Problem 1:

In one pass, **Train A** can start from the source station at time **T[0]**, halt at each station for **h** unit of time until it reaches the last station at time **T[N – 1]**, where **N** is the positive integer representing a total number of stations.

Given, **Train A's** timings at each unit of time as **T[] = {10.00, 10.04, 10.09, 10.15, 10.19, 10.22}**.

Now, suppose Railway Admin wants to add more trains to increase the frequency. So, to launch other **Train B**, for the same stations as of **Train A's**. Provided the **Train B** starts at time **t**, they would like to know the timings for **Train B**. The program should return a String array **S** (timestamp(in float) for **Train B** at each station from first to the last station like **train A**).

Note:

- The time is represented in 24-Hour.
- Start Hour should be in the range **[0, 23]**.
- Start Minute should be in the range **[0, 59]**.
- Enter start time(24 Hrs)

Examples:

Input: *t = 11.00*
Output: *11.00 11.04 11.09 11.15 11.19 11.22*
Explanation: *Start time for train B is 11.00 and also the time difference between the stations for train B is same as for train A.*
Input: *t = -26.15*
Output: *Invalid Input*
Explanation: *No such time as -26.15 exists. Hence, print "Invalid Input".*

Approach: The idea is to calculate the time differences between the stations from the given timings of Train A. Follow the steps below to solve the problem:

- From the given array **T[]**, generate an array **train_B[]** where **train_B[i]** is the time difference between **T[i]** and **T[i – 1]**, where **train_B[0] = 0.00** and **1 ≤ i ≤ 5.**

- Therefore, **train_B[] = {0.00, 0.04, 0.05, 0.06, 0.04, 0.03}**.
- If the integer part of **t** is not in the range **[0, 24]** or the decimal part of **t** is not in the range **[0, 60]**, then print **"Invalid Input"** because the integer part represents the hours and the decimal part represents the minutes.
- Otherwise, traverse over the range **[0, 5]** and print **t + train_B[i]** denoting the time for **train B** for the **i^{th}** station. Then update t as **t = t + train_B[i]**.

Below is the implementation of the above approach:

```
# Python3 program for the above approach  Function to find the timings for  train B
#having same time difference  as train_A
def findTime(train_A, N, t):

    # Stores the time for train_B
    train_B = [None] * N
    train_B[0] = 0.00;

    for i in range(1, N):
        train_B[i] = train_A[i] - train_A[i - 1];

    # Variables for typecasting
    it = int(t);

    # Check if t is valid
    if (t >= 0.0 and t <= 24.0 and (t - it) <= 60.0) :

        # Traverse from 0 to 5
        for i in range(6):

            # Update t
            x = t + train_B[i];
            ix = int(x);

            if (x - ix >= 0.60):
                x = x + 0.40;
            if (x > 24.00):
                x = x - 24.0;

            # Print the current time
            print(f"{x:.2f}", end = " ")
```

```
            t = x;

    # If no answer exist
    else:
        print("Invalid Input");

# Driver Code

# Given timings of train A
# at each station
train_A = [ 10.00, 10.04, 10.09, 10.15, 10.19, 10.22 ];

N = len(train_A);

# Given start time t
t = 11.00;

# Function Call
findTime(train_A, N, t);
```

Problem 2:

Consider a long alley with **N** doors on one side. All the doors are closed initially. You move to and fro in the alley changing the states of the doors as follows:

- You open a door that is already closed, and you close a door that is already opened.
- You start at one end go on altering the state of the doors till you reach the other end, and then you come back and start altering the states of the doors again.
- In the first go, you alter the states of doors numbered **1, 2, 3, …, N**.
- In the second go, you alter the states of doors numbered **2, 4, 6, …**.
- In the third go, you alter the states of doors numbered **3, 6, 9 …**.

  and so on…

The above procedure will continue till the **N^{th} turn** in which you alter the state of the door numbered **N**. The task is to find the number of open doors at the end of the procedure.

Examples:

Input: *N = 3*
Output: *1*

Input: *N = 100*
Output: *10*

```
# Python3 code for the above approach
import math

# Function that counts the number of
# doors opens after the Nth turn
def countOpenDoors(N):

    # Find the number of open doors
    doorsOpen = int(math.sqrt(N))

    # Return the resultant count of
    # open doors
    return doorsOpen

# Driver Code
if __name__ == '__main__':
    N = 5
    print(countOpenDoors(N))

# This code is contributed by MuskanKalra1
```

Problem 3:

write a Python program to find out all possible permutations of a string.

Method 1:

```
ini_str = "abc"
print("Initial string", ini_str)
result = []
```

```
def permute(data, i, length):
    if i == length:
        result.append(''.join(data) )
    else:
        for j in range(i, length):
            # swap
            data[i], data[j] = data[j], data[i]
            permute(data, i + 1, length)
            data[i], data[j] = data[j], data[i]
permute(list(ini_str), 0, len(ini_str))

print("Resultant permutations", str(result))
```

Method #2: Using itertools

```
from itertools import permutations
ini_str = "abc"
print("Initial string", ini_str)
permutation = [''.join(p) for p in permutations(ini_str)]
print("Resultant List", str(permutation))
```

Problem 4:

Given an **N * N** binary maze where a **0** denotes that the position can be visited and a **1** denotes that the position cannot be visited without a key, the task is to find whether it is possible to visit the bottom-right cell from the top-left cell with only one key along the way. If possible then print **"Yes"** else print **"No"**.

Example:

Input: *maze[][] = {*
{0, 0, 1},
{1, 0, 1},
{1, 1, 0}}

Output: *Yes*

Approach: This problem can be solved using recursion, for every possible move, if the current cell is **0** then without altering the status of the key check whether it is

the destination else move forward. If the current cell is **1** then the key must be used, now for the further moves the key will be set to **false** i.e. it'll never be used again on the same path. If any path reaches the destination then print **Yes** else print **No**. Below is the implementation of the above approach:

```
# Python3 implementation of the approach

# Recursive function to check whether there is
# a path from the top left cell to the
# bottom right cell of the maze

def findPath(maze, xpos, ypos, key):

    # Check whether the current cell is
    # within the maze
    if xpos < 0 or xpos >= len(maze) or ypos < 0 \
            or ypos >= len(maze):
        return False

    # If key is required to move further
    if maze[xpos][ypos] == '1':

        # If the key hasn't been used before
        if key == True:

            # If current cell is the destination
            if xpos == len(maze)-1 and ypos == len(maze)-1:
                return True

            # Either go down or right
            return findPath(maze, xpos + 1, ypos, False) or \
                findPath(maze, xpos, ypos + 1, False)

        # Key has been used before
        return False
```

```
    # If current cell is the destination
    if xpos == len(maze)-1 and ypos == len(maze)-1:
        return True

    # Either go down or right
    return findPath(maze, xpos + 1, ypos, key) or \
        findPath(maze, xpos, ypos + 1, key)


def mazeProb(maze, xpos, ypos):
    key = True
    if findPath(maze, xpos, ypos, key):
        return True
    return False


# Driver code
if __name__ == "__main__":

    maze = [['0', '0', '1'],
            ['1', '1', '1'],
            ['1', '1', '0']]
    n = len(maze)

    # If there is a path from the cell (0, 0)
    if mazeProb(maze, 0, 0):
        print("Yes")
    else:
        print("No")
```

#Method 2:

```
# Python implementation of the approach
```

```
def mazeProb(maze, n):

   for row in range(n):
      for col in range(n):

         if (row == 0 and col == 0): # Skip the first cell
            continue
         if (row == 0):
           # for first row result depend on previous col
            maze[row][col] = min(
               2, maze[row][col] + maze[row][col - 1])
         elif (col == 0):
           # for first col result depends on previous row
            maze[row][col] = min(
               2, maze[row][col] + maze[row - 1][col])
         else:
           # for other cells, result will be
           # minimum of previous row or col cell
            maze[row][col]  = min(2, maze[row][col]  + min(maze[row][col - 1],
maze[row - 1][col]))

   return maze[n - 1][n - 1]!= 2 # if last cell value is 2 then there is no
            # path available

# Driver code
maze = [ [ 0, 0, 1 ], [ 1, 0, 1 ], [ 1, 1, 0 ] ]
n = len(maze)
# If there is a path from the cell (0, 0)
if (mazeProb(maze, 3)):
   print("Yes")
else:
   print("No")
```

Problem 5:

Given two integers **A** and **M**, find the modular multiplicative inverse of **A** under modulo **M**.
The modular multiplicative inverse is an integer **X** such that:
A X ≅ 1 (mod M)

Note: The value of **X** should be in the range {1, 2, … M-1}, i.e., in the range of integer modulo **M**. (Note that **X** cannot be 0 as A*0 mod M will never be 1). The multiplicative inverse of "A modulo M" exists if and only if A and M are relatively prime (i.e. if gcd(A, M) = 1)
Input: *A = 3, M = 11*
Output: *4*

Explanation: *Since (4*3) mod 11 = 1, 4 is modulo inverse of 3(under 11).*
*One might think, 15 also as a valid output as "(15*3) mod 11"*
is also 1, but 15 is not in range {1, 2, … 10}, so not valid.

Input: *A = 10, M = 17*
Output: *12*

Explanation: *Since (10*12) mod 17 = 1, 12 is modulo inverse of 10(under 17).*

```
# Python3 program to find modular
# inverse of A under modulo M

# A naive method to find modulor
# multiplicative inverse of A
# under modulo M

def modInverse(A, M):

    for X in range(1, M):
        if (((A % M) * (X % M)) % M == 1):
            return X
    return -1



# Driver Code
```

```
if __name__ == "__main__":
    A = 3
    M = 11

    # Function call
    print(modInverse(A, M))
```

Problem 6:

Given an array of integers (both odd and even), sort them in such a way that the first part of the array contains odd numbers sorted in descending order, rest portion contains even numbers sorted in ascending order.

Examples:
Input: *arr[] = {1, 2, 3, 5, 4, 7, 10}*
Output: *arr[] = {7, 5, 3, 1, 2, 4, 10}*
Input: *arr[] = {0, 4, 5, 3, 7, 2, 1}*
Output: *arr[] = {7, 5, 3, 1, 0, 2, 4}*

```
def twoWaySort(arr, n):
    # To store odd Numbers
    odd = []
    # To store Even Numbers
    even = []
    for i in range(n):
        # If number is even push them to even vector
        if arr[i] % 2 == 0:
            even.append(arr[i])
        # If number is odd push them to odd vector
        else:
            odd.append(arr[i])
    # Sort even array in ascending order
    even.sort()
    # Sort odd array in descending order
    odd.sort(reverse=True)
```

```
    i = 0
    # First store odd numbers to array
    for j in range(len(odd)):
        arr[i] = odd[j]
        i += 1
    # Then store even numbers to array
    for j in range(len(even)):
        arr[i] = even[j]
        i += 1


arr = [1, 3, 2, 7, 5, 4]
n = len(arr)
twoWaySort(arr, n)
for i in range(n):
    print(arr[i], end=" ")
```

Problem 7:

Given an array of size n, arrange the first k elements of the array in ascending order and the remaining n-k elements in descending order.

Examples:

Input: *arr[] = {5, 4, 6, 2, 1, 3, 8, 9, -1}, k = 4*

Output: *2 4 5 6 9 8 3 1 -1*

Input: *arr[] = {5, 4, 6}, k = 2*

Output: *4 5 6*

Algorithm:

1. Store the first k elements in an array and sort that in ascending order.
2. Store the remaining n-k elements in an array and sort that in descending order.
3. Merge the two arrays by adding the elements from the second array in reverse order.

```
# Python 3 program to sort first
# k elements in increasing order
```

```
# and remaining n-k elements in
# decreasing

# Function to sort the array
def printOrder(arr, n, k):

    len1 = k
    len2 = n - k
    arr1 = [0] * k
    arr2 = [0] * (n - k)

    # Store the k elements
    # in an array
    for i in range(k):
        arr1[i] = arr[i]

    # Store the remaining n-k
    # elements in an array
    for i in range(k, n):
        arr2[i - k] = arr[i]

    # sorting the array from
    # 0 to k-1 places
    arr1.sort()

    # sorting the array from
    # k to n places
    arr2.sort()

    # storing the values in the
    # final array arr
    for i in range(n):
        if (i < k):
            arr[i] = arr1[i]
```

```
        else :
            arr[i] = arr2[len2 - 1]
            len2 -= 1

    # printing the array
    for i in range(n):
        print(arr[i], end = " ")

# Driver code
if __name__ == "__main__":
    arr = [ 5, 4, 6, 2, 1,
            3, 8, 9, -1 ]
    k = 4

    n = len(arr)

    printOrder(arr, n, k)
```

Efficient Approach: The idea is simple, sort the first k elements in increasing order and remaining n-k elements in decreasing using library function.

```
# Python3 program to sort first k elements  in increasing order and remaining  n-k elements in decreasing

# function to sort the array
def printOrder(arr, n, k):

    # Sort first k elements in ascending order
    a = arr[0:k];
    a.sort();

    # Sort remaining n-k elements in descending order
    b = arr[k:n];
    b.sort();
```

```
    b.reverse();
    return a + b;

# Driver code
arr = [ 5, 4, 6, 2, 1, 3, 8, 9, -1 ];
k = 4;
n = len(arr);

arr = printOrder(arr, n, k);

for i in range(n):
    print(arr[i], end =" ");
```

Program 8:

Given a matrix, sort the rows of matrix in ascending order followed by sorting the columns in descending order.

Examples :

Input : a[3][3] = {{1, 2, 3},
{4, 5, 6},
{7, 8, 9}};

Output : 7 8 9
4 5 6
1 2 3

Input : a[3][3] = {{3, 2, 1},
{9, 8, 7},
{6, 5, 4}};

Output : 7 8 9
4 5 6
1 2 3

Approach:

- Traverse all rows one by one and sort rows in ascending order using a simple array sort.
- Convert matrix to its transpose
- Again sort all rows, but this time in descending order.

- Again convert a matrix to its transpose

```
# Python implementation to sort the rows
# of matrix in ascending order followed by
# sorting the columns in descending order

MAX_SIZE=10

# function to sort each row of the matrix
# according to the order specified by
# ascending.
def sortByRow(mat, n, ascending):

    for i in range(n):
        if (ascending):
            mat[i].sort()
        else:
            mat[i].sort(reverse=True)

# function to find
# transpose of the matrix
def transpose(mat, n):

    for i in range(n):
        for j in range(i + 1, n):

            # swapping element at index (i, j)
            # by element at index (j, i)
            temp = mat[i][j]
            mat[i][j] = mat[j][i]
            mat[j][i] = temp

# function to sort
# the matrix row-wise
# and column-wise
def sortMatRowAndColWise(mat, n):
```

```
    # sort rows of mat[][]
    sortByRow(mat, n, True)

    # get transpose of mat[][]
    transpose(mat, n)

    # again sort rows of
    # mat[][] in descending
    # order.
    sortByRow(mat, n, False)

    # again get transpose of mat[][]
    transpose(mat, n)

# function to print the matrix
def printMat(mat, n):

    for i in range(n):
        for j in range(n):
            print(mat[i][j] , " ", end="")
        print()

#Driver code
n = 3

mat = [[3, 2, 1],
    [9, 8, 7],
    [6, 5, 4]]

print("Original Matrix:")
printMat(mat, n)

sortMatRowAndColWise(mat, n)
```

```
print("Matrix After Sorting:")
printMat(mat, n)

# This code is contributed
# by Anant Agarwal.
```

Program 9:

Problem Statement :-In a cricket match , the captain of team A is tossing the coin. If head comes, team A wins the toss and similarly if tail comes then team B will win the toss. They will do N tosses of a coin for N different matches. Your task is to complete a function “count_wins()” that takes two inputs N and R. The function should return the probability of getting exactly R head on N successive tosses of a fair coin i.e. you have to find the probability that team A won the toss. coin has a equal probability of landing a head or tail (i.e. 0.5) on each toss.

Example

- **Case 1:**
  For the input provided as follows :
  4 3

  The output of the program will be:
  0.250000

- **Case 2:**
  For the input provided as follows :
  1 1

  The output of the program will be :
  0.500000

```
import java.util.*;
public class CountWins
{
  public static int fact (int n)
  {
```

```
  if (n == 0)
    return 1;
  return n * fact (n - 1);
 }
 public static double count_wins (int n, int r)
 {
  double res;
  res = fact (n) / (fact (r) * fact (n-r));
  res = res / (Math.pow (2, n));
  return res;
 }

 public static void main (String[]args)
 {
  Scanner sc = new Scanner (System.in);
  int n = sc.nextInt ();
  int r = sc.nextInt ();
  System.out.println (count_wins (n, r));
 }

}
```

Problem 10:

Problem Statement :- Write a program that will print the product of elements of left diagonal or right diagonal whose product is Maximum.The program takes total n rows and n columns as input (n numbers will be inputed per line and each number will be separated by space).

- **For example**
  For the input provided as follows:
  3
  1 2 3
  4 5 6
  7 8 9

  The program will provide the following output:
  105

Explanation :

The product of left diagonal is 45 and the right diagonal is 105. So the output is the greatest of these two which is 105.

- **Another example,**

  5
  1 2 3 4 5
  5 1 2 3 4
  4 5 1 2 3
  3 4 5 1 2
  2 3 4 5 1

  The program will provide the following output :
  120

  Explanation :

The product of left diagonal is 1 and the product of right diagonal is 120. so the greatest of these two is 120.

```
n=int(input())
L=[]
p1=1
p2=1
for i in range(n):
   l=list(map(int,input().split()))
   L.append(l)
   p1*=L[i][i]
   p2*=L[i][n-i-1]
print(max(p1,p2))
```

Problem 11:

The Mathematics teacher gave his students a task to check if the arithmetic equation has balanced parentheses in it or not. An equation is said to be balanced if all the left parenthesis ‘(’ has a matching right parenthesis ‘)’ and the count of both should be equal.If so , the program will print 0 else the program will print 1.

- **For example ,**
  For the input provided as follows:
  ((a+b)*c-d/e)

  The output of the program will be :
  0

  Explanation :

  All the parentheses of this equation are balanced . Hence the output is 0.

- **Another example ,**
  For the input provided as follows :
  (9*(7-2)*(1*5)

  The output of the program will be :
  1

  Explanation :

  All the parentheses of this equation are not balanced . Hence the output is 1.

```
s=input()
c=0
for i in s:
   if i =='(':
      c+=1
   elif (i==')') and c>0 :
      c-=1
print(int(c>0))
```

Problem 12:

Problem Statement :- Write a program to display the sum of all non-prime numbers from the given list of integers. A prime number is a number which is divisible by 1 or itself.

First line contains a number indicating the total number of integers and the second line contains integers separated by spaces.

- **For example ,**
  For the input provided as follows:

  5
  4 6 9 3 7

  The output of the program will be:
  19

  Explanation :

Non-prime number in the list are 4,6,9. So by adding these numbers we get 4+6+9=19.

- **Another Example ,**

  For the input provided as follows
  10
  4 30 7 23 5 99 3 45 19 17

  The output of the program will be :
  178

  Explanation :

Non-prime number in the list are 4,30,99,45 and the sum of all these number is 178.

```
from collections import defaultdict
import math
d=defaultdict(int)
def ifPrime(a):
   if d[a]==2 or d[a]==1:
      return d[a]-1
   if ((a&1)==0):
      d[a]=1
      return 0
   for i in range(3,int(math.sqrt(a))+1):
```

```
        if a%i==0:
            d[a]=1
            return 0
    d[a]=2
    return 1
d[2]=2
n=int(input())
a=list(map(int,input().split()))
sum=0
for i in a:
    if ifPrime(i)==0:
        sum+=i
print(sum)
```

Problem 13:

Problem Statement :- Ankit got his 12th board examination result , but in result only subject names and marks are mentioned, there are 5 subjects in total with total 100 marks each. You have to calculate his percentage in fraction like if the percentage are 72.5% then you have to write it as 72 ½ .Ankit is asking for your help.Write a program to calculate his percentage in fraction.

First five lines of input contain marks of five subjects and you have to calculate the percentage in fraction.

- **For example ,**
  For the input provided as follows:
  70 77 73 72 80

  The output of the program will be:
  74 2/5

  Explanation:
  The percentage of above five subject is (70+77+73+72+80)/5 = 74.4 which in fraction is represented as 74 2/5

- **Another Example ,**

For the input provided as follows

33 45 67 89 38

The output of the program will be :

54 2/5

Explanation :

The percentage of above five subject marks is (33+45+67+89+38)/5=54.4 which in fraction is represented as 54 2/5

```
sum=0
L=list(map(int,input().split()))
for v in L:
   sum+=v
print(sum//5,sum%5,"\b/",end="")
print(5)
```

Problem 15:

Problem Statement :- Write a program to find the sum of all the numbers whose cube roots are available in the list.If the number is not available then display “Not Found” as an output.

- **For example ,**

  1,8,7,125,5,211,343,25,400,512

  The output of the program will be:

  981

  Explanation :

  The numbers whose cube roots is present in the list are 1+125+343+512=981

- **Another example ,**

  4,9,343,25,78,512

The output of the program will be:

Not Found

Explanation:

No cube root is present in the list . Hence the answer is “Not Found”.

```
import math
L=list(map(int,input().split(",")))
sum=0
for i in L:
   g=math.ceil(i**(1./3.))
   if int(g**3)==int(i) and g in L:
      sum+=i
if sum!=0:
   print(sum)
else:
   print("Not Found")
```

Problem 16:

Problem Statement :- There are n number of words in the string separated by space.Your task is to arrange all these words according to their ASCII value, if let’s say there are three words in string i.e “I am good” then then the output of the program according to their ASCII value is “am good I”.It is given that all the letters of the string is in lowercase.

First line of input contains n which is the number of words in the string followed by a string.

- **For example ,**

  5

  abc hki gho hkc xyz

  The output of the program will be

  abc gho hkc hki xyz

- **Another example**

  i am coding in java

The output of the program will be

am coding i in java

```
l=list(map(str,input().split()))
l=sorted(l)
for i in l:
   print(i,end=" ")
```

Problem 17:

Problem Statement :- Write a program to read a sentence and output the number occurrences of each word in a sentence.The output of the program should be sorted into ascending order. It is given that all the letters of the words are in lowercase only and separated by spaces (“ ”). Display all the words in ascending order with their count.

Example

- **Case 1:**
  For the input given below
  you are good when your thoughts are good and your deeds are good

  The output of the program will be
  and – 1
  are – 3
  deeds – 1
  good – 3
  thoughts – 1
  when – 1
  you – 1
  your – 2

- **Case 2:**
  For the input given below
  if you fail to plan you are planning to fail.

  The output of the program will be
  are – 1

fail – 2
if – 1
plan -1
planning – 1
to – 2
you – 2

```
from collections import defaultdict
d=defaultdict(set)
l=list(map(str,input().split()))
l=sorted(l)
a={}
for i in l:
   if i in a:
      a[i]+=1
   else :
      a[i]=1
for i in a:
   print(i,"-",a[i])
```

Problem 18:

Problem Statement :- Write a program to count the number of sets bits in the binary representation of an integer. First line of input contains an integer n .The output of the program should be the number of set bits of a binary representation of a number.

Examples:

- **Case 1:**
  For the input provided as follows
  7

  The output of the program will be
  3

  Explanation
  Binary representation of 7 is 111 and has 3 set bits.

- **Case 2:**
  For the input provided as follows
  17

  The output of the program will be
  2

  Explanation
  Binary representation of 17 is 10001 and has 2 set bits.

```
n=int(input())
s=0
while n:
   n&=(n-1)
   s+=1
print(s)
```

Problem 19:
Given below the heights of students according to their standing order in a queue, find the maximum difference between heights of two students such that the smaller student stands before the larger student.The first line of input contain and integer n which is the number of elements in the array followed by n space separated elements of array.The output of the program should be the maximum difference of two array elements that satisfies the above condition.

- **Example ,**
  For the input given below
  7
  3 8 10 6 2 4 6

  The output of the program will be
  7

  Explanation:
  The maximum difference between 3 and 10 is 7.

- **Another Example,**
  9
  3 5 2 9 5 8 6 12 7

  The output of the program will be
  10

  Explanation:
  The maximum difference between 2 and 12 is 10.

```
n=int(input())
l=list(map(int,input().split()))
mx=0
mn=l[0]
for i in range(n):
   mn=min(mn,l[i])
   mx=max(mx,l[i]-mn)
print(mx)
```

Problem 20:
Problem Statement :-Suresh knows binary numbers very well. Unfortunately ,he forgot his bank account password , but he remembered a number and if we reverse the binary digits of that number we will get his password.Help suresh to find his password. You will be given a variable name n which is a no of passwords to find followed by n space separated numbers.You are required to give as output and passwords showing for each number.

For Example , if n is 3 and numbers 35000,3467,94856 , you are expected to give as output 487653376, 3517972480 and 290357248.

Example

- **Case 1:**

  For the input provided as follows :
  3

35000
3467
94856

Output of the program will be :
487653376, 3517972480 , 290357248

Explanation :

when we reverse the binary digits of number 35000,3467,94856. we get 487653376,3517972480 , 290357248 as the result.

- **Case 2:**

For the input provided as follows :
2
32678
10

Output of the program will be :
65536,1342177280

Explanation :

when we reverse the binary digits of 32678 and 10. we get 65536 and 1342177280 as the result.

```
m=int(input())
while m:
   m-=1
   n=int(input())
   s=""
   for i in range(32):
      if (n&1):
         s+='1'
      else:
         s+='0'
      n>>=1
```

```
   print(int(s,2),end=", ")
print("\b\b  ")
```

Problem 21:

Problem Statement -: Raman was playing a game, in starting he has x coins at some point of the game he has to pay some coins to get into the next level of the game, during each game he can collect some coins. If at anypoint of the game numbers of coins of Raman is less than one he will lose the game. Find the minimum value of x such that Raman wins.

```
n=int(input())
arr=[]
for i in range(n):
   arr.append(int(input()))
s,a=0,0
for i in arr:
   s=s+i
   if(s<1):
      a=a+(-1*s)+1
      s=1
print(a)
```

Program 22:

Problem Statement – Given a sting , return the character that appears the minimum number of times in the string. The string will contain only ascii characters, from the ranges (“a”-”z”,”A”-”Z”,0-9), and case matters . If there is a tie in the minimum number of times a character appears in the string return the character that appears first in the string.

Input Format:

Single line with no space denoting the input string.

OutputFormat:

Single character denoting the least frequent character.

Constraints:
Length of string <=10^6

Sample Input:
cdadcda

Sample Output:
c

Explanation:
C and A both are with minimum frequency. So c is the answer because it comes first with less index.

```
s=input()
ans=[]
for i in s:
   ans.append(s.count(i))
print(s[ans.index(min(ans))])
```

Program 23:
There are some groups of devils and they splitted into people to kill them. Devils make People to them left as their group and at last the group with maximum length will be killed. Two types of devils are there namely “@” and “$”
People is represented as a string “P”

Input Format:
First line with the string for input

Output Format:
Number of groups that can be formed.

Constraints:
2<=Length of string<=10^9

Input string
PPPPPP@PPP@PP$PP

Output
7

Explanation
4 groups can be formed

- PPPPPP@
- PPP@
- PP$
- PP

Most people in the group lie in group 1 with 7 members.

```
s=input()
s=s.replace("@"," ").replace("$"," ")
s=s.split()
ans=[]
for i in s:
   ans.append(len(i)+1)
ans[-1]-=1
print(max(ans))
```

Program 24:
Problem Statement – Rahul copies in the exam from his adjacent students. But he doesn't want to be caught, so he changes words keeping the letter constant. That means he interchanges the positions of letters in words. You are the examiner and you have to find if he has copied a certain word from the one adjacent student who is giving the same exam, and give Rahul the markings he deserves.

Note that: Uppercase and lowercase are the same.

Input Format:

- First line with the adjacent student's word
- Second line with Rahul's word

Output Format:

- 0 if not copied
- 1 if copied

Constraints:

- 1<=Length of string<=10^6

Sample Input:

CAR

Acr

Sample Output:

1

```
s=input()
s1=input()
s=s.lower()
s=sorted(s)
s1=s1.lower()
s1=sorted(s1)
if s1==s:
   print(1)
else:
   print(0)
```

Problem 25:

Problem Statement – Mr. Robot is making a website, in which there is a tab to create a password. As other websites, there are rules so that the password gets complex and none can predict the password for another. So he gave some rules like:

- At least one numeric digit
- At Least one Small/Lowercase Letter
- At Least one Capital/Uppercase Letter
- Must not have space
- Must not have slash (/)

- At least 6 characters

If someone inputs an invalid password, the code prints: “Invalid password, try again”.

Otherwise, it prints: “password valid”.

Input Format:

A line with a given string as a password

Output Format:

- If someone inputs an invalid password, the code prints: “Invalid password, try again”.
- Otherwise, it prints: “password valid”, without the quotation marks.

Constraints:

- Number of character in the given string <=10^9

Sample input 1:

abjnlL09

Sample output 1:

password valid

Sample input 2:

jjnaskpk

Sample output 2:

Invalid password, try again

```
def CheckPassword(s,n):
   if n<4:
      return 0
   cap=0
```

```
    nu=0
    lo=0
    for i in range(n):
        if s[i]==' ' or s[i]=='/':
            return 0
        if s[i]>='A' and s[i]<='Z':
            cap+=1
        if s[i]>='a' and s[i]<='z':
            lo+=1
        elif s[i].isdigit():
            nu+=1

    if cap>0 and nu>0 and lo>0:
        return 1
    else:
        return 0

s=input()
a=len(s)
if CheckPassword(s,a):
    print("password valid")
else:
    print("Invalid password, try again")
```

Problem 26:

Jack and Jill are playing a string game. Jack has given Jill two strings A and B. Jill has to derive a string C from A, by deleting elements from string A, such that string C does not contain any element of string B. Jill needs help to do this task. She wants a program to do this as she is lazy. Given strings A and B as input, give string C as Output.

Example 1:

- **Input:**
  tiger -> input string A
  ti -> input string B
- **Output:**
  ger -> Output string C
- **Explanation:**
  After removing "t" and "i" from "tiger", we are left with "ger".
  So, the answer is "ger".

Example 2:

- **Input:**
  processed -> input string A
  esd -> input string B
- **Output:**
  proc -> Output string C
- **Explanation:**
  After removing "e" "s" and "d" from "processed", we are left with "proc".
  So, the answer is "proc".

Example 3:

- **Input:**
  talent -> input string A
  tens -> input string B
- **Output:**
  al -> Output string C

Explanation:

After removing "t" "e" and "n" from "talent", we are left with "al".

So, the answer is "al".

```
a = input()
b = input()
c = ""
for i in a:
    if i in b:
        continue
    else:
        c += i
print(c)
```

Problem 27:

A chocolate factory is packing chocolates into packets. The chocolate packets here represent an array arrt of N number of integer values. The task is to find the empty packets(0) of chocolate and push it to the end of the conveyor belt(array).

For Example:

- N=7 and arr = [4,5,0,1.9,0,5,0].
- There are 3 empty packets in the given set. These 3 empty packets represented as O should be pushed towards the end of the array

Example 1:

Input:

- 7 – Value of N
- [4,5,0,1,0,0,5] – Element of arr[O] to arr[N-1],While input each element is separated by newline

Output:

- 4 5 1 9 5 0 0

Example 2:

Input:

- 6

- — Value of N.
- [6,0,1,8,0,2] – Element of arr[0] to arr[N-1], While input each element is separated by newline

Output:

- 6 1 8 2 0 0

```
n=int(input())
j=0
L=[0 for i in range(n)]
for i in range(n):
   a=int(input())
   if a!=0:
      L[j]=a
      j+=1
for i in L:
   print(i,end=" ")
```

Problem 28:

Joseph is learning digital logic subject which will be for his next semester. He usually tries to solve unit assignment problems before the lecture. Today he got one tricky question. The problem statement is “A positive integer has been given as an input. Convert decimal value to binary representation. Toggle all bits of it after the most significant bit including the most significant bit. Print the positive integer value after toggling all bits”.

Constraints :

1<=N<=100

Example 1:

Input :

10 -> Integer

Output :

5 -> result- Integer

Explanation:

Binary representation of 10 is 1010. After toggling the bits(1010), will get 0101 which represents “5”. Hence output will print “5”.

```
import math
n=int(input())
k=(1<<int(math.log2(n))+1)-1
print(n^k)
```

Problem 29:

Given a maximum of four digit to the base 17 (10 – A, 11 – B, 12 – C, 13 – D … 16 – G} as input, output its decimal value.

Test Cases

Case 1

- Input – 1A
- Expected Output – 27

Case 2

- Input – 23GF
- Expected Output – 10980

```
'''The int() function converts the specified value into an integer number.
We are using the same int() method to convert the given input.
int() accepts two arguments, number and base.
Base is optional and the default value is 10.
```

In the following program we are converting to base 17'''

```
num = str(input())
print(int(num,17))
```

Problem 30:

Given a maximum of 100 digit numbers as input, find the difference between the sum of odd and even position digits

Test Cases

Case 1

- Input: 4567
- Expected Output: 2

Explanation : Odd positions are 4 and 6 as they are pos: 1 and pos: 3, both have sum 10. Similarly, 5 and 7 are at even positions pos: 2 and pos: 4 with sum 12. Thus, difference is 12 – 10 = 2

Case 2

- Input: 5476
- Expected Output: 2

Case 3

- Input: 9834698765123
- Expected Output: 1

```
num = [int(d) for d in str(input("Enter the number:"))]
even,odd = 0,0
for i in range(0,len(num)):
    if i % 2 ==0:
        even = even + num[i]
    else:
        odd = odd + num[i]

print(abs(odd-even))
```

Problem Statement – Bhojon is a restaurant company and has started a new wing in a city. They have every type of cook except the meatball artist. They had fired their last cook because the sale of meatballs in their restaurant is really great, and they can't afford to make meatballs again and again every time their stock gets empty. They have arranged a hiring program, where you can apply with their meatball. They will add the meatball in their seekh (a queue) and everytime they cut the meatball they take it and cut it on the day's quantity and then re-add the meatball in the seekh. You are the hiring manager there and you are going to say who is gonna be hired.

Day's quantity means, on that very day the company sells only that kg of meatballs to every packet.

If someone has less than a day's quantity, it will be counted as a sell.

Function Description:

- Complete the function with the following parameters:

Parameters:

Name	Type	Description
N	Integer	How many people are participating in the hiring process.
D	Integer	Day's quantity, how many grams of meatball is being sold to every packet.
Array[ ]	Integer array	Array of integers, the weight of meatballs everyone came with.

Return:

- The ith person whose meat is served at last.

Constraints:

- 1 <= N <= 10^3
- 1 <= D <= 10^3
- 1 <= Array[i] <= 10^3

Input Format:

- First line contains N.
- Second line contains D.
- After that N lines contain The ith person's meatball weight.

Output Format: The 1 based index of the man whose meatball is served at the last.

Sample Input 1:

4

2

[7 8 9 3]

Sample Output 1:
3

Explanation:
The seekh or meatball queue has [7 8 9 3] this distribution. At the first serving they will cut 2 kgs of meatball from the first meatball and add it to the last of the seekh, so after 1st time it is:
[8 9 3 5]
Then, it is: [9 3 5 6], [3 5 6 7], [5 6 7 1], [6 7 1 3], [7 1 3 4], [1 3 4 5], [3 4 5], [4 5 1], [5 1 2], [1 2 3], [2 3], [3], [1], [0]
So the last served meatball belongs to the 3rd person.

```
n = int(input())
m = 0
mxPos = 0
v = []
x = int(input())
v_input = input().split()
for i in range(n):
   v.append(int(v_input[i]))
   v[i] = (v[i]-1)//x
   if v[i] >= m:
      m = v[i]
      mxPos = i
print(mxPos+1)
```

Problem Statement 31

– Abhijeet is one of those students who tries to get his own money by part time jobs in various places to fill up the expenses for buying books. He is not placed in one place, so what he does, he tries to allocate how much the book he needs will cost, and then work to earn that much money only. He works and then buys the book respectively. Sometimes he gets more money than he needs so the money is saved for the next book. Sometimes he doesn't. In that time, if he has stored money from previous books, he can afford it, otherwise he needs money from his parents.
Now His parents go to work and he can't contact them amid a day. You are his friend, and you have to find how much money minimum he can borrow from his parents so that he can buy all the books.
He can Buy the book in any order.

Function Description:
Complete the function with the following parameters:

Name	Type	Description
N	Integer	How many Books he has to buy that day.
EarnArray[ ]	Integer array	Array of his earnings for the ith book
CostArray[ ]	Integer array	Array of the actual cost of the ith book.

Constraints:

- 1 <= N <= 10^3
- 1 <= EarnArray[i] <= 10^3
- 1 <= CostArray[i] <= 10^3

Input Format:

- First line contains N.
- Second N lines contain The ith earning for the ith book.
- After that N lines contain The cost of the ith book.

Output Format: The minimum money he needs to cover the total expense.

Sample Input 1:

3
[3 4 2]
[5 3 4]

Sample Output 1:

3

Explanation:

At first he buys the 2nd book, which costs 3 rupees, so he saves 1 rupee. Then he buys the 1st book, that takes 2 rupees more. So he spends his stored 1 rupee and hence he needs 1 rupee more. Then he buys the last book.

```
n=int(input())
L1=[0] *n
L2=[0]*n


for i in range(n):
   L2[i]=int(input())
for i in range(n):
```

```
    L1[i]=int(input())


for i in range(n):
    L2[i]=L2[i]-L1[i];

L2.sort()
su=0
ans=0
print(L2)
for i in range(n):
    su=su+L2[i]

    if su<0:
        ans = ans + abs(su)
        su=0

print(ans)
```

Problem Statement 32:

– Nobel Prize-winning Austrian-Irish physicist Erwin Schrödinger developed a machine and brought as many Christopher Columbus from as many parallel universes he could. Actually he was quite amused by the fact that Columbus tried to find India and got America. He planned to dig it further.

Though totally for research purposes, he made a grid of size n X m, and planted some people of America in a position (x,y) [in 1 based indexing of the grid], and then planted you with some of your friends in the (n,m) position of the grid. Now he gathered all the Columbus in 1,1 positions and started a race.

Given the values for n, m, x, y, you have to tell how many different Columbus(s) together will explore you as India for the first time.

Remember, the Columbus who will reach to the people of America, will be thinking that as India and hence wont come further.

Function Description:

Complete the markgame function in the editor below. It has the following parameter(s):

Parameters:

Name	Type	Description
n	Integer	The number of rows in the grid.
m	Integer	The number of columns in the grid.
x	Integer	The American cell's Row.

y	Integer	The American cell's Column.

Constraints:

- 1 <= n <= 10^2
- 1 <= m <= 10^2
- 1 <= x <= n
- 1 <= y <= m

Input Format:

- The first line contains an integer, n, denoting the number of rows in the grid.
- The next line contains an integer m, denoting the number of columns in the grid.
- The next line contains an integer, x, denoting the American cell's row.
- The next line contains an integer, y, denoting the American cell's column.

Sample Cases

Sample Input 1

2
2
2
1

Sample Output 1

1

Explanation

The only way possible is (1,1) ->(2,1) -> (2,2), so the answer is 1.

```
import math
n=int(input())-1
m=int(input())-1
x=int(input())-1
y=int(input())-1
ans=math.factorial(n+m)
ans=ans//(math.factorial(n))
ans=ans//(math.factorial(m))
ans1=math.factorial(x+y)
ans1=ans1//(math.factorial(x))
ans1=ans1//(math.factorial(y))
x1=n-x
y1=m-y
ans2=math.factorial(x1+y1)
ans2=ans2//(math.factorial(x1))
ans2=ans2//(math.factorial(y1))
print(ans-(ans1*ans2))
```

Problem Statement 33:

– Aashay loves to go to WONDERLA , an amusement park. They are offering students who can code well with some discount. Our task is to reduce the cost of the ticket as low as possible.

The cost of tickets can be removed by removing the digits from the price given. They will give some k turns to remove the digits from the price of the ticket. Your task is to help Aashay in coding a program that can help him to reduce the cost of a ticket by removing the digits from its price and getting the maximum possible discount.

Note – You cannot make the cost of a ticket zero. For eg -: If the cost of a ticket is 100, and you have 2 turns to reduce the price, the final price will be 1 and not zero.

Constraints:

- 1 <= number of tickets <= 10^5
- 1 <= K <= number of tickets

Input Format for Custom Testing:

- The first line contains a string,Tickets, denoting the given cost of each ticket.
- The next line contains an integer, K, denoting the number of tickets that is to be removed.

Sample Cases:

- **Sample Input 1**
  203
  3
- **Sample Output 1**
  0

```
import sys
n=input()
k=int(input())
n1=len(n)
if len(n)<=k:
 print(0)
 sys.exit()
a=''
i=0
while i < (n1-1) and k>0:
 if int(n[i])>int(n[i+1]):
  i+=1
  k-=1
  continue

 else:
```

```
  a+=n[i]
  i+=1

a+=n[i]
i+=1
if k>0:
 a=a[:-k]

if i<=(n1-1):
 while i < n1:
  a+=n[i]
  i+=1
print(int(a)%((10**9)+7))
```

Problem statement 35:

Shovon is an HR in a renowned company and he is assigning people to work. Now he is assigning people work in a fashion where if he assigns somework a work of cost 2, the next person will be strictly getting a job with cost equal or more than 2. Given that Shovon's company has infinite work and a number of employees, how many distributions can be possible. The cost of jobs can go 0 to 9.

Function Description:

Complete the special_numbers function in the editor below. It has the following parameter(s):

Parameters:

Name	Type	Description
N	Integer	The number of depts.
arr[ ]	Integer array	The number of employees in each dept..

Return: The function must return an INTEGER denoting the sum of answers for all distinct distributions.

Constraints:

- 1 <= n <= 100
- 1 <= arr[i] <= 200

Sample Cases:

- **Sample Input 1**

  2
  4
  1

- **Sample Output 1**
  725
- **Description**
  The ans if m = 1 is 10, which is all numbers from 0 to 9
  The ans for m = 2 is 55
  The answer for m = 3 is 220
  The answer for m = 4 is 715
  So fun(4) + fun(1) = 725

```
n=int(input())
a1=[]
for i in range(n):
 a1.append(int(input()))
dp=[0]*201
dp[1]=10
dp[2]=55
a=[1]*10
i=3
while i<201:
 s=0
 for i1 in range(10):
   s+=a[i1]
   a[i1]=s
   dp[i]+=(s*(10-i1))
   dp[i]=dp[i]%((10**9)+7)
 i+=1
s1=0
for i in a1:
 s1+=dp[i]
 s1=s1%((10**9)+7)
print(s1)
```

Problem statement 36:

In an airport , the Airport authority decides to charge some minimum amount to the passengers who are carrying luggage with them. They set a threshold weight value, say T, if the luggage exceeds the weight threshold you should pay double the base amount. If it is less than or equal to threshold then you have to pay $1.

Function Description:

Complete the weightMachine function in the editor below. It has the following parameter(s):

Parameters:

Name	Type	Description
N	Integer	number of luggage
T	Integer	weight of each luggage
weights[ ]	Integer array	threshold weight

Returns: The function must return an INTEGER denoting the required amount to be paid.

Constraints:

1 <= N <= 10^5
1 <= weights[i] <= 10^5
1 <= T <= 10^5
Input Format for Custom Testing:

The first line contains an integer, N, denoting the number of luggages.
Each line i of the N subsequent lines (where 0 <= i <n) contains an integer describing weight of ith luggage.
The next line contains an integer, T, denoting the threshold weight of the boundary wall.
Sample Cases:

Sample Input 1
4
1
2
3
4
3
Sample Output 1
5
Explanation:
Here all weights are less than threshold weight except the luggage with weight 4 (at index 3) so all pays base fare and it pays double fare.

```
def weightMachine(N,weights,T):
   amount=0
```

```
    for i in weights:
        amount+=1
        if(i>T):
            amount+=1
    return amount
N=int(input())
weights=[]
for i in range(N):
    weights.append(int(input()))
T=int(input())
print(weightMachine(N,weights,T))
```