Advanced Coding

1.Problemstatement:

Given two non-negative integers n1 and n2, where n1 <n2. The task is to find the total number of integers in the range interval [n1, n2] [both inclusive] which have no repeated digits.

Fore.g.

Supposen1=11and n2=15.

There is the number 11,which has repeated digits, but 12, 13, 14,and 15have no repeated digits. So, the output is 4.

Input	Output
11--Valueofn1 15--Valueofn2	4
101--Valueofn1 200--Valueofn2	72

Code Solution in Java

```java
import java.util.*;
class Main
{
static int repeated_digit(int n)
{
    LinkedHashSet<Integer> s = new LinkedHashSet<>();
    while (n != 0)
    {
        int d = n % 10;
        if (s.contains(d))
        {
            return 0;
        }
        s.add(d);
        n = n / 10;
    }
    return 1;
}
static int calculate(int L, int R)
{
    int answer = 0;
    for (int i = L; i < R + 1; ++i)
    {
        answer = answer + repeated_digit(i);
    }
    return answer;
}
public static void main(String[] args)
{
    Scanner sc=new Scanner(System.in);
    int L=sc.nextInt();
    int R=sc.nextInt();
    System.out.println(calculate(L, R));
}
}
```

CodeSolutionin C

```c
#include<stdio.h>
#include<stdbool.h>

void printUnique(int l, int r)
{
    int count = 0;
    for (int i=l ; i<=r ; i++)
    {
        int num = i;
        bool visited[10] = {false};
        while (num)
        {
            if (visited[num % 10])
                break;

            visited[num%10] = true;

            num = num/10;
        }

        if (num == 0)
            count++;
    }
    printf("%d",count);
}

int main()
{
    int l,r;
    scanf("%d%d",&l,&r);
    printUnique(l, r);
    return 0;
}
```

2.Problemstatement:

GivenanarrayArr[]of NintegersandapositiveintegerK. The task istocyclicallyrotatethearray clockwise by K.

Note:Keepthefirstpositionofthearray unaltered.

Example	Input	Output	Explanation
Example1	5--ValueofN {10,20,30,40,50}--Elementsof Arr[] 2--ValueofK	4050102030	Arr[]={10,20,30,40, 50}andK =2 (Two cyclical rotations) After1strotation={10, 50, 20, 30, 40} After2ndrotation={10, 40, 50, 20, 30}
Example2	4--ValueofN {10,20,30,40}--ElementsofArr[] 1 -- Value of K	40102030	Arr[]={10,20,30,40} andK=1(Onecyclical rotation) After1strotation={10, 40, 20, 30}

Constraints

- 1<N<=100
- -100<=Arr[i]<=100
- 1<=K <=100

Inputformatfortesting

- Thecandidateshouldwritethecodetoaccept theinputs separated byanewline.
- FirstInput:Accepta singlepositiveinteger valueforNrepresenting thesizeof Arr[]
- SecondInput:AcceptNnumberofintegervaluesseparatedbyanewline,aselementsofArr[ ]
- Thirdinput: AcceptasinglepositiveintegervalueforKrepresentingthenumber ofrotations.

Outputformatfortesting

- TheoutputmustbeNintegernumbers separatedbyasinglespacecharacter.
- Additionalmessagesintheoutputwillresultinthefailureoftestcases.

Instructions

- Thesystemdoesnotallowanykindofhard-codedinputvalue/ values.

- The written program code by the candidate will be verified against the input which are supplied from the system.

CodeSolutioninJava

```java
import java.util.*;
public class Main
{
public static void Rotateby(int arr[], int n)
{
      int x = arr[n - 1], i;
    for (i = n - 1; i > 0; i--)
        arr[i] = arr[i - 1];
    arr[0] = x;
}
public static void Rotate(int arr[], int d, int n)
{
    for (int i = 0; i < d; i++)
        Rotateby(arr, n);
}
public static void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        System.out.printf("%d ",arr[i]);
}
public static void main(String args[])
{
    Scanner sc=new Scanner(System.in);
    int i;
    int n=sc.nextInt();
    int arr[]=new int[n];
    for(i = 0; i < n;i++)
        arr[i]=sc.nextInt();
    int k=sc.nextInt();
    Rotate(arr, k, n);
    printArray(arr, n);
}
}
```

CodeSolutionC

```c
#include<stdio.h>
void Rotateby(int arr[], int n)
{
      int x = arr[n - 1], i;
    for (i = n - 1; i > 0; i--)
        arr[i] = arr[i - 1];
    arr[0] = x;
}
void Rotate(int arr[], int d, int n)
{
    for (int i = 0; i < d; i++)
        Rotateby(arr, n);
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ",arr[i]);
}
int main()
{
    int arr[10],i;
    int n,k;
    scanf("%d",&n);
    for(i = 0; i < n;i++)
        scanf("%d",&arr[i]);
    scanf("%d",&k);
    Rotate(arr, k, n);
    printArray(arr, n);
    return 0;
}
```

3. Given an array Arr[] of N integer numbers. The task is to rewrite the array by putting all multiples of 10at the end of the given array.

Note: The order of the numbers which are not multiples of 10 should remain unaltered, and similarly, theorder of all multiples of 10 should be unaltered.

For e.g.

Suppose N = 9 and Arr[]={10, 12, 5, 40, 30, 7, 50, 9, 10}

You have to push all multiple of 10 at the end of the

Arr[]Hence, the output is 12 5 7 9 10 40 30 50 10.

Input	Output
9 .... Value of N 10 12 5 40 30 7 50 9 10 ... Elements of Arr[]	12 5 7 9 10 40 30 50 10
9 ..... Value of N 100 21 5 6 3 7 11 89 10.... Elements of Arr[]	21 5 6 3 7 11 89 100 10

Constraints:

1 < N < = 100

.100 < = Arr[i] < = 100

Input Format for Testing:

1. First input line: Accept a single positive integer value for N representing the size of Arr[].
2. Second Input line: Accept N number of integer values separated by a new line.

Output Format for Testing:

1. The output must be N integer numbers separated by a single space character (See the outputformat in examples).
2. Additional messages in the output will result in the failure of test cases.

Code Solution in Java

```
import java.util.*;
class Main
{
public static void main (String[] args) {
    Scanner sc=new Scanner (System.in);
    int n=sc.nextInt();
    int arr[]=new int[n];
    int i;
    for(i=0;i<n;i++)
        arr[i]=sc.nextInt();
    for(i=0;i<n;i++)
       if(arr[i]%10!=0)
        System.out.printf("%d ",arr[i]);
    for(i=0;i<n;i++)
       if(arr[i]%10==0)
        System.out.printf("%d ",arr[i]);
    }
}
```

Code Solution in C

```c
#include <stdio.h>
int main()
{
    int n,i,a[10];
    scanf("%d",&n);
    for(i = 0; i < n;i++)
    {
      scanf("%d",&a[i]);
    }
    for(i = 0; i < n;i++)
    {
     if(a[i] %10 !=0)
     {
       printf("%d ",a[i]);
     }
    }
    for(i = 0; i < n;i++)
    {
     if(a[i] %10 ==0)
     {
       printf("%d ",a[i]);
     }
    }
    return 0;
}
```

4. Given an array Arr[N] of N integers and a positive integer K. The task is to divide the array into twosub-arrays from right after the Kth position and slide the left sub-array of K elements to the end.

Input	Output	Explanation
5 -- Value of N {10, 20, 30, 40, 50} -- Elements of Arr [] 2 -- Value of K	30 40 50 10 20	Arr[] = {10,20,30,40,50} and K=2 (2nd position) Divide array from after 2nd position and add left sub-array {10,20} to the end. So the output is 30 40 50 10 20
4 -- Value of N {10, 20, 30, 40} -- Elements ofArr [] 1 -- Value of K	20 30 40 10	Arr[] = {10, 20, 30, 40} and K=1 (1st position) Divide array from after 1st position and add left sub-array {10} to the end. So the output is 20 30 40 10
4 -- Value of N {10, 20, 30, 40} -- Elements ofArr[] 3 -- Value of K	40 10 20 30	Arr[] = {10, 20, 30, 40} and K=3 (3rd position) Divide array from after 3rd position and add left sub-array {10, 20, 30} to the end. So the output is 40 10 20 30

Constraints

- 1<N<=100
- -100<=Arr[i]<=100
- 1<=K<N

Code Solution in Java

```java
import java.util.*;
public class Main
{
public static void Rotateby(int arr[], int n)
{
      int x = arr[n - 1], i;
    for (i = n - 1; i > 0; i--)
        arr[i] = arr[i - 1];
    arr[0] = x;
}
public static void Rotate(int arr[], int d, int n)
{
    for (int i = 0; i < d; i++)
        Rotateby(arr, n);
}
public static void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        System.out.printf("%d ",arr[i]);
}
public static void main(String args[])
{
    Scanner sc=new Scanner(System.in);
    int i;
    int n=sc.nextInt();
    int arr[]=new int[n];
    for(i = 0; i < n;i++)
        arr[i]=sc.nextInt();
    int k=sc.nextInt();
    Rotate(arr, k, n);
    printArray(arr, n);
}
}
```

Code Solution in C

```c
#include<stdio.h>
void Rotateby(int arr[], int n)
{
      int x = arr[n - 1], i;
    for (i = n - 1; i > 0; i--)
        arr[i] = arr[i - 1];
    arr[0] = x;
}
void Rotate(int arr[], int d, int n)
{
    for (int i = 0; i < d; i++)
        Rotateby(arr, n);
}
void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++)
        printf("%d ",arr[i]);
}
int main()
{
    int arr[10],i;
    int n,k;
    scanf("%d",&n);
    for(i = 0; i < n;i++)
        scanf("%d",&arr[i]);
    scanf("%d",&k);
    Rotate(arr, k, n);
    printArray(arr, n);
    return 0;
}
```

5. For hiring a car, a travel agency charges R1 rupees per hour for the first N hours and then R2 rupees per hour. Given the total time of travel in minutes is X. The task is to find the total traveling cost in rupees.

Note: While converting minutes into hours, ceiling value should be considered as the total number ofhours.

For example: If the total travelling time is 90 minutes,

i.e. 1.5 hours, it must be considered as 2 hours.

Input	Outp ut	EXplanation

20 ---Value of R1 4 --- Value of N in hours40 --- Value of R2 300 --- Value of X in minutes	120	Total travelling hours = 300/60 =5 hours Rupees 20/hours for first 4 hours = 20 * 4 = 80 rupees Rupees 40/hours in 5th hour = 40 * 1 = 40 rupees Hence, the total travelling cost =80 + 40 = 120 rupees
30 --- Value of R1 5 --- Value of N in hours. 35 --- Value of R2 500 -- Value of X in minutes	290	Total travelling hours = 500/60 =8.33, Ceiling value of 8.33 = 9 hours Rupees 30/hours for first 5thhours = 30 * 5 = 150 rupees Rupees 35/hours in 5th hour = 35 * 4 = 140 rupees Hence, the total travelling cost =150 + 140 = 290 rupees
30--- Value of R1 10--- Value of N in hours35 alue of R2 5--- Value of X in minutes	30	Total travelling hours = 3/60 = 0.05, Ceiling value of 0.05 = 1hour Rupees 30/hour for first 10 hours = 30 * 1 = 30 rupees

Constraints:

$1 < R1 < R32 < 100$

1 < = N < = 10

1 < = X < 10000

Code Solution in Java

```
import java.util.*;
class Main
{
public static void main(String args[])
{
    Scanner sc=new Scanner(System.in);
    int r1=sc.nextInt();
    int n=sc.nextInt();
    int r2=sc.nextInt();
    int k=sc.nextInt();
    int focus,hr;
    hr = (k+59)/60;
    if(hr > n)
        focus = n*r1+(hr-n)*r2;
    else
        focus = n*r1;
    System.out.println(focus);
}
}
```

Code Solution in C

```c
#include<stdio.h>
int main()
{
  int r1,n,r2,k,focus,hr;
  scanf("%d%d%d%d",&r1,&n,&r2,&k);
  hr = (k+59)/60;
  if(hr > n)
  {
    focus = n*r1+(hr-n)*r2;
  }
  else
  {
    focus = n*r1;
  }
  printf("%d",focus);
  return 0;
}
```

6. There is a bag with three types of gemstones: Ruby of type R, Garnet of type g, and Topaz of type T. Write a program to find the total number of possible arrangements to make a series of gemstones whereno two gemstones of the same type are adjacent to each other.

Input	Output	Explanation
1-Count of R i.e. Ruby 1-Count of G i.e. Garnet 0-Count of T i.e.	2	Arrangements are RG and GR.
1-Count of R i.e. Ruby 1-Count of G i.e .Garnet1-Count of T i.e. Topaz	6	Arrangements are RGTR, GRTR,RGRT, RTGR, RTRG AND TRGR

Code Solution in C

```c
#include<stdio.h>

int countWays(int p, int q, int r, int last)
{
    if (p<0 || q<0 || r<0)
        return 0;
    if(p == 1 && q == 0 && r== 0 && last == 0)
        return 1;
    if (p==0 && q==1 && r==0 && last==1)
        return 1;
    if (p==0 && q==0 && r==1 && last==2)
        return 1;

    if (last==0)
        return countWays(p-1,q,r,1) + countWays(p-1,q,r,2);
    if (last==1)
        return countWays(p,q-1,r,0) + countWays(p,q-1,r,2);
    if (last==2)
        return countWays(p,q,r-1,0) + countWays(p,q,r-1,1);
}

int faceprep(int p, int q, int r)
{
return countWays(p, q, r, 0) + countWays(p, q, r, 1) + countWays(p, q, r, 2);
}
int main()
{
    int p,q,r;
    scanf("%d%d%d",&p,&q,&r);
    printf("%d", faceprep(p, q, r));
    return 0;
}
```

Code Solution in Java

```java
import java.util.*;
class Main{
    static int countWays(int p, int q, int r, int last)
    {
        if (p < 0 || q < 0 || r < 0)
        return 0;
        if (p == 1 && q == 0 && r == 0 && last == 0)
            return 1;
        if (p == 0 && q == 1 && r == 0 && last == 1)
            return 1;
        if (p == 0 && q == 0 && r == 1 && last == 2)
            return 1;
        if (last == 0)
            return countWays(p - 1, q, r, 1) +
            countWays(p - 1, q, r, 2);
        if (last == 1)
            return countWays(p, q - 1, r, 0) +
                countWays(p, q - 1, r, 2);
        if(last == 2)
            return countWays(p, q, r - 1, 0) +
                countWays(p, q, r - 1, 1);
        return 0;
    }
    static int faceprep(int p, int q, int r) {
        return countWays(p, q, r, 0) +
            countWays(p, q, r, 1) +
            countWays(p, q, r, 2);
    }

    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        int p=sc.nextInt();
        int q=sc.nextInt();
        int r=sc.nextInt();
        System.out.print(faceprep(p, q, r));
    }
}
```

7.Problem Description:

You are given a sorted and infinite array A[] and an element K. You need to search for the element K in the array. If found, return the index of the element, else return -1.

For Example :

Input: A[] = {1,3,5,8,12,13,17,19,28,39,103,123,140,2040,...}, K = 17

Output: 6

Input: A[] = {10,20,25,30,67,93,159,192,350,1230,1341,4533,...}, K = 23

Output: -1

Ksible questions to ask the interviewer:-

- **What is the meaning of infinite array ? (Ans : We don't know the upper bound of the array)**
- **How big can the resultant index be? (Ans: Ignore the integer overflow problem, we just want to check your logic, let us assume integer overflow won't occur)**

Brute force and Efficient solutions
We will be discussing three solutions for this problem:-

1. **Brute Force approach : Using linear Search**
2. **Increment by a constant value and using binary search**

1. Brute Force approach: Using linear Search
In the infinite array, we don't know the upper bound to apply the binary search. So simple solution we could think to start searching for K linearly from index 0 until you you find an element equal to K, then return the index. If you find an element greater than K, then return -1.

Pseudo-Code

```
int search_infiniteArray(int A[], int K)
{
  int i = 0
  while (A[i] <= K)
  {
    if(A[i] == K)
      return i
    else
      i = i + 1
  }
```

```
  return -1
}
```

Complexity Analysis

Let i be the position of the element to be searched, then the time Complexity = O(i) (Think)

Space Complexity: O(1)

Critical ideas to think!

- Since we know that the array is sorted, Can we use this info to improve the time complexity?

2. Increment by a constant value and using binary search

If we can track the interval (with the lower and upper bound) where target value reside then we can apply the binary search in that interval. Here we maintain the interval size by constant value C.

Note: In a sorted array, if we check an element at any index j, we could logically know the relative position of element K with respect to A[j].

Solution Steps

1. Initialize lower and upper index of the interval i.e. l = 0 , r = C
2. Compare the K with the value present at the upper index of the interval

- if K > A[r] then copy the upper index in the lower index and increase the upper index by C. Keep on doing this until you reach a value that is greater than K.
- If K < A[r], apply binary search in the interval from l to r

3. If found, return the index else return -1.

Pseudo-Code

```
int search_infiniteArray( int A[], int K )
{
  int l = 0
  int r = C
  while (A[r] < K)
  {
    l = r
    r = r + C
  }
```

```
  return binarySearch(A, l, r, K)
}
```

8. Write a function to remove all duplicate characters from a given string.
Note: The duplicate elements are to be removed in such a way that when reading the string from left to right, the repeated element which occurs later should be removed.
Input Format
Input contains a string.
Output Format
Return a string with non-duplicate
Return a string with non-duplicate characters,
i.e if you have a string as mettl then output should be metl,

Code

```
import java.util.*;
class Main
 {
static String removedup(String s)
{
String str="";
for(int i=0;i<s.length();i++)
{
if(str.indexOf(s.charAt(i))==-1)
str+=s.charAt(i);
}
return str;
}
public static void main(String[] args)
{
Scanner sc=new Scanner(System.in);
String s=sc.nextLine();
System.out.print(removedup(s));
}
}
```

9. Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M.

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II.

Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

I can be placed before V (5) and X (10) to make 4 and 9.
X can be placed before L (50) and C (100) to make 40 and 90.
C can be placed before D (500) and M (1000) to make 400 and 900.
Given a roman numeral, convert it to an integer.

Example 1:
Input: s = "III"
Output: 3
Explanation: III = 3.
Example 2:
Input: s = "LVIII"
Output: 58
Explanation: L = 50, V= 5, III = 3.
Example 3:
Input: s = "MCMXCIV"
Output: 1994
Explanation: M = 1000, CM = 900, XC = 90 and IV = 4.

10.Problem Description -: In this 3 Palindrome, Given an input string word, split the string into exactly 3 palindromic substrings. Working from left to right, choose the smallest split for the first substring that still allows the remaining word to be split into 2 palindromes.

Similarly, choose the smallest second palindromic substring that leaves a third palindromic substring.

If there is no way to split the word into exactly three palindromic substrings, print "Impossible" (without quotes). Every character of the string needs to be consumed.

Cases not allowed –

- After finding 3 palindromes using above instructions, if any character of the original string remains unconsumed.
- No character may be shared in forming 3 palindromes.

Constraints

- 1 <= the length of input sting <= 1000

Input

- First line contains the input string consisting of characters between [a-z].

Output

- Print 3 substrings one on each line.

Time Limit

1

Examples

Example 1

Input

nayannamantenet

Output

nayan

naman

tenet

Explanation

- The original string can be split into 3 palindromes as mentioned in the output.
- However, if the input was nayanamantenet, then the answer would be “Impossible”.

Example 2

Input

aaaaa

Output

a

a

aaa

Explanation

- The other ways to split the given string into 3 palindromes are as follows –
- [a, aaa, a], [aaa, a, a], [aa, aa, a], etc.
- Since we want to minimize the length of the first palindromic substring using left to right processing, the correct way to split is [a, a, aaa].

CPP CODE

```
#include<bits/stdc++.h>
typedef long long int lld;
#define mod 1000000007
using namespace std;
bool pali(string s)
{
   if(s.length()==1) return true;
   string s1=s;reverse(s1.begin(),s1.end());
   return (s1==s);
}
int main()
{
   speed;
   string s,s1,s2,s3;
   cin>>s;
   int l=s.length();
```

```
  for(int i=1;i<l-1;i++)
  {
     s1=s.substr(0,i);
     if(pali(s1))
        for(int j=1;j<l-i;j++)
     {
        s2=s.substr(i,j);s3=s.substr(i+j,l-i-j);
        if(pali(s2)&&pali(s3))
        {
           cout<<s1<<endl<<s2<<endl<<s3;return 0;
        }
     }
  }
  cout<<"Impossible";
  return 0;
}
```

11.Problem Statement -: In this even odd problem Given a range [low, high] (both inclusive), select K numbers from the range (a number can be chosen multiple times) such that sum of those K numbers is even.

Calculate the number of all such permutations.

As this number can be large, print it modulo (1e9 +7).

Constraints

- 0 <= low <= high <= 10^9
- K <= 10^6.

Input

- First line contains two space separated integers denoting low and high respectively
- Second line contains a single integer K.

Output

- Print a single integer denoting the number of all such permutations

Time Limit

1

Examples

Example 1

Input

4 5

3

Output

4

Explanation

There are 4 valid permutations viz. {4, 4, 4}, {4, 5, 5}, {5, 4, 5} and {5, 5, 4} which sum up to an even number.

Example 2

Input

1 10

2

Output

50

Explanation

There are 50 valid permutations viz. {1,1}, {1, 3},.. {1, 9} {2,2}, {2, 4},... {2, 10} . . . {10, 2}, {10, 4},... {10, 10}. These 50 permutations, each sum up to an even number.

CPP CODE

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int lld;
#define mod 1000000007
long e_sum(long m,long n,long K,long N)
{
if(K==1)
{
return n;
}
else
{
```

```
return (N-(m-n)*e_sum(m,n,K-1,N)%(1000000007));
}
}
int main()
{
long low,high,K,m,n,diff,Out,N,i;
scanf("%ld",&low);
scanf("%ld",&high);
scanf("%ld",&K);
diff=high-low+1;
if(diff%2==0)
{
m=diff/2;
n=m;
}
else
{
if(low%2==0)
{
m=(diff-1)/2;
n=m+1;
}
else
{
m=(diff+1)/2;
n=m-1;
}
}
N=m;
for(i=0;i<K-1;i++)
{
N=(N*(m+n))%1000000007;
}
Out=e_sum(m,n,K,N)%1000000007;
printf("%ld",Out);
return 0;
}
```

12.Roco is an island near Africa which is very prone to forest fire. Forest fire is such that it destroys the complete forest. Not a single tree is left.This island has been cursed by God , and the curse is that whenever a tree catches fire, it passes the fire to all its adjacent tree in all 8 directions,North, South, East, West, North-East, North-West, South-East, and South-West.And it is given that the fire is spreading every minute in the given manner, i.e every

tree is passing fire to its adjacent tree.Suppose that the forest layout is as follows where T denotes tree and W denotes water.

Your task is that given the location of the first tree that catches fire, determine how long would it take for the entire forest to be on fire. You may assume that the lay out of the forest is such that the whole forest will catch fire for sure and that there will be at least one tree in the forest

Input Format:

- First line contains two integers, M, N, space separated, giving the size of the forest in terms of the number of rows and columns respectively.
- The next line contains two integers X,Y, space separated, giving the coordinates of the first tree that catches the fire.
- The next M lines, where ith line containing N characters each of which is either T or W, giving the position of the Tree and Water in the ith row of the forest.

Output Format:

Single integer indicating the number of minutes taken for the entire forest to catch fire

Constrains:

- $3 \leq M \leq 20$
- $3 \leq N \leq 20$

Sample Input 1:

3 3
W T T
T W W
W T T
Sample Output 1:

5

Explanation:
In the second minute,tree at (1,2) catches fire,in the third minute,the tree at (2,1) catches fire,fourth minute tree at (3,2) catches fire and in the fifth minute the last tree at (3,3) catches fire.
Sample Input 2:
6 6
1 6
W T T T T T

T W W W W W
W T T T T T
W W W W W T
T T T T T T
T W W W W W

Sample Output 2:

16

CPP CODE

```
#include <bits/stdc++.h>
using namespace std;
char f[21][21];
int n,m;
struct node{int a,b;};
bool valid(int x,int y) {return (x>=0&&y>=0&&x<n&&y<m);}
bool step(node temp){return (temp.a==-1&&temp.b==-1);}
int main()
{
cin>>n>>m;
int x,y,i,j,count=0;int ans=1;
cin>>x>>y;x--;y--;
for(i=0;i<n;i++)
for(j=0;j<m;j++)
cin>>f[i][j];
f[x][y]='X';

queueq;
node temp;
temp.a=x;temp.b=y;
q.push(temp);
temp.a=-1;temp.b=-1;
q.push(temp);

while(!q.empty())
{
bool flag=false;
while(!step(q.front()))
{
node count=q.front();
if(valid(count.a+1,count.b)&&f[count.a+1][count.b]=='T')//a+1,b
{
```

```
if(flag==false){flag=true;ans++;}
f[count.a+1][count.b]='X';
count.a++;
q.push(count);
count.a--;
}
if(valid(count.a+1,count.b+1)&&f[count.a+1][count.b+1]=='T')//a+1,b+1
{
if(flag==false){flag=true;ans++;}
f[count.a+1][count.b+1]='X';
count.a++;count.b++;
q.push(count);
count.a--;count.b--;
}
if(valid(count.a+1,count.b-1)&&f[count.a+1][count.b-1]=='T')//a+1,b-1
{
if(flag==false){flag=true;ans++;}
f[count.a+1][count.b-1]='X';
count.a++;count.b--;
q.push(count);
count.a--;count.b++;
}
if(valid(count.a,count.b+1)&&f[count.a][count.b+1]=='T')//a,b+1
{
if(flag==false){flag=true;ans++;}
f[count.a][count.b+1]='X';
count.b++;
q.push(count);
count.b--;
}
if(valid(count.a,count.b-1)&&f[count.a][count.b-1]=='T')//a,b-1
{
if(flag==false){flag=true;ans++;}
f[count.a][count.b-1]='X';
count.b--;
q.push(count);
count.b++;
}
if(valid(count.a-1,count.b-1)&&f[count.a-1][count.b-1]=='T')//a-1,b-1
{
if(flag==false){flag=true;ans++;}
f[count.a-1][count.b-1]='X';
count.a--;count.b--;
q.push(count);
```

```
count.a++;count.b++;
}
if(valid(count.a-1,count.b+1)&&f[count.a-1][count.b+1]=='T')//a-1,b+1
{
if(flag==false){flag=true;ans++;}
f[count.a-1][count.b+1]='X';
count.a--;count.b++;
q.push(count);
count.a++;count.b--;
}
if(valid(count.a-1,count.b)&&f[count.a-1][count.b]=='T')//a-1,b
{
if(flag==false){flag=true;ans++;}
f[count.a-1][count.b]='X';
count.a--;
q.push(count);
count.a++;
}
q.pop();
}
q.pop();
if(!q.empty())
{
temp.a=-1;
temp.b=-1;
q.push(temp);
}

/*
cout<<endl;
for(i=0;i<n;i++)
{for(j=0;j<m;j++)
{cout<<f[i][j]<<" ";}cout<<endl;}
cout<<endl<<endl;
*/
}
cout<<ans;
}
```

13.Compute the nearest larger number by interchanging its digits updated.Given 2 numbers a and b find the smallest number greater than b by interchanging the digits of a and if not possible print -1.

- **Input Format**
  2 numbers a and b, separated by space.

- **Output Format**
  A single number greater than b.

  If not possible, print -1

- **Constraints**

  1 <= a,b <= 10000000

Example 1:

Sample Input:

459 500

Sample Output:
549

Example 2:

Sample Input:

645757 457765

Sample Output:
465577

CPP CODE

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
string a;
int b,c;
cin>>a>>b;
sort(a.begin(),a.end(),greater());

c=atoi(a.c_str());

if(b>c)
{cout<<-1;
```

```
return 0;}
while(b<c)
{
prev_permutation(a.begin(),a.end());
c=atoi(a.c_str());
}
next_permutation(a.begin(),a.end());
cout<<a;
}
```

14.Problem Statement:- In a Conference ,attendees are invited for a dinner after the conference.The Co-ordinator,Sagar arranged around round tables for dinner and want to have an impactful seating experience for the attendees.Before finalizing the seating arrangement,he wants to analyze all the possible arrangements.These are R round tables and N attendees.In case where N is an exact multiple of R,the number of attendees must be exactly N//R,,If N is not an exact multiple of R, then the distribution of attendees must be as equal as possible.Please refer to the example section before for better understanding.
For example, R = 2 and N = 3
All possible seating arrangements are
(1,2) & (3)
(1,3) & (2)
(2,3) & (1)
Attendees are numbered from 1 to N.

Input Format:

- The first line contains T denoting the number of test cases.
- Each test case contains two space separated integers R and N, Where R denotes the number of round tables and N denotes the number of attendees.

Output Format:

Single Integer S denoting the number of possible unique arrangements.

Constraints:

- 0 <= R <= 10(Integer)
- 0 < N <= 20 (Integer)

Sample Input 1:
1
2 5
Sample Output 1:

10

Explanation:

R = 2, N = 5

(1,2,3) & (4,5)

(1,2,4) & (3,5)

(1,2,5) & (3,4)

(1,3,4) & (2,5)

(1,3,5) & (2,4)

(1,4,5) & (2,3)

(2,3,4) & (1,5)

(2,3,5) & (1,4)

(2,4,5) & (1,3)

(3,4,5) & (1,2)

Arrangements like

(1,2,3) & (4,5)

(2,1,3) & (4,5)

(2,3,1) & (4,5) etc.

But as it is a round table,all the above arrangements are same.

CPP CODE

```
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;
map <ll,ll> dp;
int fac(int n)
{
if(dp[n])
return dp[n];
```

```
dp[n]=n*fac(n-1);
return dp[n];
}
int func(int n)
{
if(n<=3)
return 1;
return fac(n-1);
}

int main()
{
dp[0]=dp[1]=1;
int tests;cin>>tests;
while(tests--)
{int R,N,c=1;
cin>>R>>N;
if(N<=R)
{
cout<<1;
continue;
}
int a=N/R,n=N%R;
ll ans=fac(N)/(pow(fac(a),R-n) * pow(fac(a+1),n) )/fac(n)/fac(R-n);
for(int i=1;i<=n;i++)
c*=func(a);
for(int i=n+1;i<=R;i++)
c*=func(a-1);

cout<<c*ans;}
}
```

15. Pangram Checking

Given a string check if it is Pangram or not. A pangram is a sentence containing every letter in the English Alphabet.

Examples : The quick brown fox jumps over the lazy dog ” is a Pangram [Contains all the characters from ‘a’ to ‘z’]

“The quick brown fox jumps over the dog” is not a Pangram [Doesn’t contains all the characters from ‘a’ to ‘z’, as ‘l’, ‘z’, ‘y’ are missing]
We create a mark[] array of Boolean type. We iterate through all the characters of our string and whenever we see a character we mark it. Lowercase and Uppercase are considered the same. So ‘A’ and ‘a’ are marked in index 0 and similarly ‘Z’ and ‘z’ are marked in index 25. After iterating through all the characters we check whether all the characters are marked or not. If not then return false as this is not a pangram else return true.

Java CODE

```
import java.util.*;
public class digital5 {
public static void main(String ars[])
{
Scanner sc=new Scanner(System.in);
System.out.println(“Enter a string “);
String str=sc.nextLine();
String str1=str.toUpperCase();
boolean success = true;
for(char i=’A’;i<='Z';i++)
{
if(!str1.contains(String.valueOf(i)))
{
success=false;
break;
}
}
if(success)
System.out.println("Found all character");
else
System.out.println("Not found all character");
}



}
```