National Taiwan University **Introduction to Machine Learning and Deep Learning**
Department of Civil Engineering Instructor: C.-S. CHEN

Homework 1

Programming, Due 21:00, Wednesday, September 18

No Late Submission

In this homework, we would like to test your Python skills. This course will use Python for the programming homework and projects. We highly recommend testing your Python skills before enrolling in this course. Suppose you encounter difficulty in solving the problems. In that case, we encourage you to solve them by searching for answers on the internet (a very important skill to learn) or discussing it with your friends (again, a very important skill). If you have questions, please join the TA hours listed on the course syllabus.

HW Submission Procedure (請仔細閱讀)

1. 請 submit 至 online judge，逾期不受理。
2. 對於題目以及 OJ 有疑問，請寄信給作業 1 負責的助教(詹承諺 r12521601@ntu.edu.tw, 張瑋哲 r12527a01@ntu.edu.tw)或是直接在 discord 詢問。

Total 100%

1. **(20 points)** Please use the Jupyter notebook `HW1_1.ipynb`. Consider the grade records containing five exams for class 01 and class 02, where class 01 has n students and class 02 has m students. Prompt the online judge for a string containing the random seed, the number of students in class01 (n), and the number of students in class02 (m). Create an array of n by 5 for class 01 and an array of m by 5 for class 02 with randomly generated grades between 40 – 100 (100 included). Output the mean and standard deviation with two decimal places for class01 and class02. Output the mean and standard deviation with two decimal places of each exam for all students.

Input

A string: ‘s, n, m’ where s is the random seed, n is the number of students in class01, and m is the number of students in class02.

Output

A n by 5 array (grade records for class01)
A m by 5 array (grade records for class02)
A (n+m) by 5 array (grade records for both classes)
The mean and standard deviation with two decimal places for class01 and class02
The mean and standard deviation of each exam with two decimal places for all students.

Sample Input 1:

```
42, 3, 2
```

Sample Output 1:

```
Grades for class01 are:
[[ 78  91  68  54  82]
 [ 47 100  60  78  97]
 [ 58  62  50  50  63]]
Grades for class02 are:
[[92 75 79 63 42]
 [61 92 41 63 83]]
Grades for both classes are:
[[ 78  91  68  54  82]
 [ 47 100  60  78  97]
 [ 58  62  50  50  63]
 [ 92  75  79  63  42]
 [ 61  92  41  63  83]]
The mean and std for class01 are: 69.20, 16.89
The mean and std for class02 are: 69.10, 17.43
The mean and std for exam01 are: 67.20, 15.89
The mean and std for exam02 are: 84.00, 13.67
The mean and std for exam03 are: 59.60, 13.31
The mean and std for exam04 are: 61.60, 9.65
The mean and std for exam05 are: 73.40, 19.06
```

Sample Input 2:

```
6, 4, 3
```

Sample Output 2:

```
Grades for class01 are:
[[50 49 75 60 82]
 [85 55 82 56 65]
 [41 51 53 97 66]
 [87 73 84 86 44]]
Grades for class02 are:
[[73 48 42 52 60]
 [67 71 42 45 66]
 [55 65 62 90 83]]
Grades for both classes are:
[[50 49 75 60 82]
 [85 55 82 56 65]
 [41 51 53 97 66]
 [87 73 84 86 44]
 [73 48 42 52 60]
 [67 71 42 45 66]
 [55 65 62 90 83]]
The mean and std for class01 are: 67.05, 16.45
The mean and std for class02 are: 61.40, 13.93
The mean and std for exam01 are: 65.43, 16.26
The mean and std for exam02 are: 58.86, 9.83
The mean and std for exam03 are: 62.86, 16.60
The mean and std for exam04 are: 69.43, 19.37
The mean and std for exam05 are: 66.57, 12.33
```

2. **(20 points)** Please use the Jupyter Notebook `HW1_2.ipynb`. We have learned to use 2D list to represent a matrix. This representation makes sense if a matrix is full of nonzero values. However, for a sparse matrix (i.e. a matrix that is comprised of mostly zeros), like the one below:

   A list may not be the most efficient way to represent the matrix since it contains a lot of zeros.

$$\begin{bmatrix} 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 \\ 0 & 0 & 6 & 0 & 0 \end{bmatrix}$$

   An alternative is to use dictionary, where the row and column indexes can be joined as tuples and used as the keys, and we only have to record nonzero values. A dictionary representation of the above matrix can be obtained:

```
{(0,3):5,(2,1):10,(3,4):6,(4,2):6}
```

   Please write a program for users to input the dimensions (i.e. the numbers of rows and columns) of a matrix and the number of nonzero integers within this matrix. Please parse the input and first use a dictionary to store these nonzero integers and their locations, and then reconstruct a full matrix based upon these input numbers (note: each item takes up 2-slot space, and space is used to separate items from the same row).

Input

nrows,ncols,nNonZeros
row_idx1,col_idx1,nonzero1
row_idx2,col_idx2,nonzero2
...
row_idxN,col_idxN,nonzeroN

Output

A dictionary with nonzero integers and their locations
A full matrix with both zero and nonzero integers

Sample Input 1:

```
5,5,4
0,3,5
2,1,10
3,4,6
4,2,6
```

Sample Output 1:

```
{(0, 3): 5, (2, 1): 10, (3, 4): 6, (4, 2): 6}
```

```
 0  0  0  5  0
 0  0  0  0  0
 0 10  0  0  0
 0  0  0  0  6
 0  0  6  0  0
```

Sample Input 2:

```
5,6,5
0,2,1
2,1,99
4,3,6
3,5,5
0,0,45
```

Sample Output 2:

```
{(0, 2): 1, (2, 1): 99, (4, 3): 6, (3, 5): 5, (0, 0): 45}
45  0  1  0  0  0
 0  0  0  0  0  0
 0 99  0  0  0  0
 0  0  0  0  0  5
 0  0  0  6  0  0
```

3. **(20 points)** Please use the Jupyter notebook `HW1_3.ipynb`. Create a n-by-m matrix containing the even integers from 2 to (n×m)×2. Create a second m-by-2 matrix containing the random integers of 4 or 9 with a given random seed s. Print these two matrices and their matrix multiplication. (hint: for the second matrix, please generate a matrix with a random number of 0 or 1 and use a mask to change them to 4 or 9)

Input:

n,m,s

Output:

A n-by-m matrix containing the even integers from 2 through (n×m)×2
A m-by-2 matrix containing the random integers of 4 or 9 with a given random seed s
A resulting n-by-2 matrix

Sample Input 1:

```
3,5,12
```

Sample Output 1:

```
The first matrix is:
[[ 2  4  6  8 10]
 [12 14 16 18 20]
 [22 24 26 28 30]]
The second matrix is:
```

```
[[9 9]
 [4 9]
 [9 4]
 [9 9]
 [4 4]]
The multiplication results of the first and second matrices are:
[[200 190]
 [550 540]
 [900 890]]
```

Sample Input 2:

```
1,6,100
```

Sample Output 2:

```
The first matrix is:
[[ 2  4  6  8 10 12]]
The second matrix is:
[[4 4]
 [9 9]
 [9 9]
 [4 4]
 [4 4]
 [4 9]]
The multiplication results of the first and second matrices are:
[[218 278]]
```

4. **(20 points)** Please use the Jupyter notebook `HW1_4.ipynb`. You are to use NumPy's random module with a given seed to generate random integers, ranging from 0 to 120 ($0 \leq x < 120$), in a 1D array of size n. Adjust the grade records below 40 to 40 and above 100 to 100. Then, change the shape of the 1D array into an r by c 2D array. Last, you are to find those rows with a mean larger than the threshold m and report the 2D array and the sum of those rows. You should report dimension inconsistency if the 1D array cannot be converted into an r by c 2D array.

Input

A string containing the following information:

- n (size of the 1D array)
- r (number of rows)
- c (number of columns)
- m (an integer threshold)
- seed (random seed)

Output

You should report 'Dimension is inconsistent' or
- the 2D array
- the sum of those rows with a mean larger than m

Sample Input 1:

```
12, 3, 3, 60, 0
```

Sample Output 1:

```
Dimension is inconsistent
```

Sample Input 2:

```
6, 2, 3, 80, 42
```

Sample Output 2:

```
The adjusted 2D array
[[100  51  92]
 [ 40 100  71]]
The sum of those rows with a mean larger than 80
[243]
```

Sample Input 3:

```
12, 4, 3, 60, 0
```

Sample Output 3:

```
The adjusted 2D array
[[ 44  47 100]
 [ 64  67  67]
 [100  40  83]
 [ 40 100  40]]
The sum of those rows with a mean larger than 60
[191 198 223]
```

5. **(20 points)** Please use the Jupyter notebook `HW1_5.ipynb`. Write a program that performs polynomial simplification and polynomial multiplication. A polynomial is defined as:

   $P(x) = a_n x^n + \cdots + a_1 x + a_0$

   For simplicity, consider the coefficients a_i as integers and the power $n \geq 0$. Use a dictionary to represent a polynomial that only saves non-zero terms. The first line of input will be several (power, coefficient) pairs, which represent the non-zero terms of $P_1(x)$, and the second line represents $P_2(x)$. Note that input may not be the polynomial after simplifying; that is, the same power may occur more than once and you should sum them all. Please output the non-zero terms of $P_1(x)$, $P_2(x)$ and the result of $P_1(x) \times P_2(x)$ in a dictionary form. If the polynomial is a zero-polynomial, please output 0 directly.

Input

(power,coefficient) pairs of $P_1(x)$ without simplifying
(power,coefficient) pairs of $P_2(x)$ without simplifying

Output

A dictionary with non-zero terms of $P_1(x)$ in ascending order
A dictionary with non-zero terms of $P_2(x)$ in ascending order
A dictionary with non-zero terms of $P_1(x) \times P_2(x)$ in ascending order

Sample Input 1:

```
(0,1) (0,2)
(0,3) (1,4)
```

Sample Output 1:

```
The first polynomial after simplification: {0: 3}
The second polynomial after simplification: {0: 3, 1: 4}
The product of two polynomials: {0: 9, 1: 12}
```

Sample Input 2:

```
(0,1) (2,3) (4,5) (4,-7) (2,5)
(10,9) (8,7) (6,5) (6,-5) (7,-1)
```

Sample Output 2:

```
The first polynomial after simplification: {0: 1, 2: 8, 4: -2}
The second polynomial after simplification: {7: -1, 8: 7, 10: 9}
The product of two polynomials: {7: -1, 8: 7, 9: -8, 10: 65, 11: 2, 12:
58, 14: -18}
```

Sample Input 3:

```
(0,1) (2,3) (4,5) (4,-5) (2,-3) (0,-1)
(0,1) (1,2) (2,3) (3,4) (4,5)
```

Sample Output 3:

```
The first polynomial after simplification: 0
The second polynomial after simplification: {0: 1, 1: 2, 2: 3, 3: 4, 4: 5}
The product of two polynomials: 0
```