



**SCHOOL OF MECHANICAL & MANUFACTURING ENGINEERING**

**PROGRAM: MS IN ROBOTICS & INTELLIGENT MACHINE ENGINEERING**

Course Code	CSE - 860
Course Title	Artificial Intelligence
Semester/Year	1 <sup>st</sup> / Fall - 2023

Instructor	Dr. Yasar Ayaz
------------	----------------

<b>ASSIGNMENT #</b>	<b>03</b>
---------------------	-----------

Assignment Title	PYTHON
------------------	--------

Student Name	Hamza Ahmad Shahzad
Student Registration #	450037

## Difficulty Level (Medium)

1.

The screenshot shows the HackerRank 'Piling Up' problem page. The problem description states: 'There is a horizontal row of  $n$  cubes. The length of each cube is given. You need to create a new vertical pile of cubes. The new pile should follow these directions: if  $cube[i]$  is on top of  $cube[j]$  then  $sideLength[j] \geq sideLength[i]$ . When stacking the cubes, you can only pick up either the leftmost or the rightmost cube each time. Print Yes if it is possible to stack the cubes. Otherwise, print No.

**Example**  
 $blocks = [1, 2, 3, 8, 7]$   
Result: No

After choosing the rightmost element, 7, choose the leftmost element, 1. After that, the choices are 2 and 8. These are both larger than the top block of size 1.  
 $blocks = [1, 2, 3, 7, 8]$   
Result: Yes

Choose blocks from right to left in order to successfully stack the blocks.

**Input Format**  
The first line contains a single integer  $T$ , the number of test cases.

The code editor shows a Python solution:

```
1 import sys
2
3 def test_cubes(cubes):
4     t_cube = 0
5
6     if cubes[0] > cubes[len(cubes)-1]:
7         t_cube = cubes[0]
8         cubes.pop(0)
9     else:
10        t_cube = cubes[len(cubes)-1]
11        cubes.pop(len(cubes)-1)
12
13    while len(cubes) > 0:
14        if t_cube == cubes[0]:
15            t_cube = cubes.pop(0)
16        elif t_cube == cubes[len(cubes)-1]:
17            t_cube = cubes.pop(len(cubes)-1)
18        elif (cubes[0] > cubes[len(cubes)-1]) and (t_cube >= cubes[0]):
19            t_cube = cubes.pop(0)
20        elif (cubes[0] < cubes[len(cubes)-1]) and (t_cube >= cubes[len(cubes)-1]):
21            t_cube = cubes.pop(len(cubes)-1)
22
23    return True
24
25 if __name__ == '__main__':
26     T = int(input())
27     for _ in range(T):
28         cubes = list(map(int, input().split()))
29         result = test_cubes(cubes)
30         print('Yes' if result else 'No')
```

Figure 1: Piling Up

2.

The screenshot shows the HackerRank 'Write a Function' problem page. The problem description states: 'years: The year can be evenly divided by 4, is a leap year, unless: The year can be evenly divided by 100, it is NOT a leap year, unless: The year is also evenly divisible by 400. Then it is a leap year. This means that in the Gregorian calendar, the years 2000 and 2400 are leap years, while 1800, 1900, 2100, 2200, 2300 and 2500 are NOT leap years. Source

**Task**  
Given a year, determine whether it is a leap year. If it is a leap year, return the Boolean True, otherwise return False.

Note that the code stub provided reads from STDIN and passes arguments to the `is_leap` function. It is only necessary to complete the `is_leap` function.

**Input Format**  
Read `year`, the year to test.

**Constraints**  
 $1900 \leq year \leq 10^5$

**Output Format**

The code editor shows a Python solution:

```
1 def is_leap(year):
2     leap = False
3
4     if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
5         leap = True
6
7     return leap
8
9 > year = int(input()) ...
```

Figure 2: Write a Function

3.

**Problem**

Kevin and Stuart want to play the 'The Minion Game'.

**Game Rules**

Both players are given the same string,  $S$ .

Both players have to make substrings using the letters of the string  $S$ .

Stuart has to make words starting with consonants.

Kevin has to make words starting with vowels.

The game ends when both players have made all possible substrings.

**Scoring**

A player gets +1 point for each occurrence of the substring in the string  $S$ .

**For Example:**

String  $S = \text{BANANA}$

Kevin's vowel beginning word = ANA

Here, ANA occurs twice in BANANA. Hence, Kevin will get 2 Points.

For better understanding, see the image below:

**Code Editor:**

```
def minion_game(string):
    k=0
    s=0
    vowels="AaEeIiOoUu"
    for i in range(len(string)):
        if string[i] in vowels:
            k=k+len(string)-i
        else:
            s=s+len(string)-i

    if k>s:
        print("Kevin",k)
    elif k==s:
        print("Draw")
    else:
        print("Stuart",s)

if __name__ == '__main__': ...
```

Figure 3: The Minion Game

4.

**Problem**

Consider the following:

- A string,  $s$ , of length  $n$  where  $s = c_0c_1 \dots c_{n-1}$ .
- An integer,  $k$ , where  $k$  is a factor of  $n$ .

We can split  $s$  into  $\frac{n}{k}$  substrings where each substring,  $t_i$ , consists of a contiguous block of  $k$  characters in  $s$ . Then, use each  $t_i$  to create string  $u_i$  such that:

- The characters in  $u_i$  are a subsequence of the characters in  $t_i$ .
- Any repeat occurrence of a character is removed from the string such that each character in  $u_i$  occurs exactly once. In other words, if the character at some index  $j$  in  $t_i$  occurs at a previous index  $< j$  in  $t_i$ , then do not include the character in string  $u_i$ .

Given  $s$  and  $k$ , print  $\frac{n}{k}$  lines where each line  $i$  denotes string  $u_i$ .

**Example**

$s = \text{'AAABBCADDE'}$

$k = 3$

There are three substrings of length 3 to consider: 'AAA', 'BCA' and 'DDE'. The first substring is all 'A' characters, so  $u_1 = \text{'A'}$ . The second substring has all distinct characters, so  $u_2 = \text{'BCA'}$ . The third substring has 2 different characters, so  $u_3 = \text{'DE'}$ . Note that a subsequence maintains the original order of characters encountered. The order of

**Code Editor:**

```
def merge_the_tools(string, k):
    split_string=(string[i:i+k] for i in range(0,len(string),k))
    for i in split_string:
        u=[]
        u.append(i[0])
        for j in range(1,len(i)):
            if i[j] in u:
                continue
            else:
                u.append(i[j])
        print(''.join(str(e) for e in u))

if __name__ == '__main__': ...
```

Figure 4: Merge the Tools!

5.

hackerrank.com/challenges/python-time-delta/problem?isFullScreen=true

HackerRank | Prepare > Python > Date and Time > Time Delta

The first line contains  $T$ , the number of testcases.  
Each testcase contains 2 lines, representing time  $t_1$  and time  $t_2$ .

**Constraints**

- Input contains only valid timestamps
- $year \leq 3000$ .

**Output Format**

Print the absolute difference ( $t_1 - t_2$ ) in seconds.

**Sample Input 0**

```
2
Sun 10 May 2015 13:54:36 -0700
Sun 10 May 2015 13:54:36 -0000
Sat 02 May 2015 19:54:36 +0530
Fri 01 May 2015 13:54:36 -0000
```

**Sample Output 0**

```
25200
88200
```

**Explanation 0**

Change Theme Language: Python 3

```
1 #!/bin/python3
2
3 import math
4 import os
5 import random
6 import re
7 import sys
8
9 # Complete the time_delta function below.
10 from datetime import datetime
11
12 if __name__ == '__main__':
13     t = int(input())
14     for _ in range(t):
15         s1 = input()
16         s2 = input()
17         t1 = datetime.strptime(s1, "%a %d %b %Y %H:%M:%S %z")
18         t2 = datetime.strptime(s2, "%a %d %b %Y %H:%M:%S %z")
19         print(abs(int((t1-t2).total_seconds())))
```

Line: 19 Col: 49

Upload Code as File Test against custom input Run Code Submit Code

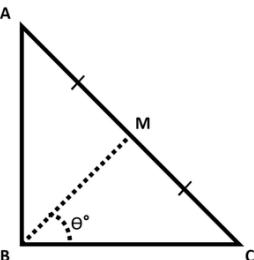
Figure 5: Time Delta

6.

hackerrank.com/challenges/find-angle/problem?isFullScreen=true

HackerRank | Prepare > Python > Math > Find Angle MBC

**Problem**



$ABC$  is a right triangle,  $90^\circ$  at  $B$ .  
Therefore,  $\angle ABC = 90^\circ$ .  
Point  $M$  is the midpoint of hypotenuse  $AC$ .  
You are given the lengths  $AB$  and  $BC$ .  
Your task is to find  $\angle MBC$  (angle  $\theta$ , as shown in the figure) in degrees.

**Input Format**

Change Theme Language: Python 3

```
1 import math
2
3 ab=int(input())
4 bc=int(input())
5
6 ca=math.hypot(ab,bc)
7 mc=ca/2
8 bca=math.asin(1*ab/ca)
9 bm=math.sqrt((bc**2+mc**2)-(2*bc*mc*math.cos(bca)))
10 mbc=math.asin(math.sin(bca)*mc/bm)
11
12 print(int(round(math.degrees(mbc),0)),'\u00B0',sep='')
```

Line: 12 Col: 55

Upload Code as File Test against custom input Run Code Submit Code

Figure 6: Find Angle MBC

7.

**Problem**

There is an array of  $n$  integers. There are also 2 disjoint sets,  $A$  and  $B$ , each containing  $m$  integers. You like all the integers in set  $A$  and dislike all the integers in set  $B$ . Your initial happiness is 0. For each  $i$  integer in the array, if  $i \in A$ , you add 1 to your happiness. If  $i \in B$ , you add  $-1$  to your happiness. Otherwise, your happiness does not change. Output your final happiness at the end.

**Note:** Since  $A$  and  $B$  are sets, they have no repeated elements. However, the array might contain duplicate elements.

**Constraints**

- $1 \leq n \leq 10^5$
- $1 \leq m \leq 10^5$
- $1 \leq \text{Any integer in the input} \leq 10^9$

**Input Format**

The first line contains integers  $n$  and  $m$  separated by a space.  
 The second line contains  $n$  integers, the elements of the array.  
 The third and fourth lines contain  $m$  integers,  $A$  and  $B$ , respectively.

**Output Format**

Output a single Integer, your total happiness.

**Submissions**

**Leaderboard**

**Code Editor:**

```

1 n,m=list(map(int, input().split()))
2 ns=list(map(int, input().split()))
3 h=set(map(int, input().split()))
4 s=set(map(int, input().split()))
5 res=0
6 for x in ns:
7     if x in h:
8         res+=1
9     elif x in s:
10        res-=1
11 print(res)
  
```

Line: 11 Col: 11

Upload Code as File Test against custom input Run Code Submit Code

Figure 7: No Idea!

8.

**Problem**

You are given  $n$  words. Some words may repeat. For each word, output its number of occurrences. The output order should correspond with the input order of appearance of the word. See the sample input/output for clarification.

**Note:** Each input line ends with a "\n" character.

**Constraints:**

- $1 \leq n \leq 10^5$
- The sum of the lengths of all the words do not exceed  $10^6$
- All the words are composed of lowercase English letters only.

**Input Format**

The first line contains the integer,  $n$ .  
 The next  $n$  lines each contain a word.

**Output Format**

Output 2 lines.  
 On the first line, output the number of distinct words from the input.  
 On the second line, output the number of occurrences for each distinct word according to their appearance in the input.

**Sample Input**

**Code Editor:**

```

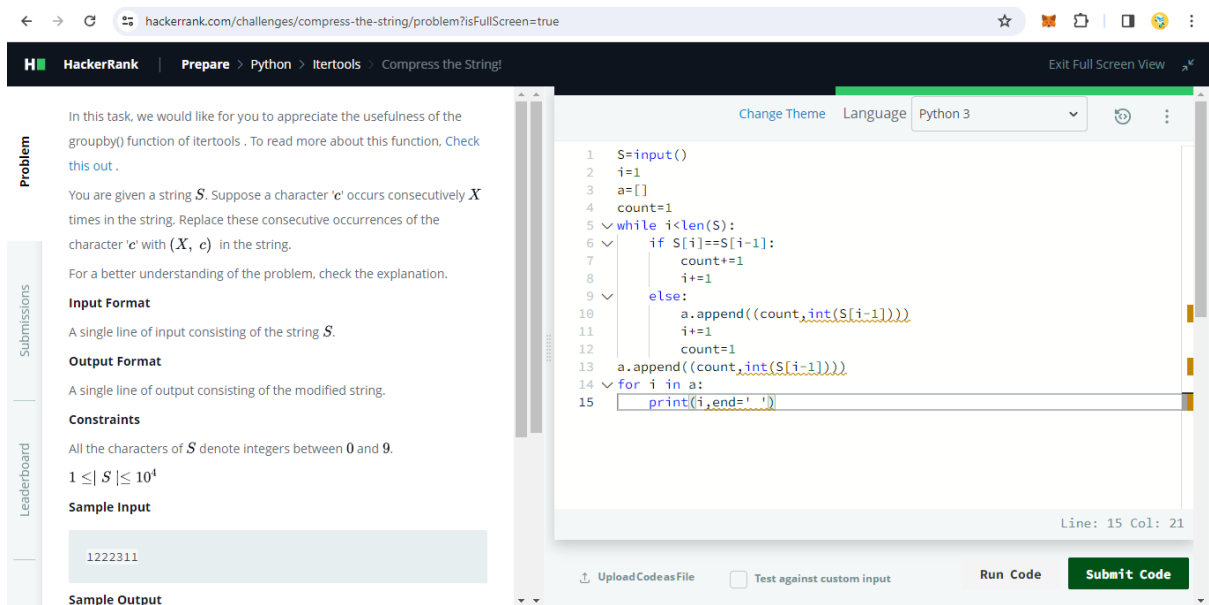
1 from collections import OrderedDict
2
3 d=OrderedDict()
4 n=int(input())
5 for i in range(n):
6     s=input()
7     if s in d.keys():
8         d[s]+=1
9     else:
10        d[s]=1
11 print(len(d.keys()))
12 print(' '.join([str(d[k]) for k in d.keys()]))
  
```

Line: 12 Col: 47

Upload Code as File Test against custom input Run Code Submit Code

Figure 8: Word Order

9.



**Problem**

In this task, we would like for you to appreciate the usefulness of the `groupby()` function of `itertools`. To read more about this function, [Check this out](#).

You are given a string  $S$ . Suppose a character 'c' occurs consecutively  $X$  times in the string. Replace these consecutive occurrences of the character 'c' with  $(X, c)$  in the string.

For a better understanding of the problem, check the explanation.

**Input Format**

A single line of input consisting of the string  $S$ .

**Output Format**

A single line of output consisting of the modified string.

**Constraints**

All the characters of  $S$  denote integers between 0 and 9.

$1 \leq |S| \leq 10^4$

**Sample Input**

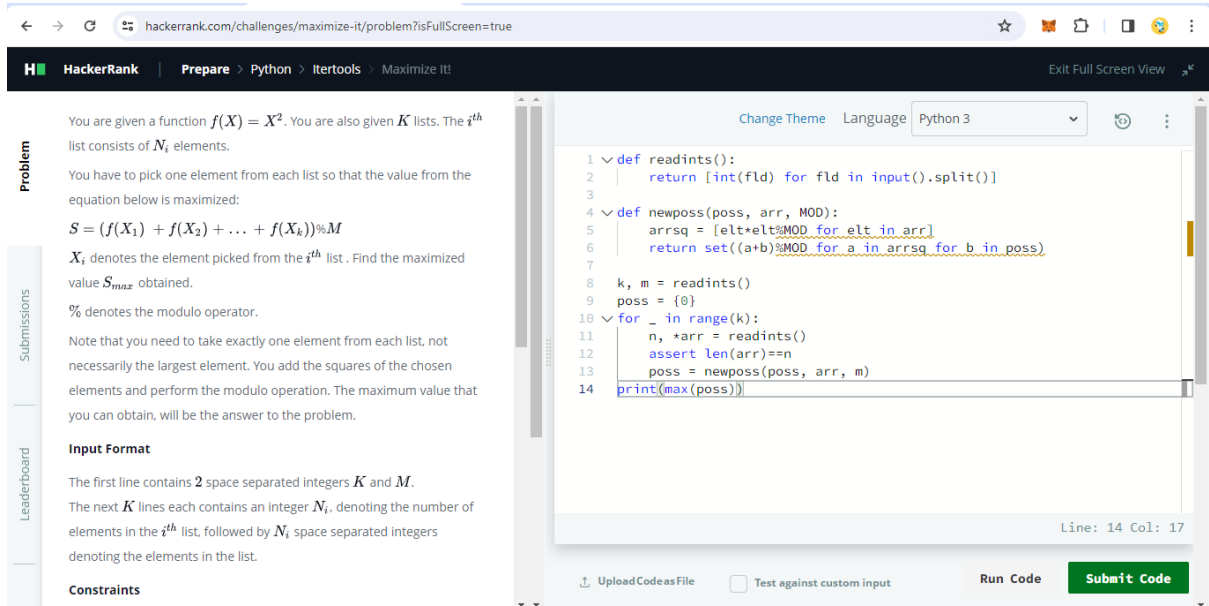
```
1222311
```

**Sample Output**

```
(3,2)311
```

Figure 9: Compress the String!

## Difficulty Level (Hard)



**Problem**

You are given a function  $f(X) = X^2$ . You are also given  $K$  lists. The  $i^{th}$  list consists of  $N_i$  elements.

You have to pick one element from each list so that the value from the equation below is maximized:

$$S = (f(X_1) + f(X_2) + \dots + f(X_K)) \% M$$

$X_i$  denotes the element picked from the  $i^{th}$  list. Find the maximized value  $S_{max}$  obtained.

$\%$  denotes the modulo operator.

Note that you need to take exactly one element from each list, not necessarily the largest element. You add the squares of the chosen elements and perform the modulo operation. The maximum value that you can obtain, will be the answer to the problem.

**Input Format**

The first line contains 2 space separated integers  $K$  and  $M$ .

The next  $K$  lines each contains an integer  $N_i$ , denoting the number of elements in the  $i^{th}$  list, followed by  $N_i$  space separated integers denoting the elements in the list.

**Constraints**

Figure 10: Maximize It!