1. **(20%, Intersection of Sets)** Use one-dimensional arrays to solve the following problem. Read in two sets of numbers, each having 5 numbers. After reading all values, display the unique elements common to both sets of numbers.
2. **(20%, Largest row and column)** Write a program that randomly fills in 0s and 1s into a 5-by-5 matrix, prints the matrix, and finds the first row and column with the most 1s.
3. **(20%, Sort two-dimensional array)** Write a function to sort a two-dimensional array using the following header:

void sort (int m[][2], int row);

The function performs a primary sort on rows, and a secondary sort on columns. For example, the following array

{{4, 2}, {1, 7}, {4, 5}, {1, 2}, {1, 1}, {4, 1}}

will be sorted to

{{4, 5}, {4, 2}, {4, 1}, {1, 7}, {1, 2}, {1, 1}}

Write a test program that prompt the user to enter the number of input value-pairs and then enter the value-pair one by one. Then, sort the value pairs by the function sort and display the result.

The following is a sample run:

Enter the number of value-pairs: 3

Input value-pair: 4 2

Input value-pair: 1 7

Input value-paer:4 5

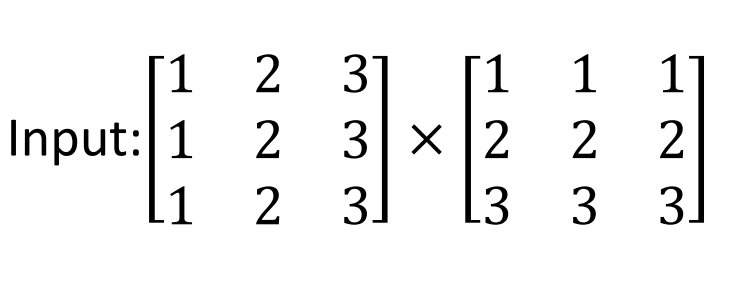
The sorted result is

{{4, 5}, {4, 2}, {1, 7}}

1. **(20%)** An n\*m two-dimensional matrix can be multiplied by another m\*p matrix to give a matrix whose elements are the sum of the products of the elements within a row from the first matrix and the associated elements of a column of the second matrix. Both matrices should either be square matrices, or the number of columns of the first matrix should equal the number of rows of the second matrix.

To calculate each element of the resultant matrix, multiply the first element of a given row from the first matrix and the first element of a given column in the second matrix, add that to the product of the second element of the same row and the second of the same column, and keep doing so until the last elements of the row and column have been multiplied and added to the sum.

Write a program to calculate the product of 2 matrices and store the result in the third matrix.



1. **(20%)** The bean machine is a device for statistics experiments. It consists of an upright board with evenly spaced nails in a triangular form, as shown below:



Fig. 1 Each ball takes a random path and falls into a slot

Balls are dropped from the opening of the board, Every time a ball hits a nail, it has a 50% chance of falling to the left or to the right. The piles of balls are accumulated in the slots at the bottom of the board.

Write a program that simulates the bean machine. Your program should prompt the user to enter the number of the balls and the number of the slots in the machine. Simulate the falling of each ball by printing its path. For example, the path for the ball in the Fig. 1(b) is LLRRLLR and the path for the ball in Figure 1(c) is RLRRLRR. Display the final buildup of the balls in the slots in a histogram. Next page shows a sample run of the program:

Enter the number of balls to drop: 5

Enter the number of slots in the bean machine: 8

LRLRLRR

RRLLLRR

LLRLLRR

RRLLLLL

LRLRRLR

o

o

ooo

*Hint*: Create an array named slots. Each element in slots stores the number of balls in a slot. Each ball falls into a slot via a path. The number of Rs in a path is the position of the slot where the ball falls. For example, for the path LRLRLRR, the ball falls into slots[4], and for the path RRLLLLL, the ball falls into slots[2].