1   3   3   2
   2   5   3   4   3   3   2   2   4   1
   3   2   2   4   5   3   3   5   1   5
   4   2   3   4   4   5   2   2   4   4
   3   5   3   3   3   5   4   2   5   4
   2   1   1   3   4   2   1   3   5   4
   2   1   1   5   1   1   2   2   4   4
   5   4   5   5   2   1   2   2   2   4
   5   5   2   4   3   4   4   3   5   2
---------------------------------------

MASK
   0   0   0   1   0   0   0   0   0   0
   1   1   0   0   0   0   1   0   0   0
   1   0   0   0   0   0   1   1   0   0
   1   0   0   0   0   1   1   0   0   0
   0   0   1   0   0   0   0   0   0   0
   1   0   1   1   1   0   0   0   0   0
   0   0   0   0   1   0   0   0   1   0
   0   0   0   1   0   0   0   0   0   0
   1   0   1   1   0   0   0   0   0   0
   1   1   0   0   0   0   0   0   1   0
---------------------------------------


ClassID         Samples         TP            Precision
      1              20          4                    0.2
      2              10          3                    0.3
      3              30          8               0.266667
      4               5          1                    0.2
      5              35          8               0.228571
```

Assume that we have five classes from 1 to 5. Also, for each class, we have to store the number of samples, true positive and precision values.

a) Use **map** to store classID and the number of samples, true positive, precision. Initialize the map with the values given in the figure.

b) Print the map. An example output is shown in the figure. **You must print the map with the same format in the figure.**

c) Define an **integer vector** with 100 items for **labels**. In the vector, the first 20 items must be 1, then successive 10 items 2, then successive 30 items 3 and so on.

d) Use a generator function (i.e., **int gt (void)**) and appropriate algorithm to initialize labels vector (**Hint: Do not use loop)**.

e) Print the labels vector. An example output is shown in the figure. **You must print the vector with the same format in the figure.**

f) Define an **integer vector** with 100 items for **predictions**. In the vector, the items are randomly generated in the interval [1,5].

g) Use a generator function (i.e., **int rand15 (void)**) and appropriate algorithm to initialize predictions vector (**Hint: Do not use loop)**.

h) Print the predictions vector. An example output is shown in the figure. **You must print the vector with the same format in the figure.**

i) Define an **integer vector** with 100 items for **mask**. In the vector, the items that are matched with prediction and label vectors will be 1. Other items will be 0.

j) Use a predicate function (i.e., **bool evaluate (int)**) and appropriate algorithm to obtain mask vector (**Hint: Do not use loop)**.

k) Print the mask vector. An example output is shown in the figure. **You must print the vector with the same format in the figure.**

l) For each class, calculate true positive values, which is the number of 1's in the mask vector. For example, we know that the first 20 items belonging to class 1. Also, 4 values that are corresponding to 1's in the first 20 items of the mask vector. Therefore, TP is 4 for class 1. Similarly, the successive 10 items are belonging to class 2. Also, 3 values that are corresponding to 1's in the items between 20 and 30 of the mask vector. Therefore, TP is 3 for class 2. Update the map with the true positive values.

m) For each class, calculate precision values as follows:

$$precision = \frac{TP}{Samples}$$

Update the map with the precision values.

n) Print the updated map. An example output is shown in the figure. **You must print the map with the same format in the figure.**

Assume that we have point cloud data which include N points and each point consists of x, y and z coordinates. The distribution of points in the radius neighbourhood is so important to analyze geometric structure. In the figure, radius neighborhood is showed in two dimension for clarity. A circle is centered around a search point (red point) and points circled with the radius are called the neighbor points.

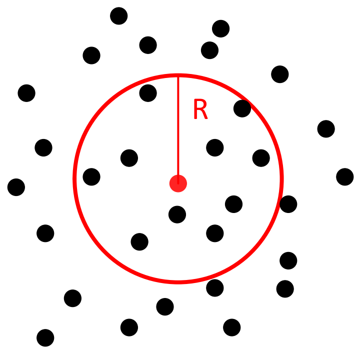
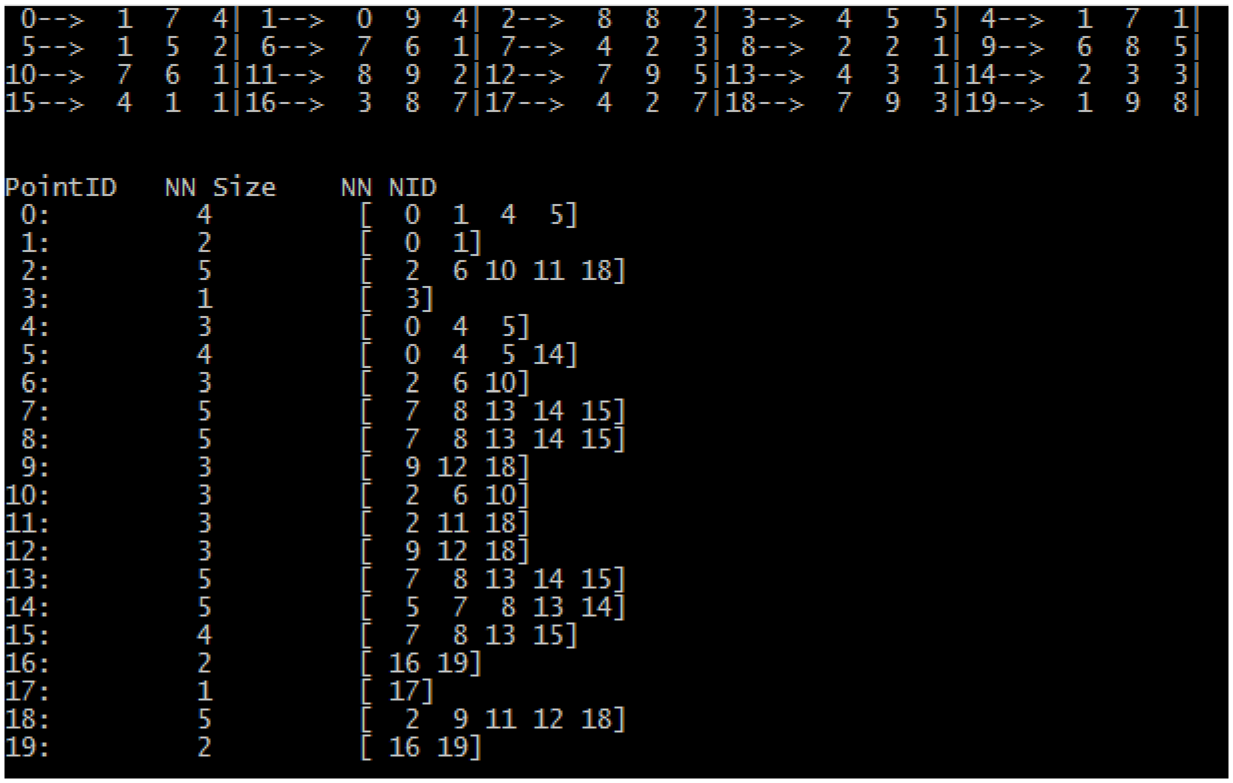


Figure 1. The neighbors of point in 2D



```
 0-->  1  7  4| 1-->  0  9  4| 2-->  8  8  2| 3-->  4  5  5| 4-->  1  7  1|
 5-->  1  5  2| 6-->  7  6  1| 7-->  4  2  3| 8-->  2  2  1| 9-->  6  8  5|
10-->  7  6  1|11-->  8  9  2|12-->  7  9  5|13-->  4  3  1|14-->  2  3  3|
15-->  4  1  1|16-->  3  8  7|17-->  4  2  7|18-->  7  9  3|19-->  1  9  8|


PointID   NN Size    NN NID
 0:         4         [  0  1  4  5]
 1:         2         [  0  1]
 2:         5         [  2  6 10 11 18]
 3:         1         [  3]
 4:         3         [  0  4  5]
 5:         4         [  0  4  5 14]
 6:         3         [  2  6 10]
 7:         5         [  7  8 13 14 15]
 8:         5         [  7  8 13 14 15]
 9:         3         [  9 12 18]
10:         3         [  2  6 10]
11:         3         [  2 11 18]
12:         3         [  9 12 18]
13:         5         [  7  8 13 14 15]
14:         5         [  5  7  8 13 14]
15:         4         [  7  8 13 15]
16:         2         [ 16 19]
17:         1         [ 17]
18:         5         [  2  9 11 12 18]
19:         2         [ 16 19]
```

Figure 2. Output of the stl program

a) Define a structure (i.e., **struct Point**) to store each point coordinates. The structure includes id value indicating the order of the point and x, y, and z coordinates.

b) Define a **global Point vector** with 20 items for **points**.

c) Use **map** to store center pointID and neighbor points in the specified radius. Define a map in the main.

d) Use a generator function (i.e., **Point randPoint (void)**) and appropriate algorithm to initialize point vector. You will generate random numbers between 0 and 9. (**Hint: Do not use loop)**.

e) Print the points by calling the **printVector** function you defined. An example output is shown in the figure. **You must print the vector with the same format in the figure.**

f) Use a predicate function (i.e. **bool radiusNN(Point &)**) and appropriate algorithm or algorithms to get point in the neighborhood. Since this operation must be applied to all points as center points, you can use the loops in the main to search each point neighbors (**Hint: In the loop, the vectors which is used iteratively must be cleared before using again. You can use the global static integer to iterate over seach points**). In the radiusNN function, define a double radius as 0.3 and calculate the distance between search point and the point in the vector. If the dist is greater than radius return **true**, otherwise return **false**.

$$\text{dist} = \sqrt{(searchPx - Px)^2 + (searchPy - Py)^2 + (searchPz - Pz)^2}$$

g) Use the map to hold the pointID and its neighbor points in vector. Print the map by calling the **printMap** function you defined. An example output is shown in the figure. **You must print the map with the same format in the figure (Hint**: **Print the only ID of neighbors point).**