PLACEMENT LELO

TCS CodeVita Season 13 Complete Guide:

This PDF includes:

- CodeVita Round Details
- How to Prepare for CodeVita
- Important Topics
- CodeVita Syllabus
- **Practice Questions with Solutions**
- **Previous Year Questions with Solutions**
- Handling Status Messages in CodeVita

Pdf by - @PLACEMENTLELO

CodeVita Rounds Details:

TCS CodeVita Season 13 is conducted in three rounds.

Round 1 : Pre-Qualifier

- This round is divided into 2 sub-rounds i.e.,
  - Round 1 Zone 1
  - Round 1 Zone 2
- This is the first round of the competition. In this round the participants have to solve 6 questions within a time-limit of 6 hours. The difficulty level of this round is lowest with comparison to other round of the competition.
- It is not compulsory for the participants to solve all the questions. There may be a case that you may proceed to the next round, even if you have solved half of the questions.

Round 2 : Qualifier

- This is the second round of the competition. In this round the participants have to solve 6 questions but the difficulty level of this round is pretty high than the previous one. If you clear this round, than you can surely get a job in TCS if you want so, and that too at a pretty good package.

Round 3 : Final Round

- This is a new Round introduced by TCS Codevita this year, it is an additional round to get the top 3 finalist of the tournament. These 3 students will be awarded in the next round with is Grand Finale Ceremony.

Round 4 : Grand Finale Ceremony

- This is the final round of the competition. And being so, it is the most difficult round too. The previous two rounds are online rounds, but this round takes place at a **particular venue decided by TCS.**
- In this round the participants have to solve 10 questions.
- Clearing this round will give you a pleasing cash price, and a delightful job offer in TCS.
- It is not that easy to reach upto this round. According to previous year stats, there were more than 200k participants, but only 25 reached up to this round.
- The finalists stand a chance to travel** to India for the Season 13 LIVE Grand Finale experience.

Important Exam Dates:

Round	Description	Date	Time
M_1	MockVita 1 [All Regions]	29th - 30th August 2025	03:00 PM to 03:00 PM IST(24 hrs)
M_2	MockVita 2 [All Regions]	12th - 13th September 2025	03:00 PM to 03:00 PM IST(24 hrs)
R_1Z_1	Round 1 Zone 1	10th - 11th October 2025	03:00 PM to 03:00 PM IST(24 hrs))
R_1Z_2	Round 1 Zone 2	31st Oct - 1st Nov 2025	03:00 PM to 03:00 PM IST(24 hrs)
R_2	Round 2 (Qualifier)	14th - 15th November 2025	03:00 PM to 03:00 PM IST(24 hrs)
F	Final Round	To be Declared	To be Declared
GFC	Grand Finale Ceremony	To be Declared	To be Declared

TCS CodeVita Apply Link – Click Here

Latest Jobs and Internships are regularly uploaded on our WhatsApp Channel:
https://whatsapp.com/channel/0029Va4bojk90x2rq1LgdD1a

TCS CodeVita Telegram and Whatsapp Groups:

TCS CodeVita Exam Answers will be uploaded in this channel for Free:
https://telegram.me/+_hn3cBQVbGliYTI9

TCS CodeVita Discussion Group:
https://telegram.me/+oZ4x3k1RtXdkNWU1

TCS CodeVita Exam Answers Whatsapp Channel:
https://whatsapp.com/channel/0029Vb67dZr4o7qMrkr3x70h

Must Join All the Groups !

How to Prepare for CodeVita:

What is TCS CodeVita?

TCS Code Vita is a Global coding contest being organized every year by TATA Consultancy Services Limited (TCS). The contest's main objective is to identify passionate coders and sharpen their programming skills through some real-life computing practices.

What excites us to crack CodeVita?

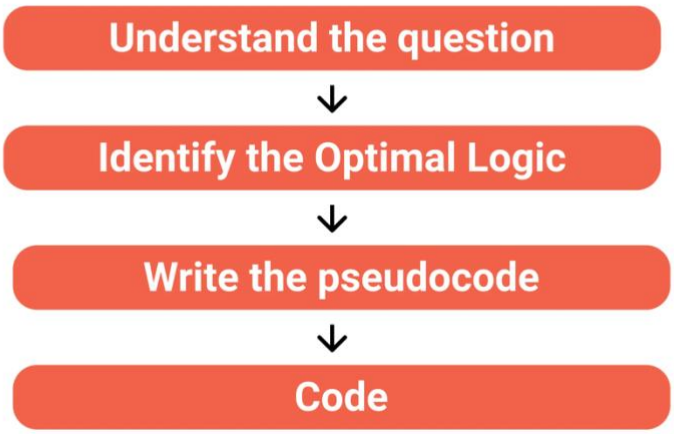
- Global Ranking to Top Coders
- Top 3 Coders to win total prize money of US$20,000/-
- A chance to explore an exciting career with one of the world's most powerful brand
- A chance to compete with some of The Best Coders in the world
- A Platform to showcase programming skills
- Students who progressed to the Finals get an opportunity to travel to India for the Grand Finale.

How to prepare for TCS CodeVita?

In order to prepare for TCS CodeVita, it is necessary to develop immense programming skills.

Here are some of the tips on how to prepare for the online programming round of TCS CodeVita Season 13!

4 Step Approach for CodeVita Questions



Understanding the question:

To understand the problem statement, do follow the below guidelines.

- Try to read and understand the problem statement.
- Try to analyse the problem statement whether it requires Logical, Mathematical, or some Algorithmic Techniques to be handled.
- Frame the format of Input and the Output.
- Check for the Constraints to pick out the apt logic or algorithms wisely.
- Be strong in Algorithmic Thinking.

Identify the Optimal Logic:

- Here get an understanding of the sample input and sample output.
- Try to find the optimal logic behind the problem.

Write the pseudocode:

- Once after getting out the logic, try to frame the pseudocode of the program.

Code:

- After framing out the pseudocode, now try to implement it as a code in the language with which you are comfortable.

Language allowed and good offline compilers

Here you could either use the online compiler given in the TCS CodeVita test or you can go with the offline compiler of your choice. The main thing is that your code should pass both the public and the private test cases. If you go with offline compilers, do ensure that you meet with the appropriate versions as mentioned below.

Language	Compiler / Interpreter Versions
C	gcc 9.3.0
C++	g++ 9.3.0
C#	Mono 6.12.0.122
Java	Open jdk 11.0.11
Perl	5.30.0
PHP	7.4.3
Python	Python 3.8.10
Ruby	3.0.0

Tips to Pass the Test Cases

- Do write the program that solves both the public and the private test cases.
- The program should be able to handle as much as data it can manage.
- Ensure that it follows time and space complexity.

Notes

- Once you start the test, you will not be able to pause the test
- The coding test exactly consists of 6 coding questions to be solved in 6 hours.
- Do check the internet connection before getting into the test.
- Read the questions carefully and start solving the questions you are familiar with.
- Avoid plagiarism.

TCS Codevita Syllabus:

TCS CodeVita Syllabus by PLACEMENTLELO
Scenario-based questions
Mathematics Oriented
Recursion
Dynamic Programming
Arrays
Pattern programming

TCS CodeVita Syllabus by PLACEMENTLELO
Greedy Approach
String-based
Searching

The TCS CodeVita Pre-Qualifier Round consists of 6 coding questions to be answered in 6 hours. These questions range from easy to hard.

TCS CodeVita Important Topics

Here are some of the important topics from the above syllabus.

Recurrence
Optimal Interleaving
Permutation
All Direction Traversal
Optimal parenthesis
Recursion stack holding the result
Pointer holding result
All Direction Traversal with constraints

Searching
Exhaustive nested searching
Binary Search
BS Variants

Maths
Time
Circumference
Concatenate numbers
Arithmetic expression
Module/remainder pattern
Dijikstra's

Dynamic Programming
Subsequence
Table Building
Memorization

Strings
Subsequence
Interleaved access
Stat
Parsing

Reversal kind
Pattern Matching

Here are a few easy topics and their logic to get practiced.

Question	Logic
Constellation	Pattern
Minimum Gifts	Array, Greedy Approach
Prime Time Again	Array and Looping
Minimize the Sum	Looping, STL/Collections, Greedy Approach
Railway Station	Looping
Count Pairs	Array, Greedy Approach

TCS CodeVita Questions | Previous Year Questions & Solutions

Important topics

Here are some of the most asked concepts/questions in TCS CodeVita. Have a look at them and start solving them now.

Important Topics
Max Flow Problem

Segment Trees
Bit-masking Dynamic Programming
Pattern matching and Pattern searching : Z algorithm, Rabin-Karp algorithm, KMP algorithm
Depth First Search and Breadth-First Search
Dijkstra, Kruskal
Coin Sum
Subset Sum
Knapsack
Min Cost Path
DFS with memorization
DFS in matrices

TCS CodeVita Questions

Here are some of the TCS CodeVita questions from previous year's papers with detailed solutions.

TCS CodeVita Questions #Q1

Constellation

Three characters { #, *, . } represents a constellation of stars and galaxies in space. Each galaxy is demarcated by # characters. There can be one or many stars in a given galaxy. Stars can only be in the shape of vowels { A, E, I, O, U }. A collection of * in the shape of the

vowels is a star. A star is contained in a 3x3 block. Stars cannot be overlapping. The dot(.) character denotes empty space.

Given 3xN matrix comprising of { #, *, . } character, find the galaxy and stars within them.

Note: Please pay attention to how vowel A is denoted in a 3x3 block in the examples section below.

Constraints

3 <= N <= 10^5

Input

Input consists of a single integer N denoting the number of columns.

Output

The output contains vowels (stars) in order of their occurrence within the given galaxy. The galaxy itself is represented by the # character.

Example 1

Input

18

* . * # * * * # * * * # * * * . * .

* . * # * . * # . * . # * * * * * *

* * * # * * * # * * * # * * * * . *

Output

U#O#I#EA

Explanation

As it can be seen that the stars make the image of the alphabets U, O, I, E, and A respectively.

Example 2

Input

12

.#.***#.*.

.#..*.#***

#.#*.*

Output

U#I#A

Explanation

As it can be seen that the stars make the image of the alphabet U, I, and A.

Possible solution:

Input:

12

.#.***#.*.

* . * # . . * . # * * *

* * * # . * * * # * . *

C++

```
#include <iostream>
using namespace std;
int main()
{
	int n,x1,y1;
	cin>>n;
	char x[3][n];
	for(int i=0;i<3;i++)
	{
		for(int j=0;j<n;j++)
		{
		cin>>x[i][j];
	}
	}
	for(int i=0;i<n;i++)
	{
	if(x[0][i]=='#' && x[1][i]=='#' && x[2][i]=='#')
		{
		cout<<'#';
	}
		else if(x[0][i]=='.' && x[1][i]=='.' && x[2][i]=='.')
		{}
		else
		{
		char a,b,c,a1,b1,c1,a2,b2,c2;
		x1 = i;
		a = x[0][x1];
		b = x[0][x1+1];
		c = x[0][x1+2];
		a1 = x[1][x1];
		b1 = x[1][x1+1];
		c1 = x[1][x1+2];
		a2 = x[2][x1];
		b2 = x[2][x1+1];
		c2 = x[2][x1+2];
		if(a=='.' && b=='*' && c=='.' && a1=='*' && b1=='*' && c1=='*' && a2=='*'
&& b2=='.' && c2=='*')
			{
```

```
                    cout<<"A";
                    i = i + 2;
                }
                if(a=='*' && b=='*' && c=='*' && a1=='*' && b1=='*' && c1=='*' && a2=='*'
&& b2=='*' && c2=='*')
                    {
                    cout<<"E";
                    i = i + 2;
                }
                if(a=='*' && b=='*' && c=='*' && a1=='.' && b1=='*' && c1=='.' && a2=='*'
&& b2=='*' && c2=='*')
                    {
                    cout<<"I";
                    i = i + 2;
                }
                if(a=='*' && b=='*' && c=='*' && a1=='*' && b1=='.' && c1=='*' && a2=='*'
&& b2=='*' && c2=='*')
                    {
                    cout<<"O";
                    i = i + 2;
                }
                if(a=='*' && b=='.' && c=='*' && a1=='*' && b1=='.' && c1=='*' && a2=='*'
&& b2=='*' && c2=='*')
                    {
                    cout<<"U";
                    i = i + 2;
                }
        }
        }
}
```

OUTPUT
U#I#A

Java 8

```
import java.util.*;
public class Main {
   public static void main(String[] args) throws Exception {
      // Your code here!
      Scanner sc=new Scanner( System.in );
      int n=sc.nextInt();
      char gal[][] = new char [3][n];
      for(int i=0;i<3;i++){
         String a=sc.next();
```

```
        for(int j=0;j<n;j++)
        gal[i][j]=a.charAt(j);
      }
      for(int i=0;i<n;)
      {
        if(gal[0][i]=='#')//||gal[0][i+1]=='#')
        {
          System.out.print("#"); i++; continue;
        }
        if(gal[0][i]=='.' && gal[1][i]=='.' && gal[2][i]=='.')
        {
          i++; continue;
        }
        if(gal[0][i]=='.' && gal[0][i+2]=='.' && gal[2][i+1]=='.')
        System.out.print("A");
        else if(gal[1][i+1]=='.')
        {
          if(gal[0][i+1]=='.')
          System.out.print("U");
          else
          System.out.print("O");
        }
        else if(gal[1][i]=='.' && gal[1][i+2]=='.')
        System.out.print("I");
        //else if(gal[0][i]=='#')
        //System.out.print("#");
        else
        System.out.print("E");

        i+=3;
      }
      // System.out.println("XXXXXXXX");
    }
```

OUTPUT
U#I#A

Python

```
from collections import deque
def initialize():
    q = deque()
    # A
    s = ""
    q.append(['.', '*', '*'])
    q.append(['*', '*', '.'])
    q.append(['.', '*', '*'])
```

```
    s = ''.join(map(str, q))
    vowels[s] = 'A'
    q.clear()
    #E
    q.append(['*', '*', '*'])
    q.append(['*', '*', '*'])
    q.append(['*', '*', '*'])
    s = ''.join(map(str, q))
    vowels[s] = 'E'
    q.clear()

    #I
    q.append(['*', '.', '*'])
    q.append(['*', '*', '*'])
    q.append(['*', '.', '*'])
    s = ''.join(map(str, q))
    vowels[s] = 'I'
    q.clear()
    #O
    q.append(['*', '*', '*'])
    q.append(['*', '.', '*'])
    q.append(['*', '*', '*'])
    s = ''.join(map(str, q))
    vowels[s] = 'O'
    q.clear()
    #U
    q.append(['*', '*', '*'])
    q.append(['.', '.', '*'])
    q.append(['*', '*', '*'])
    s = ''.join(map(str, q))
    vowels[s] = 'U'
    q.clear()
     return vowels
vowels = {}
vowels = initialize()
n = int(input())
x = []
for i in range(n):
    x.append(['.', '.', '.'])
for i in range(3):
    l = []
    l = list(input())
    for j in range(n):
        x[j][i] = l[j]

constellation = ""
```

```
star = deque()
for i in range(n):
    if len(star) ==3:
        s = ''.join(map(str, star))
        if s in vowels:
            constellation += vowels[s]
            star.clear()
        else:
            star.popleft()
    if x[i]==['#', '#', '#']:
        star.clear()
        constellation += '#'
        continue
    star.append(x[i])
if len(star)==3:
    s = ''.join(map(str, star))
    if s in vowels:
        constellation += vowels[s]

print(constellation, end="")
```

OUTPUT
U#I#A

TCS CodeVita Questions #Q2

Prime Time Again

Here on earth, our 24-hour day is composed of two parts, each of 12 hours. Each hour in each part has a corresponding hour in the other part separated by 12 hours: the hour essentially measures the duration since the start of the daypart. For example, 1 hour in the first

part of the day is equivalent to 13, which is 1 hour into the second part of the day.

Now, consider the equivalent hours that are both prime numbers. We have 3 such instances for a 24-hour 2-part day:

5~17

7~19

11~23

Accept two natural numbers D, P >1 corresponding respectively to a number of hours per day and the number of parts in a day separated by a space. D should be divisible by P, meaning that the number of hours per part (D/P) should be a natural number. Calculate the number of instances of equivalent prime hours. Output zero if there is no such instance. Note that we require each equivalent hour in each part in a day to be a prime number.

Example:

Input: 24 2

Output: 3 (We have 3 instances of equivalent prime hours: 5~17, 7~19, and 11~23.)

Constraints

10 <= D < 500

2 <= P < 50

Input

The single line consists of two space-separated integers, D and P corresponding to the number of. hours per day and number of parts in a day respectively

Output

Output must be a single number, corresponding to the number of instances of equivalent prime number, as described above

Example 1

Input

36 3

Output

2

Explanation

In the given test case D = 36 and P = 3

Duration of each daypart = 12

2~14~X

3~15~X

5~17~29 - an instance of equivalent prime hours

7~19~31 - an instance of equivalent prime hours

11~23~X

Hence the answer is 2.

Possible solution:

Input:

49 7

C++

```
#include<bits/stdc++.h>
using namespace std;
bool isprime(int n)
{
	if(n==1)
		return false;
	for(int i=2;i<=(int)sqrt(n);i++)
	{
		if(n%i==0)
			return false;
	}
	return true;
}
int main()
{
	int D,P,i,j,p,t=1;
	cin>>D>>P;
	p=D/P;
	int time[p][P];
	for(i=0;i<P;i++)
	{
		for(j=0;j<p;j++)
		{
```

```
                time[j][i]=t++;
            }
        }
        t=0;
        for(i=0;i<p;i++)
        {
            bool flag=true;
            for(j=0;j<P;j++)
            {
                if(!isprime(time[i][j]))
                {
                    flag=false;
                    break;
                }
            }
            if(flag)
                t++;
        }
        cout<<t;
}
```

OUTPUT

0

Java 8

```
import java.util.*;
public class Main {
  public static boolean isprime(int n)
  {
    if(n==1) return false;
    for(int i=2;i<=(int) Math.sqrt(n) ; i++){
      if(n%i==0)
        return false;
    }
    return true;
  }
  public static void main(String[] args) throws Exception {
    // Your code here!
    Scanner sc= new Scanner(System.in);

    // System.out.println("XXXXXXXX");
    int i,j,D,P,p,t=1;
    D=sc.nextInt();
```

```
        P=sc.nextInt();
        p=D/P;
        int time[][] = new int [p][P];
        for(i=0;i<P;i++){
           for(j=0;j<p;j++){
              time[j][i]=t++;
           }
        }
        t=0;
        for(i=0;i<p;i++)
        {
           boolean flag=true;
           for(j=0;j<P;j++)
           {
              if(!isprime(time[i][j]))
              {
                 flag=false; break;
              }
           }
           if(flag) t++;
        }
        System.out.println(t);
     }
}
```

OUTPUT

0

Python

```
primes = set()
def generate(lim):
    for i in range(2, lim):
        x = 1
        for j in primes:
            if (i%j==0):
                x = 0
                break
                if x==1:
            primes.add(i)
d, p = map(int, input().split())
generate(d)
c = 0
inv = d//p
```

```
for i in range(inv):
   if i in primes:
      x = 1
      for j in range(1, p):
         if (j*inv+i) not in primes:
            x = 0
            break
      if x==1:
         c+=1
print(c)
```

OUTPUT
0

TCS CodeVita Questions #Q3

Minimum Gifts

A Company has decided to give some gifts to all of its employees. For that, the company has given some rank to each employee. Based on that rank, the company has made certain rules to distribute the gifts.

The rules for distributing the gifts are:

Each employee must receive at least one gift.

Employees having higher ranking get a greater number of gifts than their neighbours.

What is the minimum number of gifts required by the company?

Constraints

1 < T < 10

1 < N < 100000

1 < Rank < 10^9

Input

First line contains integer T, denoting the number of test cases.

For each test case:

First line contains integer N, denoting the number of employees.

Second line contains N space separated integers, denoting the rank of each employee.

Output

For each test case print the number of minimum gifts required on a new line.

Example 1

Input

2

5

1 2 1 5 2

2

1 2

Output

7

3

Explanation

For test case 1, adhering to the rules mentioned above,

Employee # 1 whose rank is 1 gets one gift

Employee # 2 whose rank is 2 gets two gifts

Employee # 3 whose rank is 1 gets one gift

Employee # 4 whose rank is 5 gets two gifts

Employee # 5 whose rank is 2 gets one gift

Therefore, total gifts required is 1 + 2 + 1 + 2 + 1 = 7

Similarly, for testcase 2, adhering to rules mentioned above,

Employee # 1 whose rank is 1 gets one gift

Employee # 2 whose rank is 2 gets two gifts

Therefore, total gifts required is 1 + 2 = 3

Possible solution:

Input:

2

5

1 2 1 5 2

2

1 2

C++

```
#include<bits/stdc++.h>
using namespace std;
long long arr[100010];
long long brr[100010];
int main()
{
  int test_case;
  cin >> test_case;
  for(int i = 1; i <= test_case; i++)
  {
    int n;
    long long gift = 0, temp = 0;
    cin >> n;
    for(int i = 0; i < n; i++)
    {
      cin >> arr[i];
    }
    brr[0] = 1;
    for(int i = 1; i < n; i++)
    {
      if(arr[i] > arr[i-1])
      {
        brr[i] = brr[i-1] + 1;
      }
      else
      {
        brr[i] = 1;
      }
```

```
    }
    gift = brr[n-1];
    for(int i = n-2; i >= 0; i--)
    {
      if(arr[i] > arr[i+1])
      {
        temp = brr[i+1] + 1;
      }
      else
        temp = 1;
      gift = gift + max(temp, brr[i]);
      brr[i] = temp;
    }
    cout << gift << endl;
  }
  return 0 ;
}
```

Output

7

3

Java

```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int T = sc.nextInt();
        while (T-- > 0) {
            int n = sc.nextInt();
            int[] ar = new int[n];
            for (int i = 0; i < n; i++)
                ar[i] = sc.nextInt();
            int[] gifts = new int[n];
            gifts[0] = 1;
            // Left to Right Neighbors
            for (int i = 1; i < n; i++) {
                if (ar[i] > ar[i - 1])
                  gifts[i] = gifts[i - 1] + 1;
                else
                   gifts[i] = 1;
            }
            // Right to Left Neighbors
            for (int i = n - 2; i >= 0; i--) {
```

```
            if (ar[i] > ar[i + 1] && gifts[i] <= gifts[i + 1])
                gifts[i] = gifts[i + 1] + 1;
        }
        long total = 0;
        for (int gift : gifts)
            total += gift;
        System.out.println(total);
    }
  }
}
```

Output
7

3

```
t = int(input())
while t>0:
   n = int(input())
   a = list(map(int, input().split()))
   gifts = [1]
   #gifts.append(1)
   for i in range(1, n):
     if a[i] > a[i-1]:
       gifts.append(gifts[i-1]+1)
     else:
       gifts.append(1)
   for i in range(n-2,-1,-1):
     if a[i]>a[i+1]:
       gifts[i] = max (1 + gifts[i+1], gifts[i])
   g = 0
   for i in range(n):
     g += gifts[i]
   print(g)
   t-=1
```

Output
7

3

TCS CodeVita Questions #Q4

Minimize the sum

Given an array of integers, perform atmost K operations so that the sum of elements of final array is minimum. An operation is defined as follows -

Consider any 1 element from the array, arr[i].

Replace arr[i] by floor(arr[i]/2).

Perform next operations on the updated array.

The task is to minimize the sum after utmost K operations.

Constraints

1 <= N, K <= 10^5.

Input

First line contains two integers N and K representing size of array and maximum numbers of operations that can be performed on the array respectively.

Second line contains N space separated integers denoting the elements of the array, arr.

Output

Print a single integer denoting the minimum sum of the final array.

Input

4 3

20 7 5 4

Output

17

Explanation

Operation 1 -> Select 20. Replace it by 10. New array = [10, 7, 5, 4]

Operation 2 -> Select 10. Replace it by 5. New array = [5, 7, 5, 4].

Operation 3 -> Select 7. Replace it by 3. New array = [5,3,5,4].

Sum = 17.

Possible Solution

Input:

4 3

20 7 5 4

Output

C++

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   long int n,k,temp,sum=0;
   cin>>n;
   cin>>k;
   vector<int> v;
   for(int i=0;i<n;i++)
   {
       cin>>temp;
       sum=sum + temp;
       v.push_back(temp);
   }
   make_heap(v.begin(),v.end());
   long int maxi = 0,res = 0;
   for(int i=0;i<k;i++)
   {
      maxi=v.front();
      sum-=maxi;
      pop_heap(v.begin(), v.end());
      v.pop_back();
      res = maxi / 2;
      sum+=res;
      v.push_back(res);
      push_heap(v.begin(),v.end());
   }
   cout<<sum;
}
```

Java 8

```
import java.util.*;
public class Main {
   public static void main(String[] args) throws Exception {
      // Your code  here!
      Scanner sc= new Scanner(System.in);
      // System.out.println("XXXXXXXX")
      int n,k,i,s=0;
      n=sc.nextInt();
```

```
        k=sc.nextInt();
        int a[]= new int [n];

        for(i=0;i<n;i++) a[i]=sc.nextInt();
        while(k-- > 0)
        {
           int mx=0,p=0;
           for(i=0;i<n;i++){
           if(a[i]>mx) {
              mx=a[i];
              p=i;
           }
           }
           a[p]=a[p]/2;
        }
        for(i=0;i<n;i++) s+=a[i];
        System.out.println(s);
    }
}
```

Python

```
n, k = map(int, input().split())
frequency = []
for i in range(100000):
    frequency.append(0)
sum = 0
a = list(map(int, input().split()))
for i in range(n):
    frequency[a[i]] += 1
j = 100000-1
while j>0 and k>0:
    while frequency[j]!=0 and k>0:
        k -= 1
        frequency[j] -=1
        frequency[j//2] += 1
         j-=1

for i in range(100000):
    sum += i*frequency[i]
print(sum)
```

TCS CodeVita Questions #Q5

Railway Station

Given schedule of trains and their stoppage time at a Railway Station, find minimum number of platforms needed.

Note -

If Train A's departure time is x and Train B's arrival time is x, then we can't accommodate Train B on the same platform as Train A.

Constraints

1 <= N <= 10^5

0 <= a <= 86400

0 < b <= 86400

Number of platforms > 0

Input

First line contains N denoting number of trains.

Next N line contain 2 integers, a and b, denoting the arrival time and stoppage time of train.

Output

Single integer denoting the minimum numbers of platforms needed to accommodate every train.

Example 1

Input

3

10 2

5 10

13 5

Output

2

Explanation

The earliest arriving train at time t = 5 will arrive at platform# 1. Since it will stay there till t = 15, train arriving at time t = 10 will arrive at platform# 2. Since it will depart at time t = 12, train arriving at time t = 13 will arrive at platform# 2.

Example 2

Input

2

2 4

6 2

Output

2

Explanation

Platform #1 can accommodate train 1.

Platform #2 can accommodate train 2.

Note that the departure of train 1 is same as arrival of train 2, i.e. 6, and thus we need a separate platform to accommodate train 2.

Possible Solution

C++

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
	int n;
	cin>>n;
	int a[n],b[n];
	for(int i=0;i<n;i++)
	{
		cin>>a[i]>>b[i];
		b[i]=a[i]+b[i];
	}
	sort(a,a+n);
```

```
        sort(b,b+n);
        int p=1,r=1,i=1,j=0;
        while(i<n && j<n)
        {
                if(a[i]<=b[j])
                {
                        p++;
                        i++;
                }
                else if(a[i]>b[j])
                {
                        p--;
                        j++;
                }
                if(p>r)
                        r=p;
        }
        cout<<r;
}
```

Java 8

```
import java.util.Arrays;
import java.util.Scanner;
public class Main {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      int n = sc.nextInt();
      int[] arr = new int[n];
      int[] dep = new int[n];
      for (int i = 0; i < n; i++) {
         arr[i] = sc.nextInt();
         int stoppage = sc.nextInt();
         dep[i] = arr[i] + stoppage;
      }
      Arrays.sort(arr);
      Arrays.sort(dep);
      int i = 1, j = 0, currPlatforms = 1, ans = 1;
      while (i < n && j < n) {
         if (arr[i] <= dep[j]) {
            i++;
            currPlatforms++;
         }
         else {
            currPlatforms--;
            j++;
```

```
            }
            ans = Math.max(ans, currPlatforms);
        }
        System.out.println(ans);
    }
}
```

Python 3

```
n = int(input())
arr = []
dep = []
for i in range(n):
  a, d = map(int, input().split())
  d += a
  arr.append(a)
  dep.append(d)
arr.sort()
dep.sort()
i = 1
j = 0
plf = 1
maxplf = 1
while i<n and j<n:
  if arr[i] <= dep[j]:
    i+=1
    plf+=1
    maxplf = max ( maxplf, plf )
    continue
  j+=1
  plf-=1
print(maxplf)
```

TCS CodeVita Questions #Q6

Count Pairs

Given an array of integers A, and an integer K find number of happy elements.

Element X is happy if there exists at least 1 element whose difference is less than K i.e. an element X is happy if there is another element in the range [X-K, X+K] other than X itself.

Constraints

1 <= N <= 10^5

0 <= K <= 10^5

0 <= A[i] <= 10^9

Input

First line contains two integers N and K where N is size of the array and K is a number as described above. Second line contains N integers separated by space.

Output

Print a single integer denoting the total number of happy elements.

Example 1

Input

6 3

5 5 7 9 15 2

Output

5

Explanation

Other than number 15, everyone has at least 1 element in the range [X-3, X+3]. Hence they are all happy elements. Since these five are in number, the output is 5.

Example 2

Input

3 2

1 3 5

Output

3

Explanation

All numbers have at least 1 element in the range [X-2, X+2]. Hence they are all happy elements. Since these three are in number, the output is 3.

Possible Solution

Input:

3 2

1 3 5

C++

```
#include <bits/stdc++.h>
using namespace std;
int pairs(int elementlst[],int n,int z){
 int count=0;
  for(int i=0;i<n;i++){
    int a=elementlst[i];
    int id1=i;
    int id2=i;
    if(i==0){
      while(elementlst[id2+1]==a)
        id2+=1;
      if(elementlst[id2+1]<=a+z && elementlst[id2+1]>=a-z)
        count+=1;
    }
    else if(i<n-1){
      while(elementlst[id2+1]==a)
        id2+=1;
      while(elementlst[id1-1]==a)
        id1-=1;
      if(((elementlst[id1-1]<=a+z) && (elementlst[id1-1]>=a-z)) || ((elementlst[id2+1]<=a+z)
&& (elementlst[id2+1]>=a-z)))
        count+=1;
    }
    else{
      while(elementlst[id1-1]==a)
        id1-=1;
      if(elementlst[id1-1]<=a+z && elementlst[id1-1]>=a-z)
        count+=1;
    }
  }
```

```
 return count;
}
int main() {
        int n,z;
        cin>>n>>z;
        int elementlst[n];
        for(int i=0;i<n;i++){
           cin>>elementlst[i];
        }
        sort(elementlst,elementlst+n);
        cout<<pairs(elementlst,n,z);
        return 0;
}
```

Java 8

```
import java.util.*;
public class Main {
   public static int pairs(int a[], int n , int z)
   {
      int c=0,i;
      for(i=0;i<n;i++)
      {
         int aa=a[i];
         int id1=i; int id2=i;
         if(i==0)
         {
            while(a[id2+1]==aa)
            id2++;
            if(a[id2+1]<=aa+z && a[id2+1]>=aa-z)
            c++;
         }
         else if(i<n-1)
         {
            while(a[id2+1]==aa)
            id2+=1;
            while(a[id1-1]==aa)
            id1-=1;
            if(((a[id1-1]<=aa+z) && (a[id1-1]>=aa-z)) || (
(a[id2+1]<=aa+z) && (a[id2+1]>=aa-z)))
            c+=1;
         }
         else
         {
```

```
            while(a[id1-1]==aa)
            id1-=1;
            if(a[id1-1]<=aa+z && a[id1-1]>=aa-z)
            c+=1;
          }
        }
        return c;
    }
    public static void main(String[] args) throws Exception {
        // Your code  here!
        Scanner sc= new Scanner(System.in);
        // System.out.println("XXXXXXXX")
        int n,k,i,s=0;
        n=sc.nextInt();
        k=sc.nextInt();
        int a[]= new int [n];

        for(i=0;i<n;i++) a[i]=sc.nextInt();
        Arrays.sort(a);
        System.out.print(pairs(a,n,k));
    }
}
```

Python 3

```
n, k = map(int, input().split())
a = set()
a = set(map(int, input().split()))
a = list(a)
if n==1:
    print("0")
else:
    a = sorted(a)
c = 0
for i in range(1, len(a)-1):
    if a[i]-a[i-1]>k and a[i+1]-a[i]>k :
        c += 1
if a[1]-a[0]>k:
    c+=1
if a[len(a)-1]-a[len(a)-2]>k:
    c +=1
print(n-c)
```

TCS CodeVita Questions #Q7

Critical Planets

The war between Republic and Separatists is escalating. The Separatists are on a new offensive. They have started blocking the path between the republic planets (represented by integers) so that these planets surrender due to the shortage of food and supplies. The Jedi council has taken note of the situation and they have assigned Jedi Knight Skywalker and his Padawan Ahsoka to save the critical planets from blockade (Those planets or system of planets which can be accessed by only one path and may be lost if that path is blocked by separatist).

Skywalker is preparing with the clone army to defend the critical paths. He has assigned Ahsoka to find the critical planets. Help Ahsoka to find the critical planets(C) in ascending order. You only need to specify those planets which have only one path between them and they cannot be accessed by any other alternative path if the only path is compromised.

Constraints

M <= 10000

N <= 7000

Input

First line contains two space separated integers M and N, where M denotes the number of paths between planets and N denotes the number of planets.

Next M lines, each contains two space separated integers, representing the planet numbers that have a path between them.

Output

C lines containing one integer representing the critical planet that they need to save in ascending order of the planet number if no planet is critical then print -1

Time Limit

1

Example 1

Input

3 4

0 1

1 2

2 3

Output

0

1

2

3

Explanation

Since all the planets are connected with one path and cannot be accessed by any alternative paths hence all the planets are critical.

Example 2

Input

7 6

0 2

0 1

1 2

2 3

4 5

3 4

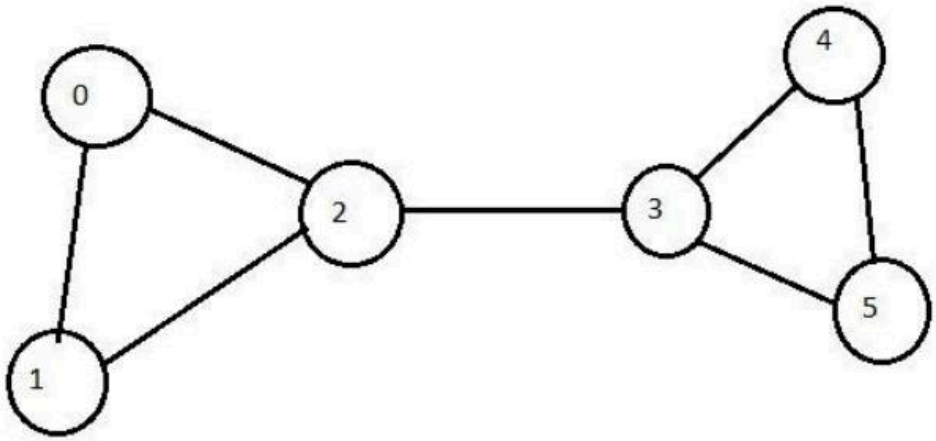
3 5

Output

2

3

Explanation



If the republic loose the path between 2 and 3 then the two system of planets will not be able to communicate with each other. Hence 2 and 3 are critical planets.

Possible Solution:

C++

```
#include <bits/stdc++.h>
typedef long long int lli;
#define pb push_back
using namespace std;
vector<int> adj[100001];
int visited[100001] , in[100001] ,low[100001];
int timer;
set<int> s;
void dfs(int node , int pre)
{
  visited[node] = 1;
  in[node] = low[node] = timer;
  timer++;
  for(int i : adj[node])
  {
    if(i == pre)
```

```
      continue;
    if(visited[i] == 1)
    {
     low[node] = min(low[node] , in[i]);
    }
    else
    {
     dfs(i , node);
     if(low[i] > in[node])
      s.insert(node) , s.insert(i);
     low[node] = min(low[node] , low[i]);
    }
  }
}
int main()
{
  int edge, vertex , a , b;
  cin >> edge >> vertex;
  for(int i = 0;i < edge;i++)
  {
    cin >> a >> b;
    adj[a].pb(b);
    adj[b].pb(a);
  }
  dfs(0 , -1);
  for(int i : s)
    cout << i << endl;
   return 0;
}
```

Java 8

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.TreeSet;
public class Main {
  static List<List<Integer>> graph;
  static TreeSet<Integer> criticalPlanets;
  static boolean[] visited;
  static int[] inTime;
  static int[] low;
  static int timer = 0;
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int m = sc.nextInt();
```

```
    int n = sc.nextInt();
    graph = new ArrayList<>(n);
    for (int i = 0; i < n; i++)
      graph.add(new ArrayList<>());
    visited = new boolean[n];
    inTime = new int[n];
    low = new int[n];
    criticalPlanets = new TreeSet<>();
    for (int i = 0; i < m; i++) {
      int u = sc.nextInt();
      int v = sc.nextInt();
      graph.get(u).add(v);
      graph.get(v).add(u);
    }
    dfs(0, -1);
    for (int i : criticalPlanets)
      System.out.println(i);
  }
  public static void dfs(int node, int parent) {
    visited[node] = true;
    inTime[node] = low[node] = timer++;
    for (int neighbour : graph.get(node)) {
      if (neighbour == parent)
        continue;
      if (visited[neighbour])
        low[node] = Math.min(low[node], inTime[neighbour]);
      else {
        dfs(neighbour, node);
        if (low[neighbour] > inTime[node]) {
          criticalPlanets.add(neighbour);
          criticalPlanets.add(node);
        }
        low[node] = Math.min(low[node], low[neighbour]);
      }
    }
  }
}
```

Python 3

```
timer=0
def dfs (cur, par, timer):
    vis[cur]=1
    if cur!=0:
      low[cur] = trav[cur] = trav[par] + 1
```

```
    timer += 1
    for i in g[cur]:
        if i == par:
            continue
        if vis[i]==1:
            low[cur] = min ( low[cur], trav[i] )
        else:
            dfs(i, cur, timer)
            if low[i]>low[cur]:
                defend.add(i)
                defend.add(cur)
            low[cur] = min (low[cur], low[i])
e, n = map(int, input().split())
g = []
vis = []
low = []
trav = []
for i in range(n):
  l = []
  vis.append(0)
  low.append(0)
  trav.append(0)
  g.append(l)
for i in range(e):
  a, b = map(int, input().split())
  g[a].append(b)
  g[b].append(a)
defend = set()
dfs(0, -1, timer)
defend = list(defend)
defend.sort()
for i in defend:
  print(i)
```


TCS CodeVita Questions (Previous Years)

Practicing the previous year's TCS CodeVita questions will better understand the standard of questions asked.

Q1. Consecutive Prime Sum

Some prime numbers can be expressed as a sum of other consecutive prime numbers. For example 5 = 2 + 3, 17 = 2 + 3 + 5 + 7, 41 = 2 + 3 + 5 + 7 + 11 + 13. Your task is to find out how many prime numbers which satisfy this property are present in the range 3 to N subject to a constraint that summation should always start with number 2.

Write code to find out the number of prime numbers that satisfy the above-mentioned property in a given range.

S.no	Input	Output	Comment
1	20	2	(Below 20, there are 2 such members: 5 and 17) 5 = 2+3 17 = 2+3+5+7
2	15	1	

Input Format: First line contains a number N

Output Format: Print the total number of all such prime numbers which are less than or equal to N.

Constraints: 2<N<=12,000,000,000

Sample Possible solution:

```
#include <stdio.h>
int prime(int b)
{
   int j,cnt;
  cnt=1;
    for(j=2;j<=b/2;j++)
    {
       if(b%j==0)
       cnt=0;
    }
    if(cnt==0)
    return 1;
    elsereturn 0;
}
int main() {
 int i,j,n,cnt,a[25],c,sum=0,count=0,k=0;
 scanf("%d",&n);
 for(i=2;i<=n;i++)
 {
    cnt=1;
    for(j=2;j<=n/2;j++)
    {
       if(i%j==0)
       cnt=0;
    }
    if(cnt==1)
    {
      a[k]=i;
      k++;
      }
```

```
}
for(i=0;i<k;i++)
{
   sum=sum+a[i];
  c= prime(sum);
  if(c==1)
  count++;
}
printf("%d",count);
return 0;
}
```

Output:

```
20
2
```

Q2. Bank Compare

There are two banks – Bank A and Bank B. Their interest rates vary. You have received offers from both banks in terms of the annual rate of interest, tenure, and variations of the rate of interest over the entire tenure.You have to choose the offer which costs you least interest and reject the other. Do the computation and make a wise choice.

The loan repayment happens at a monthly frequency and Equated Monthly Installment (EMI) is calculated using the formula given below :

EMI = loanAmount * monthlyInterestRate / (1 – 1 / (1 + monthlyInterestRate)^(numberOfYears * 12))

Constraints:

1 <= P <= 1000000

1 <=T <= 50

1<= N1 <= 30

1<= N2 <= 30

Input Format:

- First line: P principal (Loan Amount)
- Second line: T Total Tenure (in years).
- Third Line: N1 is the number of slabs of interest rates for a given period by Bank A. First slab starts from the first year and the second slab starts from the end of the first slab and so on.
- Next N1 line will contain the interest rate and their period.
- After N1 lines we will receive N2 viz. the number of slabs offered by the second bank.
- Next N2 lines are the number of slabs of interest rates for a given period by Bank B. The first slab starts from the first year and the second slab starts from the end of the first slab and so on.
- The period and rate will be delimited by single white space.

Output Format: Your decision either Bank A or Bank B.

Explanation:

Example 1

Input

10000

20

3

5 9.5

10 9.6

5 8.5

3

10 6.9

5 8.5

5 7.9

Output: Bank B

Example 2

Input

500000

26

3

13 9.5

3 6.9

10 5.6

3

14 8.5

6 7.4

6 9.6

Output: Bank A

Possible solution:

```
#include #includeint main() {
double p,s,mi,sum,emi,bank[5],sq;
int y,n,k,i,yrs,l=0;
  scanf("%lf",&p);
scanf("%d",&y);
for(k=0;k<2;k++)
{
scanf("%d",&n);
sum=0;
for(i=0;i<n;i++)
```

```
{
 scanf("%d",&yrs);
 scanf("%lf",&s);
 mi=0;
 sq=pow((1+s),yrs*12);
 emi= (p*(s))/(1-1/sq);
 sum= sum + emi;
}bank[l++]=sum;
}
if(bank[0]<bank[1])

printf(“Bank A”);

else

printf(“Bank B”);

return 0;

}
```


Q3. Counting Rock Samples

Juan Marquinho is a geologist and he needs to count rock samples in order to send it to a chemical laboratory. He has a problem: The laboratory only accepts rock samples by a range of its size in ppm (parts per million).

Juan Marquinho receives the rock samples one by one and he classifies the rock samples according to the range of the laboratory. This process is very hard because the number of rock samples may be in millions.

Juan Marquinho needs your help, your task is to develop a program to get the number of rocks in each of the ranges accepted by the laboratory.

Input Format:

An positive integer S (the number of rock samples) separated by a blank space, and a positive integer R (the number of ranges of the laboratory); A list of the sizes of S samples (in ppm), as positive integers separated by space R lines where the ith line containing two positive integers, space separated, indicating the minimum size and maximum size respectively of the ith range.

Output Format:

R lines where the ith line containing a single non-negative integer indicating the number of the samples which lie in the ith range.

Constraints: 10 ? S ? 10000 1 ? R ? 1000000 1?size of each sample (in ppm) ? 1000

Example 1

Input: 10 2

345 604 321 433 704 470 808 718 517 811

300 350

400 700

Output: 2 4

Explanation:

There are 10 samples (S) and 2 ranges (R). The samples are 345, 604,811. The ranges are 300-350 and 400-700. There are 2 samples in the first range (345 and 321) and 4 samples in the second range (604, 433, 470, 517). Hence the two lines of the output are 2 and 4

Example 2

Input: 20 3

921 107 270 631 926 543 589 520 595 93 873 424 759 537 458 614 725 842 575 195

1 100

50 600

1 1000

Output: 1 12 20

Explanation:

There are 20 samples and 3 ranges. The samples are 921, 107 195. The ranges are 1-100, 50-600 and 1-1000. Note that the ranges are overlapping. The number of samples in each of the three ranges are 1, 12 and 20 respectively. Hence the three lines of the output are 1, 12 and 20.

Possible Solution:

```
#include
int main() {
int a[1000],s,i,j,t,l1,l2,c=0;
scanf("%d",&s);
scanf("%d",&t);
for(i=0;i<s;i++)
scanf("%d",&a[i]);
for(i=0;i<t;i++)
{
scanf("%d %d",&l1,&l2);
for(j=0;j<s;j++)
{
 if((a[j]>=l1)&&(a[j]<=l2))
 c++;
}
printf("%dn ",c);
c=0;
}
```

```
return 0;
}
```

Q4. kth largest factor of N

A positive integer d is said to be a factor of another positive integer N if when N is divided by d, the remainder obtained is zero. For example, for number 12, there are 6 factors 1, 2, 3, 4, 6, 12. Every positive integer k has at least two factors, 1 and the number k itself.Given two positive integers N and k, write a program to print the kth largest factor of N.

Input Format: The input is a comma-separated list of positive integer pairs (N, k)

Output Format: The kth highest factor of N. If N does not have k factors, the output should be 1.

Constraints: 1<N<10000000000. 1<k<600. You can assume that N will have no prime factors which are larger than 13.

Example 1

- **Input**: 12,3

- **Output**: 4

Explanation: N is 12, k is 3. The factors of 12 are (1,2,3,4,6,12). The highest factor is 12 and the third-largest factor is 4. The output must be 4

Example 2

- **Input**: 30,9
- **Output**: 1

Explanation: N is 30, k is 9. The factors of 30 are (1,2,3,5,6,10,15,30). There are only 8 factors. As k is more than the number of factors, the output is 1.

Possible Solution:

```
#include <stdio.h>
int main() {
int n,k,i,c=0;
scanf("%d",&n);
scanf("%d",&k);
for(i=n;i>=1;i--)
{
 if((n%i)==0)
 c++;
 if(c==k)
 {
 printf("%d",i);
```

```
 break;
 }
}
if(c!=k)
printf("1");
return 0;
}
```

Q5. Collecting Candies

Krishna loves candies a lot, so whenever he gets them, he stores them so that he can eat them later whenever he wants to.

He has recently received N boxes of candies each containing Ci candies where Ci represents the total number of candies in the ith box. Krishna wants to store them in a single box. The only constraint is that he can choose any two boxes and store their joint contents in an empty box only. Assume that there are infinite number of empty boxes available.

At a time he can pick up any two boxes for transferring and if both the boxes say contain X and Y number of candies respectively, then it takes him exactly X+Y seconds of time. As he is too eager to collect all of them he has approached you to tell him the minimum time in which all the candies can be collected.

Input Format:

- The first line of input is the number of test case T
- Each test case is comprised of two inputs
- The first input of a test case is the number of boxes N
- The second input is N integers delimited by whitespace denoting the number of candies in each box

Output Format: Print minimum time required, in seconds, for each of the test cases. Print each output on a new line.

Constraints:

- 1 ?T?10
- 1 ?N? 10000
- 1 ? [Candies in each box] ? 100009

Sample Input and Output:

1	1 4 1 2 3 4	19
2	1 5 1 2 3 4	34

The explanation for sample input-output 1:

4 boxes, each containing 1, 2, 3, and 4 candies respectively.Adding 1 + 2 in a new box takes 3 seconds.Adding 3 + 3 in a new box takes 6 seconds.Adding 4 + 6 in a new box takes 10 seconds.Hence total time taken is 19 seconds. There could be other combinations also, but overall time does not go below 19 seconds.

The explanation for sample input-output 2:

5 boxes, each containing 1, 2, 3, 4, and 5 candies respectively.Adding 1 + 2 in a new box takes 3 seconds.Adding 3 + 3 in a new box takes 6 seconds.Adding 4 + 6 in a new box takes 10 seconds.Adding 5 + 10 in a new box takes 15 seconds.Hence total time taken is 34 seconds. There could be other combinations also, but overall time does not go below 33 seconds.

Possible Solution:

```
#include <stdio.h>
int main() {
int n,i,k=0,sum=0,s1=0,t,temp=0,j;
long c[100009],s[100009];
scanf("%d",&t);
for(int l=0;l<t;l++)
{
scanf("%d",&n);
for(i=0;i<n;i++)
scanf("%ld",&c[i]);
for(i=0;i<n;i++)
```

```
{
for(j=i+1;j<n;j++)
{
if(c[i]>c[j])
{
temp=c[i];
c[i]=c[j];
c[j]=temp;
}
}
}
sum=0;
k=0;
for(i=0;i<n;i++)
{
sum=sum+c[i];
s[k]=sum;
k++;
}
s1=0;
for(i=1;i<k;i++)
s1=s1+s[i];
printf("%dn",s1);
}
return 0;
}
```

Q6. Football League

Football League Table Statement : All major football leagues have big league tables. Whenever a new match is played, the league table is updated to show the current rankings (based on Scores, Goals For (GF), Goals Against (GA)). Given the results of a few matches among teams, write a program to print all the names of the teams in ascending order (Leader at the top and Laggard at the bottom) based on their rankings.

Rules: A win results in 2 points, a draw results in 1 point and a loss is worth 0 points. The team with the most goals in a match wins the match. Goal Difference (GD) is calculated as Goals For (GF) Goals Against (GA). Teams can play a maximum of two matches against each other Home and Away matches respectively.

The ranking is decided as follows: Team with maximum points is ranked 1 and minimum points is placed last Ties are broken as follows Teams with same points are ranked according to Goal Difference(GD).

If Goal Difference(GD) is the same, the team with higher Goals For is ranked ahead

If GF is same, the teams should be at the same rank but they should be printed in case-insensitive alphabetic according to the team names. More than 2 matches of same teams, should be considered as Invalid Input.

A team can't play matches against itself, hence if team names are same for a given match, it should be considered Invalid Input

Input Format: First line of input will contain number of teams (N) Second line contains names of the teams (Na) delimited by a whitespace character Third line contains number of matches (M) for which results are available Next M lines contain a match information tuple {T1 T2 S1 S2}, where tuple is comprised of the following information

- T1 Name of the first team
- T2 Name of the second team
- S1 Goals scored by the first team
- S2 Goals scored by the second team

Output Format: Team names in order of their rankings, one team per line OR Print "Invalid Input" where appropriate.

Constraints: 0< N <=10,000 0<=S1,S2

Example: Consider 5 teams Spain, England, France, Italy, and Germany with the following fixtures:

- Match 1: Spain vs. England (3-0) (Spain gets 2 points, England gets 0)
- Match 2: England vs. France (1-1) (England gets 1 point, France gets 1)
- Match 3: Spain vs. France (0-2) (Spain gets 0 points, France gets 2)

Table 1. Points Table after 3 matches

Spain	2	3	2	1
England	2	1	4	-3
France	2	3	1	2
Italy	0	0	0	0
Germany	0	0	0	0

Since Italy and Germany are tied for points, the goal difference is checked. Both have the same, so, Goals For is checked. Since both are the same. Germany and Italy share the 4th rank. Since Germany appears alphabetically before Italy, Germany should be printed before Italy. Then the final result is: France Spain England Germany Italy

Sample:

S.no	Input	Output
1	5 Spain England France Italy Germany 3 Spain England 3 0 England France 1 1 Spain France 0 2	France Spain England Germany Italy
2	5 Spain England France Italy Germany 3 Spain England 3 0 England France 1 1 Spain Spain 0 2	Invalid input

Q7. Sorting Boxes

The parcel section of the Head Post Office is in a mess. The parcels that need to be loaded to the vans have been lined up in a row in an arbitrary order of weights. The Head Post Master wants them to be sorted in the increasing order of the weights of the parcels, with one exception.He wants the heaviest (and presumably the most valuable) parcel kept nearest his office.

You and your friend try to sort these boxes and you decide to sort them by interchanging two boxes at a time. Such an interchange needs effort equal to the product of the weights of the two boxes.

The objective is to reposition the boxes as required with minimum effort.

Input Format: The first line consists of two space-separated positive integers giving the number of boxes (N) and the position of the Head Post Masters office (k) where the heaviest box must be.

The second line consists of N space separated positive integers giving the weights of the boxes. You may assume that no two weights are equal.

Output Format: The output is one line giving the total effort taken to get the boxes in sorted order, and the heaviest in position k.

Constraints: N<=50 and Weights <= 1000

Sample Input-output:

S.no	Input	Output
1	5 2 20 50 30 80 70	3600
2	6 3 30 20 40 80 70 60	Invalid input

Q8. On A Cube

A solid cube of 10 cm x 10cm x 10 cm rests on the ground. It has a beetle on it, and some sweet honey spots at various locations on the surface of the cube. The beetle starts at a point on the surface of the cube and goes to the honey spots in order along the surface of the cube.

- If it goes from a point to another point on the same face (say X to Y), it goes in an arc of a circle that subtends an angle of 60 degrees at the center of the circle
- If it goes from one point to another on a different face, it goes by the shortest path on the surface of the cube, except that it never travels along the bottom of the cube

The beetle is a student of cartesian geometry and knows the coordinates (x, y, z) of all the points it needs to go to. The origin of coordinates it uses is one corner of the cube on the ground, and the z-axis points up.Hence, the bottom surface (on which it does not crawl) is z=0, and the top surface is z=10.The beetle keeps track of all the distances traveled, and rounds the distance traveled to two decimal places once it reaches the next spot so that the final distance is a sum of the rounded distances from spot to spot.

Input Format: The first line gives an integer N, the total number of points (including the starting point) the beetle visits

The second line is a set of 3N comma separated non-negative numbers, with up to two decimal places each. These are to be interpreted in groups of three as the x, y, z coordinates of the points the beetle needs to visit in the given order.

Output Format: One line with a number giving the total distance traveled by the beetle accurate to two decimal places. Even if the distance traveled is an integer, the output should have two decimal places.

Constraints: None of the points the beetle visits is on the bottom face (z=0) or on any of the edges of the cube (the lines where two faces meet)

2<=N<=10

Sample Input-Output:

```
Input
3
1,1,10,2,1,10,0,5,9
Output
6.05
Input
3
1,1,10,2,1,10,0,1,9
Output
4.05
```

Q9. Coins Distribution Question (or Coins Required Question)

Problem Description

Find the minimum number of coins required to form any value between 1 to N, both inclusive. Cumulative value of coins should not exceed N. Coin denominations are 1 Rupee, 2 Rupee and 5 Rupee.

Let's understand the problem using the following example. Consider the value of N is 13, then the minimum number of coins required to formulate any value between 1 and 13, is 6. One 5 Rupee, three 2 Rupee and two 1 Rupee coins are required to realize any value between 1 and 13. Hence this is the answer.

However, if one takes two 5 Rupee coins, one 2 rupee coins and two 1 rupee coins, then to all values between 1 and 13 are achieved. But since the cumulative value of all coins equals 14, i.e., exceeds 13, this is not the answer.

Input Format

A single integer value

Output Format

Four Space separated Integer Values

1st – Total Number of coins

2nd – number of 5 Rupee coins.

3rd – number of 2 Rupee coins.

4th – number of 1 Rupee coins.

Constraints

0<n<1000

Sample Input:

13

Sample Output:

6 1 3 2

Explanation:

The minimum number of coins required is 6 with in it:

minimum number of 5 Rupee coins = 1

minimum number of 2 Rupee coins = 3

minimum number of 1 Rupee coins = 2

Using these coins, we can form any value with in the given value and itself, like below: Here the given value is 13

For 1 = one 1 Rupee coin

For 2 = one 2 Rupee coin

For 3 = one 1 Rupee coin and one 2 Rupee coins

For 4 = two 2 Rupee coins

For 5 = one 5 Rupee coin

For 6 = one 5 Rupee and one 1 Rupee coins

For 7 = one 5 Rupee and one 2 Rupee coins

For 8 = one 5 Rupee, one 2 Rupee and one 1 Rupee coins

For 9 = one 5 Rupee and two 2 Rupee coins

For 10 = one 5 Rupee, two 2 Rupee and one 1 Rupee coins

For 11 = one 5 Rupee, two 2 Rupee and two 1 Rupee coins

For 12 = one 5 Rupee, three 2 Rupee and one 1 Rupee coins

For 13 = one 5 Rupee, three 2 Rupee and two 1 Rupee coins

Solution in Python:

```
#get the input
number = int(input())



#find the number of five rupee coins#maximize possible number of coins, after leavinf out 4 rupees t
o make all possible values till 4
five = int((number-4)/5)



#find the number of one rupee coins#of the remaining value, number of one rupee coins is 1, if the v
alue is odd & it is 2 if the value is evenif((number-5*five) % 2) == 0:
    one=2else:
    one=1#find the number of two rupee coins#the rest of the amount will be from two rupee coins
two=(number-5*five-one)//2#print total coins, five ruppe coins, two rupee coins, one rupee coins
print(one+two+five,five,two,one)
```

Q10. Philaland Coins Question

Problem Description

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task to make the purchase of items at the Island easier by distributing various coins with different values.

Manish has come up with a solution that if we make coins category starting from $1 till the maximum price of the item present on Island, then we can purchase any item easily. He added following example to prove his point. Let's suppose the maximum price of an item is 5$ then we can make coins of {$1, $2, $3, $4, $5} to purchase any item ranging from $1 to $5. Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {$1, $2, $3}. According to him, any item can be purchased one time ranging from $1 to $5. Everyone was impressed with both of them. Your task is to help Manisha come up with the minimum number of denominations for any arbitrary max price in Philaland.

Input Format

First line contains an integer T denoting the number of test cases. Next T lines contains an integer N denoting the maximum price of the item present on Philaland.

Output Format

For each test case print a single line denoting the minimum number of denominations of coins required.

Constraints

1<=T<=100

1<=N<=5000

Sample Input 1:

2

10

5

Sample Output 1:

4

3

Sample Input2:

3

1

5

7

Sample Output2:

1

3

3

Explanation:

For test case 1, N=10.

According to Manish {$1, $2, $3,... $10} must be distributed. But as per Manisha only {$1, $2, $3, $4} coins are enough to purchase any item ranging from $1 to $10. Hence minimum is 4. Likewise denominations could also be {$1, $2, $3, $5}. Hence answer is still 4.

For test case 2, N=5.

According to Manish {$1, $2, $3, $4, $5} must be distributed. But as per Manisha only {$1, $2, $3} coins are enough to purchase any item ranging from $1 to $5. Hence minimum is 3. Likewise denominations could also be {$1, $2, $4}. Hence answer is still 3.

Solution in Python:

```
#get input of number of cases
cases=int(input())



#for each case, find the coins requiredfor i in range(1,cases+1):
    #get input of value
    value=int(input())
```

```
#set coin counter at 0
coincount = 0#count the number of coins #logic number of coins required will be one more than th
e earlier rupee value that was a power of 2while value>=1:
    value=value//2
    coincount=coincount+1#print the answer    print (coincount)
```

Frequently Asked Questions (FAQs)

Q1. I'm getting the message "User already logged in. Please try after 15 minutes". What do I do?
A1. If you're getting error "User already logged in. Please try after 15 mins", it means you have not logged out properly of your previous session. The app will give you the option to log out of all your previous sessions. Once you click that, all previous sessions will be invalidated, and you can login to a fresh session. If you're still getting an error, kindly write to us

Q2. "How to log in and take part in the contest?"
A2. Taking part in CodeVita Season 13 is a three-part activity.
1. One qualifies as per the eligibility criteria.
2. One has registered and has achieved 100% Profile completion status
3. On the day of the contest, participants can directly log in to the contest site https://codevita.tcsapps.com They can use their registered Email ID or Username to login, once they have entered Username or Email ID they will be prompted to enter the One-Time Code (OTC) from the Microsoft Authenticator App that they have configured during the registration for authentication. Once into the contest, the timer will start and run for the next 6 hours irrespective of personal breaks. No submissions can be made post expiry of these 6 hours.

Q3. "Can I submit solution in any language?"
A3. CodeVita supports 8 programming languages viz. C, C++, C#, Java, Perl, PHP, Python and Ruby. Extensions corresponding to their source code files are .c, .cpp, .cs, .java, .pl, .php, .py and .rb respectively. Only these submissions will be accepted by the system.

Q4. "What does attribution of code mean?"
A4. Attribution means the practice of acknowledging the original source. CodeVita is an open book contest which allows participants to refer to the internet. During the CodeVita contest if the participants have referred to any web resource, then they are expected to

declare the actual web URL. This is referred to as ‘Attribution of Code’
If the participant has written the entire code from scratch by oneself, then the ‘Attribution of code’ does not apply.

Q5. “Where do I mention the attributed code?”

A5. Before you submit your solution on CodeVita system, there will be an option which you have to select – whether you have referred to external source code. If ‘Yes’, then you must give the entire details of the source code. You will have to provide a complete link/URL of the code which you have referred to.
Code attribution must be done at submission time only. Offline mails to attach references to submitted code will not be entertained and should not be attempted also.

Q6. “What is proper attribution of Code?”
A6. Providing the entire link/URL of the source code is proper attribution of Code. Through the link/URL we should be able to access/view the lines of source code that are used in your submissions.
Generic entries like ‘Google.com’, ‘www.codechef.com’, ‘stackoverflow’, ‘stackoverflow.com’, ‘Friend’s Code’, ‘Internet’, and other incomplete entries will not be accepted.

Q7. “What happens if I do not attribute the code I have referred to?”
A7. If two or more people’s submission contains the same code that is referred to from an external source, the system has no way to know that, unless proper attribution is provided. This will lead to code getting tagged as plagiarized and the participant will not be considered for the next round of the contest. It is in the participant’s best interest that s/he provides attribution for all code referred to from external sources during the Submit / Submit (final) actions.

Q8. “I am getting errors like ‘program doesn't exist’, '404 error’, and sometimes my submit button is not visible.”
A8. Before logging in, request you to kindly delete cached files by pressing Ctrl + Shift + Del, then ensure the checkbox for 'Cached Images and Files' is checked, and then click on 'Clear Data'. Ignoring this step might cause the erroneously cached files to misbehave on the site.

Q9. “Can I take personal breaks in between the contest?”
A9. Start the contest only once you are ready. Once the contest is started the timer will continue for the next 6 hours irrespective of personal breaks. You must manage your time accordingly. For breaks, please log out of the system by pressing the logout button otherwise there will be a waiting period of 15 minutes to log-in again.

Q10. “When I click on compile and run, its showing ‘Problem does not exists’ or “I am not able to view the ‘Upload solution’ button while submitting my code” Or “I get an error message that site cannot be reached”
A10. Please crosscheck this troubleshooting checklist:
– Ensure you are using a wired/reliable network. Mobile network usage is not

recommended, as there can be network fluctuations.
– Compatible browsers are Chrome and Firefox.
– Browser being used should be updated to the latest versions.
– Ensure that JavaScript is enabled on your browser.
– Ensure that any org/institute/system level proxy is not blocking any content of CodeVita site.
– Clear the browser cache and download files, then try again.
If you use 'Online Editor' option and while trying to submit the code you find the 'Problem Does not Exist' error, please "logout" of the current session, close any online editor tabs and then try to login again.

Q11. “Since I have logged in, my clock is showing my time as 00:00:00.”
A11. CodeVita timer uses hybrid method of maintaining every participant’s time. That is, it uses the server time in combination with local system time. If your contest is not over and you think that showing above time is wrong, then often your local system time is incorrect. Please rectify the same. Please clear browser cache and files history. Then logout and login again for the new time to get reflected. You can also refer contest end time written below to timer.

Q12. “I want the answers to all the questions asked in the contest”
A12. Please note that we do not reveal the answers to the CodeVita questions.
You must successfully solve the question in the system to know the correct answer.

Handling various status messages found in CodeVita:

1. Compile Time Error:
A successful compilation simply returns silently. Hence your aim should be that your
program is so agreeable with the compiler that the compiler happily returns silently.
If you get a CTE, do as follows.

- First and foremost, you must ensure that you use the same versions of compilers that the server-side uses. A list of compilers is provided here.
- If you are using the same compiler, then mentally you must treat Warnings as errors because warnings prevent the compilers from returning silently. So, get rid of all Warnings in your compilation process.
- If you have meticulously followed the above, it is highly unlikely that you will get CTE.
- In extremely rare cases, a negligibly small possibility exists that under heavy load, the servers and hence the compilers malfunction and hence your compilation fails. The probability of this happening is less than 0.01% because CodeVita engineering is now mature enough to handle thousands of concurrent compilations. In your thinking you should simply discount the possibility that the compiler has malfunctioned, but if you have good enough reason to suspect that this is what might have happened, then simply resubmit the code after some time. This kind of behavior is temporal and the probability of CTE vanishing on its own is high.
- Finally, whenever a CTE occurs the CodeVita Judge provides the exact error message that the compiler has produced. Using this message as a clue you should be able to troubleshoot past your compilation problems.

2. Runtime Error (RTE):
A Runtime error is caused because either your program or the runtime has thrown some

exception. Since RTE could be caused because of N number of reasons it is difficult to pin-
point and hence provide a crisp error message unlike CTE.

Let us divide RTE into two parts
1. Systemic Faults and
2. Submitted Program Faults.

Systemic Faults: Systemic Faults can give rise to RTE if the Judge is not properly configured to evaluate submissions. In most cases the configurations are on a per-language basis.
However systemic faults exhibit binary behavior i.e., either it will work for all submissions in a given language or it may work for none of the submissions in that given language. It cannot be that some programs in a given language receive RTE and some don't. If this happens to be the case, then it is almost always Submitted Program Fault.

It is also possible that one language is configured properly, but some others aren't. Let's say that C is properly configured, and Java is not properly configured. In this case all the C submissions will be devoid of Systemic Faults whereas all the Java programs will be susceptible to Systemic Faults. Faults in configuration of one language do not have an impact on behavior of other language. It is the duty and task of CodeVita Engineering team that systemic faults are eliminated before the Judge is thrown open to CodeVita participants. Unlike CTE, RTE is not a transient fault and does not change behavior even under load.

Submitted Program Faults: In 99.99% of cases, RTE is caused by Submitted Program Faults. Very rarely, and we do not recollect any instance in past seasons of CodeVita that a Systemic Fault causing RTE has ever been exposed in Live Rounds. Submitted Program Faults could be caused because of any reasons, but not limited to those stated below:

- Failure to adhere to input and output specifications is the number one cause of RTE.
- Known programming violations like Illegal memory access or Null Pointer Exception etc. results in RTE.
- RTE are more common to languages like C and C++ where static time type checking is lenient and hence faults manifest only at runtime.
- Any logical mistake that leads to throwing an exception receives an RTE. If you have participated in previous seasons of CodeVita, I request you share your stories on how you got past RTEs.

3. Time Limit Exceeded (TLE):

In automated code evaluation environments optimal utilization of shared resources such as CPU and memory are key to delivering good performance. However, no matter how well a platform is engineered, performance can deteriorate if submitted code is a CPU or Memory hog. So, one poorly written program can affect the evaluation times of several others. In such situations, platforms, including CodeVita, have no choice but to abort the rogue program. The threshold when this behavior kicks in, in CodeVita, is different for different questions.

For example, if a problem is purely compute-intensive i.e., has CPU affinity then the programs may have a smaller threshold, say 1 second. Likewise, if a program requires a lot of memory accesses to be performed, it may have a slightly higher threshold, say 2 seconds. Problem text explicitly articulates the time limit under which the program should finish in, in units of seconds. An intelligent reader will have already figured out that the moment this status message is received, one must minimize the runtime of the program.

To cite a contextual example, let's say a question in CodeVita requires you to sort millions of elements. If you implement a naive algorithm like Bubble Sort whose Order Complexity is O(n^2) you are almost certain to receive a TLE. A solution to get past TLE would be to implement a better sorting algorithm, say Quick Sort whose Order Complexity is O (n log n).

This will drastically reduce the sorting time and the program can finish within thresholds.
Now that we have seen what a TLE is and why it occurs and how system responds to it, let's see some of the ways in which you can overcome TLE

- It is good practice to insert timestamps in your code to know how much time is spent in different parts of your code. So, in case you get a TLE, you already know where your bottlenecks are.
- Your choice of data structure and algorithm plays a critical role in assessing whether you will or will not receive a TLE.
- Keep a profiler handy and more importantly know how to use it so that in case of TLE you may get insights into the runtime of your code.
- With TLE, some good programmers have a reverse gripe i.e., they write so optimized code that they feel that the thresholds are too lenient. Such programmers are advised to have patience. May be with a few questions, few programmers can get away even with sub-optimal code, but it cannot happen always. Keep up the good habit of writing optimized code. There will come a question where only optimal code will pass, and the rest will receive TLE. It's just that not all questions are geared towards figuring out if the participant can write optimal code. So please don't be too hung up if you have written a (n^2) complexity algorithm and your friend's (n^3) implementation also passes the evaluation.

So, all the best to you all and let there be no more TLEs. Sophisticated programmers use many brilliant techniques, like commenting on certain sections of the code and figure out the bottlenecks if a TLE status gets converted into Wrong Answer status.

4. Memory Limit Exceeded (MLE):
Just like TLE appears due to longer than allowed runtime execution times, MLE occurs due to higher than permissible memory utilization. More memory utilization is a function of two things
– the language of your choice
– how you handle memory allocation / deallocation in your code

Memory footprint of languages: Standalone languages like C and C++ have better memory footprint than managed runtime languages like Java and C#. Interpreted languages like Perl, Python, Ruby etc. are somewhere in between. CodeVita systems are 64-bit, and we calibrate the footprint of languages before setting memory limits on them. It can be safely said for the kind of problems that are asked in CodeVita, the memory footprint of language runtimes far exceed the amount of memory that a program may possibly need to use to arrive at the correct solution. Hence MLE status is almost always due to poor memory management strategy implemented in the program.
Memory Management in own code: Statically allocating large chunks of memory or speculatively allocating memory based on own understanding of questions is usually the main reason for MLE. Other than that poor data structures and algorithms also cause more than permissible usage of memory. In CodeVita, memory footprint of every process is tracked and rogue processes using more memory are terminated and a status of MLE is returned to the submitter of that program. There are no general rules on how to reduce memory utilization. The rules are language and context dependent. So, ensure that you are aware of how-to techniques to reduce memory utilization in language of your choice.

5. Wrong Answer:
Wrong Answer is caused because your program didn't give the same output as expected for 1 or more test cases.
Whatever questions are being asked in CodeVita goes through thorough testing. So even if your program executes successfully on your system, which does not indicate its correctness. It should pass through all the private test cases of our problem. So before raising any queries, verify your program thoroughly.

6. Accepted:
Accepted status comes when your program has passed all the test cases i.e., it provided the same output as expected.
If your program shows this status, your problem is solved, and you should move on to the next problem.

7. Presentation Error:
Presentation Error status comes when your program output differs from the expected output by a whitespace character.
Even If your program shows this status, your problem is considered as solved. So don't try to get it in Accepted status and move on to the next problem.

8. Difference between Accepted and Presentation Error:
Both Accepted and Presentation Error status are the same. If either of these two statuses come while evaluating your program, consider it as solved and move on to the next problem.

9. Public Test Case Vs Private Test Case:
While submitting solutions on CodeVita site, if using Code Editor, upon clicking on Compile and Run button, The program gets evaluated for public testcases only which are given in problem text. Getting a Presentation Error or Accepted status here only means that your program is compiling and executing successfully against public testcases. Simply executing Compile and Run is not enough. In order to run private testcases against programs one needs to click on the Submit (Final) button. This submission will be considered for Ranking. This can end up in any of the 8 statuses. Also, getting Accepted or Presentation Error while Compile and Run does not guarantee that the same status will appear after Submit (Final) functionality. As described above, this is because Compile and Run is run against public testcases while Submit (Final) is run against private testcases. Public testcases are those testcases which are present in problem text whereas private testcases are hidden and never known to participants.