Ex. No. : **Date :**

Alice and Strings

Problem Statement:

Two strings A and B comprising of lower-case English letters are compatible if they are equal or can be made equal by following this step any number of times:

- Select a prefix from the string A (possibly empty), and increase the alphabetical value of all the characters in the prefix by the same valid amount. For example, if the string is xyz and we select the prefix xy then we can convert it to yx by increasing the alphabetical value by 1. But if we select the prefix xyz then we cannot increase the alphabetical value.

Your task is to determine if given strings A and B are compatible.

Input format
First line: String A
Next line: String B

Output format
For each test case, print YES if string A can be converted to string B, otherwise print NO.

Constraints
$1 \leq len(A) \leq 1000000$
$1 \leq len(B) \leq 1000000$

Sample Input
abaca
cdbda

Sample Output
YES

Explanation
The string abaca can be converted to bcbda in one move and to cdbda in the next move.

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{
   char str1[1000000], str2[1000000];
   int flag = 1;
   scanf("%s", str1);
   scanf("%s", str2);
   int a = strlen(str1);
   int b = strlen(str2);
   if(a==b)
   {  for(int i=a-1; i>=0; i--)
      {
         while(str1[i]!=str2[i])
         {
            for(int j=0; j<=i; j++)
            {  if(str1[j]<'z')
               str1[j]++;
               else
               {  flag=0;
                  break;
               }
```

```
            if(flag==0)
            break;
        }
    }
  }
}
else
flag=0;
if(flag==0)
printf("NO");
else
printf("YES");
return 0;
}
```

Ex. No. : **Date :**

Pizza Confusion

Problem Statement:

Joey loves to eat Pizza. But he is worried as the quality of pizza made by most of the restaurants is deteriorating. The last few pizzas ordered by him did not taste good :(. Joey is feeling extremely hungry and wants to eat pizza. But he is confused about the restaurant from wherehe should order. As always he asks Chandler for help.

Chandler suggests that Joey should give each restaurant some points, and then choose the restaurant having maximum points. If more than one restaurant has same points, Joey can choose the one with lexicographically smallest name.

Joey has assigned points to all the restaurants, but can't figure out which restaurant satisfies Chandler's criteria. Can you help him out?

Input Format:
First line has N, the total number of restaurants.
Next N lines contain Name of Restaurant and Points awarded by Joey, separated by a space.
Restaurant name has no spaces, all lowercase letters and will not be more than 20 characters.

Output Format:
Print the name of the restaurant that Joey should choose.

Constraints:
1 <= N <= 105
1 <= Points <= 106

Sample Input
3
Pizzeria 108
Dominos 145
Pizzapizza 49

Sample Output
Dominos

Program:

```
#include <stdio.h>
#include <string.h>
int main()
{
    int n;
    scanf("%d", &n);
    char res[n][21];
    int rate[n];
    for (int i=0; i<n; i++)
    {
        scanf("%s", res[i]);
        scanf("%d", &rate[i]);
    }
    int max = rate[0];
    char ans[20];
    strcpy(ans, res[0]);
    for (int i=1; i<n; i++)
    {
        if (rate[i] > max)
        {
            max = rate[i]
```

```
        strcpy (ans , res[i]);
    }
    else if (rate[i] == max)
    {
        if(strcmp(res[i], ans)<0)
        strcpy (ans, res[i]);
    }
}
printf(" %s", ans);
return 0; }
```

Ex. No. : **Date :**

Password

Problem Statement:

Danny has a possible list of passwords of Manny's facebook account. All passwords length is odd. But Danny knows that Manny is a big fan of palindromes. So, his password and reverse of his password both should be in the list.

You have to print the length of Manny's password and it's middle character.

Note: The solution will be unique.

Input Format
The first line of input contains the integer N, the number of possible passwords.
Each of the following N lines contains a single word, its length being an odd number greater than 2 and lesser than 14. All characters are lowercase letters of the English alphabet.

Output Format
The first and only line of output must contain the length of the correct password and its central letter.

Constraints
$1 \leq N \leq 100$

Sample Input
4
abc
def
feg
cba

Sample Output
3 b

Program:

```
#include<stdio.h>
#include<string.h>
int main()
{ int n,flag=0;
  char temp;
  scanf("%d",&n);
  char words[n][14];
  for(int i=0;i<n;i++)
  scanf("%s",words[i]);
  char reverse[14];
  for(int i=0;i<n-1;i++)
  {
    strcpy(reverse,words[i]);
    int size=strlen(reverse);
   for(int k=0;k<size/2;k++)
   {
     temp=revesse[k];
     reverse[k]=reverse[size-k-1];
     reverse[size-k-1]=temp;
   }
  for(int j=i+1;j<n;j++)
```

```
    {
        if(strcmp(reverse,words[j])==0)
        {
            flag=1;
            break;
        }
    }
    if(flag==1)
    break;
}
int len=strlen(reverse);
printf("%d %c",len,reverse(len/2));
return 0;
}
```

Ex. No. : **Date :**

Find the Factor

Problem Statement:

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number) and then return the pth element of the list, sorted ascending. If there is no pth element, return 0.

Example
n = 20
p = 3
The factors of 20 in ascending order are {1, 2, 4, 5, 10, 20}. Using 1-based indexing, if p = 3, then 4 is returned. If p > 6, 0 would be returned.

Function Description
Complete the function pthFactor in the editor below.

pthFactor has the following parameter(s):
int n: the integer whose factors are to be found
int p: the index of the factor to be returned

Returns:
int: the long integer value of the pth integer factor of n or, if there is no factor at that index, then 0 is returned

Constraints
$1 \leq n \leq 1015$
$1 \leq p \leq 109$

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The first line contains an integer n, the number to factor.
The second line contains an integer p, the 1-based index of the factor to return.

Sample Input

STDIN		Function
10	→	n = 10
3	→	p = 3

Sample Output
5

Explanation
Factoring n = 10 results in {1, 2, 5, 10}. Return the p = 3rd factor, 5, as the answer.

Program:

```
# include<stdio.h>
long pthFactor(long n,long p)
{
    int count=0;
    for(long i=1; i<=n; ++i)
    {
        if(n%. i==0)
        {
            count++;
            if(count==p)
            {
                return i;
            }
        }
    }
    return 0;
}
```

Ex. No. : **Date :**

Prime or Not?

Problem Statement:

Given an integer, if the number is prime, return 1. Otherwise return its smallest divisor greater than 1.

Example
n = 24
The number 24 is not prime: its divisors are [1, 2, 3, 4, 6, 8, 12, 24]. The smallest divisor greater than 1 is 2.

Function Description
Complete the function isPrime in the editor below.

isPrime has the following parameter(s):
long n: a long integer to test

Returns
int: if the number is prime, return 1; otherwise returns the smallest divisor greater than 1

Constraints
$2 \leq n \leq 1012$

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The only line of input contains the long integer to analyze, n.

Sample Input

STDIN		Function
2	→	n = 2

Sample Output
1

Explanation
As 2 is a prime number, the function returns 1.

Program:

```
#include<stdio.h>
#include<math.h>
long is prime (long n)
    {
      if (n<=1)
      return 0;
      for (long i= 2 ; i<= sqrt (n) ; i++)
      {
        if (n%. i==0) {
          return 0;
        }
      }
        return 1;
      }
    int main ()
    {
      long n ;
      scanf("%lld", &n);
      long result = is prime (n);
      if (result==1) {
        printf("\n"); }
      else {
        printf("%d", result); }
      return 0; }
```

Ex. No. : **Date :**

4th Bit

Problem Statement:

A binary number is a combination of 1s and 0s. Its nth least significant digit is the nth digit starting from the right starting with 1. Given a decimal number, convert it to binary and determine the value of the the 4th least significant digit.

Example
number = 23

- Convert the decimal number 23 to binary number: $23^{10} = 2^4 + 2^2 + 2^1 + 2^0 = (10111)_2$.
- The value of the 4th index from the right in the binary representation is 0.

Function Description
Complete the function fourthBit in the editor below.

fourthBit has the following parameter(s):
int number: a decimal integer

Returns:
int: an integer 0 or 1 matching the 4th least significant digit in the binary representation of number.

Constraints
$0 \leq number < 231$

Input Format for Custom Testing
Input from stdin will be processed as follows and passed to the function.
The only line contains an integer, number.

Sample Input

STDIN		Function
---		---
32	→	number = 32

Sample Output
0

Explanation

- Convert the decimal number 32 to binary number: 3210 = (100000)2.
- The value of the 4th index from the right in the binary representation is 0.

Program:

```
# include<stdio.h>
int fourthBit(int number)
 {
     int binary[32];
     int i=0;
     while(number>0)
      {
         binary[i]=number%2;
         number/=2;
          i++;
       }
  if(i>=4)
     {
        return binary[3];
     }
     else
     return 0;
 }
```

Ex. No. : **Date :**

The Power Sum

Problem Statement:

Find the number of ways that a given integer, X, can be expressed as the sum of the Nth powers of unique, natural numbers.
For example, if X = 13 and N = 2, we have to find all combinations of unique squares adding up to 13. The only solution is $2^2 + 3^2$.

Function Description

Complete the powerSum function in the editor below. It should return an integer that represents the number of possible combinations.

powerSum has the following parameter(s):

X: the integer to sum to
N: the integer power to raise numbers to

Input Format
The first line contains an integer X.
The second line contains an integer N.

Constraints
$1 \leq X \leq 1000$
$2 \leq N \leq 10$

Output Format
Output a single integer, the number of possible combinations calculated.

Sample Input
10
2

Sample Output
1

Explanation
If X = 10 and N = 2, we need to find the number of ways that 10 can be represented as the sum of squares of unique numbers.

$10 = 1^2 + 3^2$

This is the only way in which 10 can be expressed as the sum of unique squares.

Program:

```
# include <stdio.h>
int powerSum (int x, int m, int n)
{
    int p= Pow(.
    if (x==0) {
        return 1;
    }
    if (x< 0) {
        return 0;
    }
    int count =0;
    for( int i= m;  ;i++) {
        int power =1;
        for(int j=0; j<n ; j++) {
            power *= i;
        }
        if (power > x) {
            break;
        }
        count += powerSum (x-power,
                           i+1, n);
    }
    return count;
```

Ex. No. : **Date :**

Hack the Money

Problem Statement:

You are a bank account hacker. Initially you have 1 rupee in your account, and you want exactly N rupees in your account. You wrote two hacks, first hack can multiply the amount of money you own by 10, while the second can multiply it by 20. These hacks can be used any number of time. Can you achieve the desired amount N using these hacks.

Constraints:

1<=T<=100
1<=N<=10^12

Input

- The test case contains a single integer N.

Output
For each test case, print a single line containing the string "1" if you can make exactly N rupees or "0" otherwise.

SAMPLE INPUT
1

SAMPLE OUTPUT
1

SAMPLE INPUT
2

SAMPLE OUTPUT
0

Program:

```
int myfunc(int n) {
    while (n>1) {
        if (n==200) {
            n=1;
        }
        else if (n%10==0) {
            n/=10;
        }
        else if (n%20==0) {
            n/=20;
        }
        else {
            return 0;
        }
    }
    return (n==0)? 0:1;
}
```

Ex. No. : **Date :**

Number of Divisors

Problem Statement:

You are given two numbers n and k. For each number in the interval [1, n], your task is to calculate its largest divisor that is not divisible by k. Print the sum of these divisors.

Note: k is a prime number.

Input format

- Each test case consists of one line containing two integers n and k.

Output format
The output must contain the answer for each test case on a different line.
Each answer consists of a single integer.

Constraints
$1 \leq n \leq 1000000000$
$2 \leq k \leq 1000000000$

SAMPLE INPUT
10 3

SAMPLE OUTPUT
41

SAMPLE INPUT
10 2

SAMPLE OUTPUT
36

Explanation
In the first test case, f (x) from 1 to 10 is [1, 2, 1, 4, 5, 2, 7, 8, 1, 10], sum of which is 41.
In the second test case, f (x) from 1 to 10 is [1, 1, 3, 1, 5, 3, 7, 1, 9, 5].

Program:

```
#include <stdio.h>
long long largestDivisorNodivisibleByk
    (long long x, long long k) {
        while (x % k == 0) {
            x /= k; }
        return x;
    }
long long compatSum(long long n, long long k);
    long long sum = 0;
    for(long long i=1; i<=n; i++) {
        sum += largestDivisorNdivisibleBykFor(i,k)
    }
    return sum; }
}
int main() {
    long long n, k;
    if(scanf("%lld %lld", &n, &k) != 2) {
        if printf(stderr, "Invalid Input");
        }
    return;
}
```

Ex. No. : **Date :**

Recursive Digit Sum

Problem Statement:

We define super digit of an integer x using the following rules:
Given an integer, we need to find the super digit of the integer.

- If x has only 1 digit, then its super digit is x.
- Otherwise, the super digit of x is equal to the super digit of the sum of the digits of x.

For example, the super digit of 9875 will be calculated as:
super_digit(9875) 9+8+7+5 = 29, super_digit(29) 2 + 9 = 11, super_digit(11) 1 + 1 = 2
super_digit(2) = 2

You are given two numbers n and k. The number p is created by concatenating the string n k times. Continuing the above example where n = 9875, assume your value k = 4. Your initial p = 9875 9875 9875 9875 (spaces added for clarity).
superDigit(9875987598759875)5+7+8+9+5+7+8+9+5+7+8+9+5+7+8+9 = 116
superDigit(116)1+1+6 = 8, superDigit(8)=8
All of the digits of p sum to 116. The digits of 116 sum to 8. 8 is only one digit, so it's the super digit.

Function Description
Complete the function superDigit in the editor below. It must return the calculated super digit as an integer.
superDigit has the following parameter(s): n: a string representation of an integer, k: an integer, the times to concatenate n to make p

Input Format
The first line contains two space separated integers, n and k.
Constraints: $1 \leq n \leq 10100000$, $1 \leq k \leq 105$
Output Format
Return the super digit of ***p***, where ***p*** is created as described above.

Sample Input
148 3
Sample Output
3

Explanation
Here ***n = 148*** and ***k = 3***, so ***p = 148148148.***
super_digit(P)= super_digit(148148148)
= super_digit(1+4+8+1+4+8+1+4+8)
= super_digit(39)
= super_digit(3+9)
= super_digit(12)
= super_digit(1+2)
= super_digit(3)
= 3.

Program:

```
#include<stdio.h>
#include<string.h>
int superdigit(const char * n, int k) {
        long long sum=0;
        for(int i=0; n[i] != '\0'; i++) {
            sum += n[i]-'0';
        }
        sum = k;
        while (sum >=10) {
            long long temp =0;
            while (sum >0){
                temp += sum % 10;
                sum =10;
            }
            sum = temp;
        }
        return (int )(sum);
}
int main() {
    char n[100001];
    int k;
    scanf("%s %d", n, &k);
    printf("%d\n", superdigit(n, n));
    return 0;
}
```