Algorithm & Flowchart

Scanned with OKEN Scanner

Ex. No.: 1 Date:

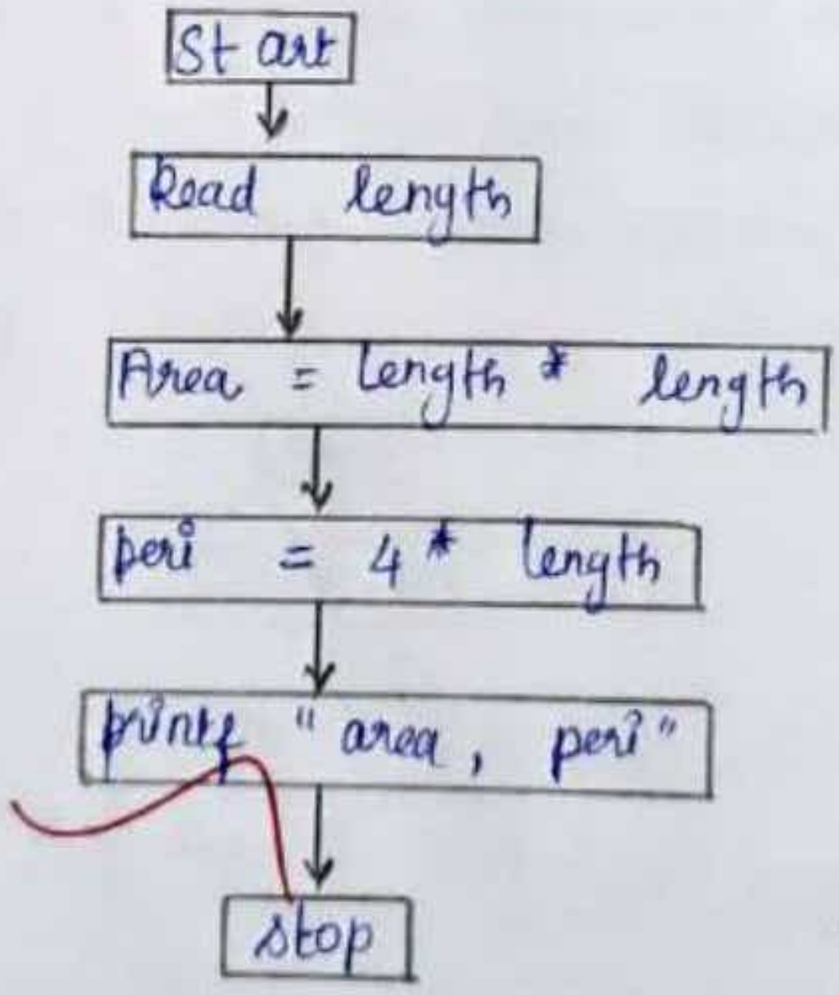
Calculate Area and Perimeter

Write an Algorithm and draw a Flowchart to Calculate the area and perimeter of a square.

Algorithm:

Step 1: Start
Step 2: Read length
Step 3: Calculate
Area = length * length
Step 4: Calculate
peri = 4 * length
Step 5: print " area , peri "
Step 6: Stop.

Flowchart:



Scanned with OKEN Scanner

Ex. No.: 2 **Date:**

Days to Year Conversion

Write an Algorithm and draw a Flowchart to convert the given days into years & months.

Algorithm:

Step 1 : Start

Step 2 : [Input the number of days]
Input total days.

step 3 : [compute years]
Years = Total days div 365

step 4 : [compute remaining days]
Rem = Total days mod 365

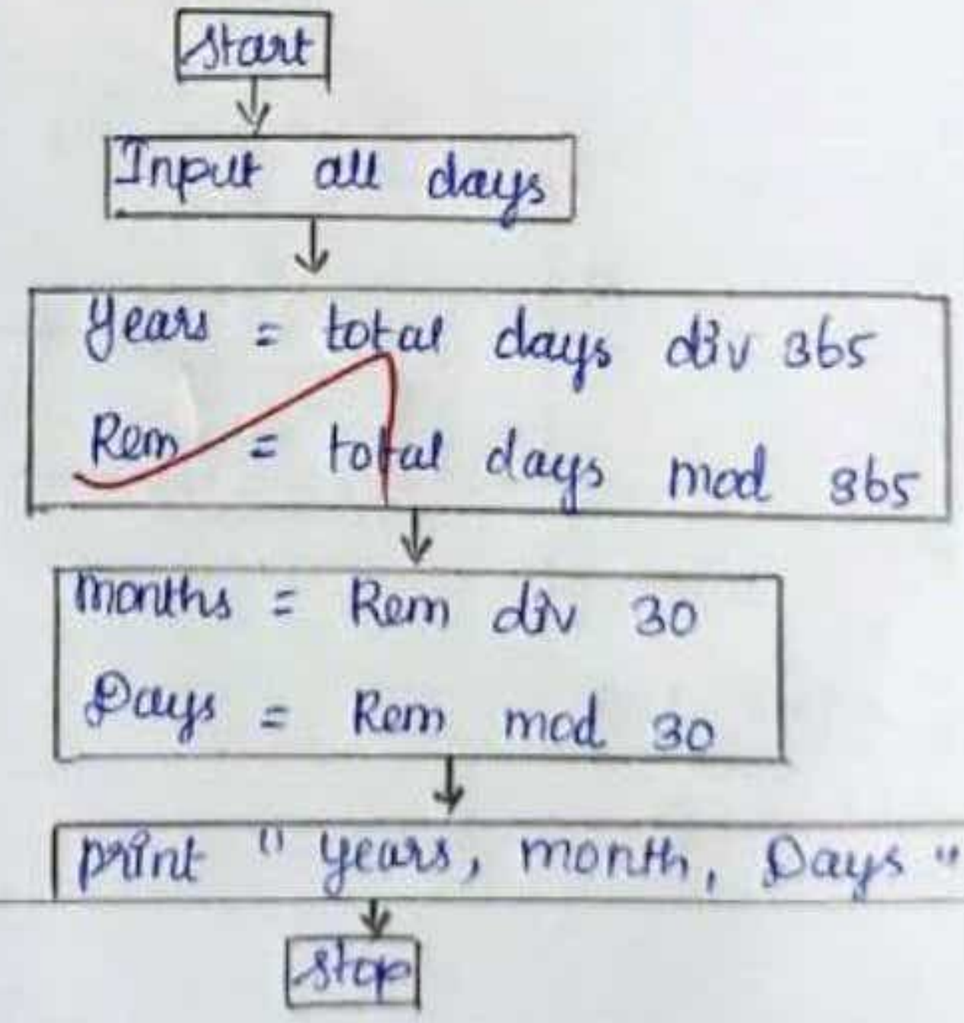
step 5 : [Compute months]
months = Rem div 30

Flowchart:

step6 : [compute Remaining days]
Days = Rem mod 30

step7 : printf "years, months, Days"

Step8 : Stop



Scanned with OKEN Scanner

Ex. No.: 3 **Date:**

Prime Number

Write an Algorithm and draw a Flowchart to check whether the given number is Prime or not.

Algorithm:

step 1 : start

step 2 : Read n, step 3 : Initialize i = 2

step 4 : if i=2 and i<n, goto step8

step 5 : i % 2 == 0 goto step 8

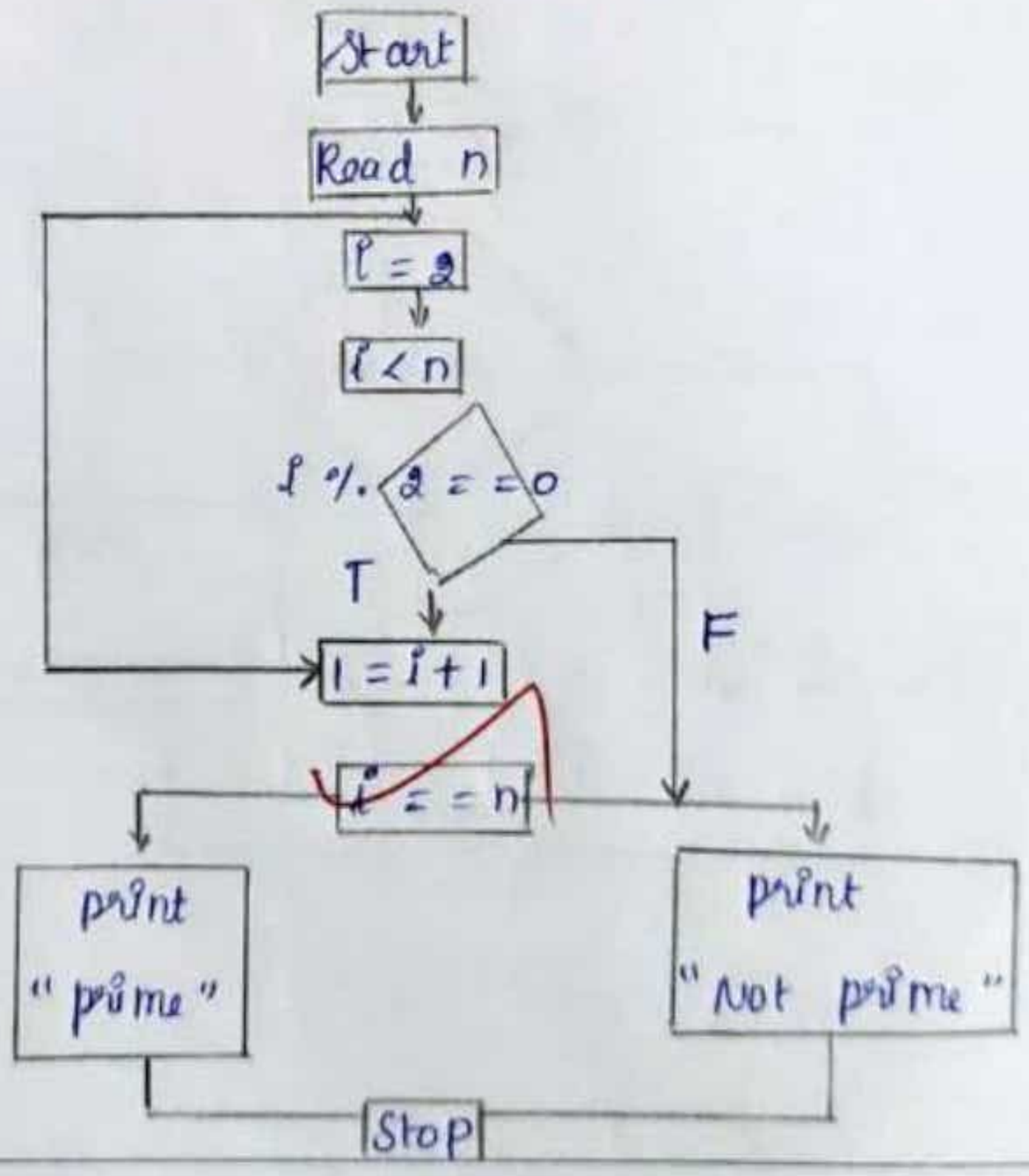
step 6 : i == n

step 7 : i = i+1 , goto step 4

step 8 : if i == n , print "Not prime" else print "prime"

step 7 : stop

Flowchart:



Scanned with OKEN Scanner

Ex. No.: 4 **Date:**

Leap Year

Write an Algorithm and draw a Flowchart to check whether the given year is Leap year or not.

Algorithm:

Step 1 : Start

Step 2 : Read year

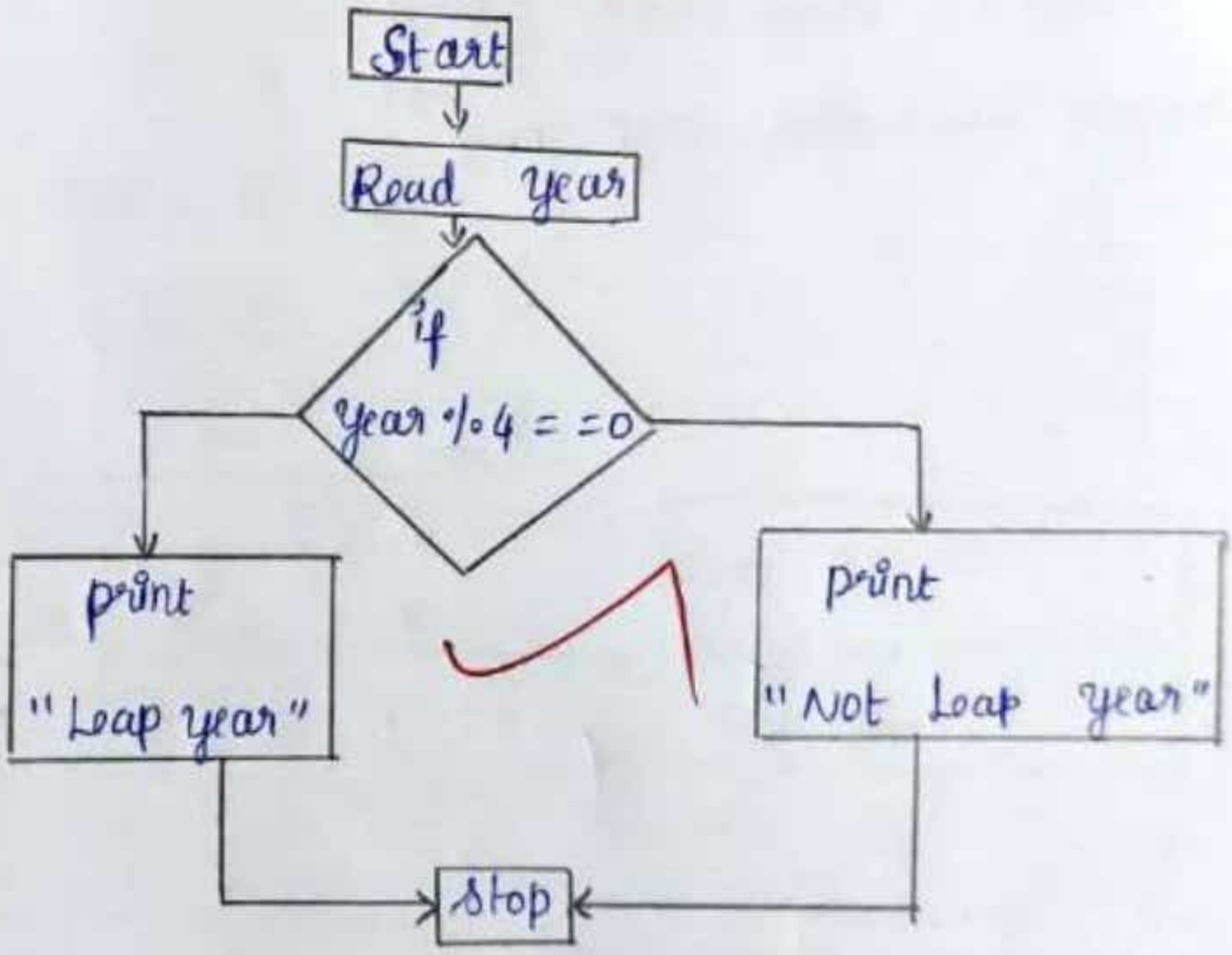
Step 3 : If year %. 4 ==0, then it is leap year

step 4 : else year %. 4 ! =0, then Not Leap year

step 5 : print "Leap year" or "Not Leap year"

Step 6 : Stop

Flowchart:



Scanned with OKEN Scanner

Ex. No.: 5 **Date:**

Palindrome Number

Write an Algorithm and draw a Flowchart to check whether the given number is palindrome number or not.

Algorithm:

Step 1 : Start

Step 2 : Read n

Step 3 : Declare temp = n, rev = 0

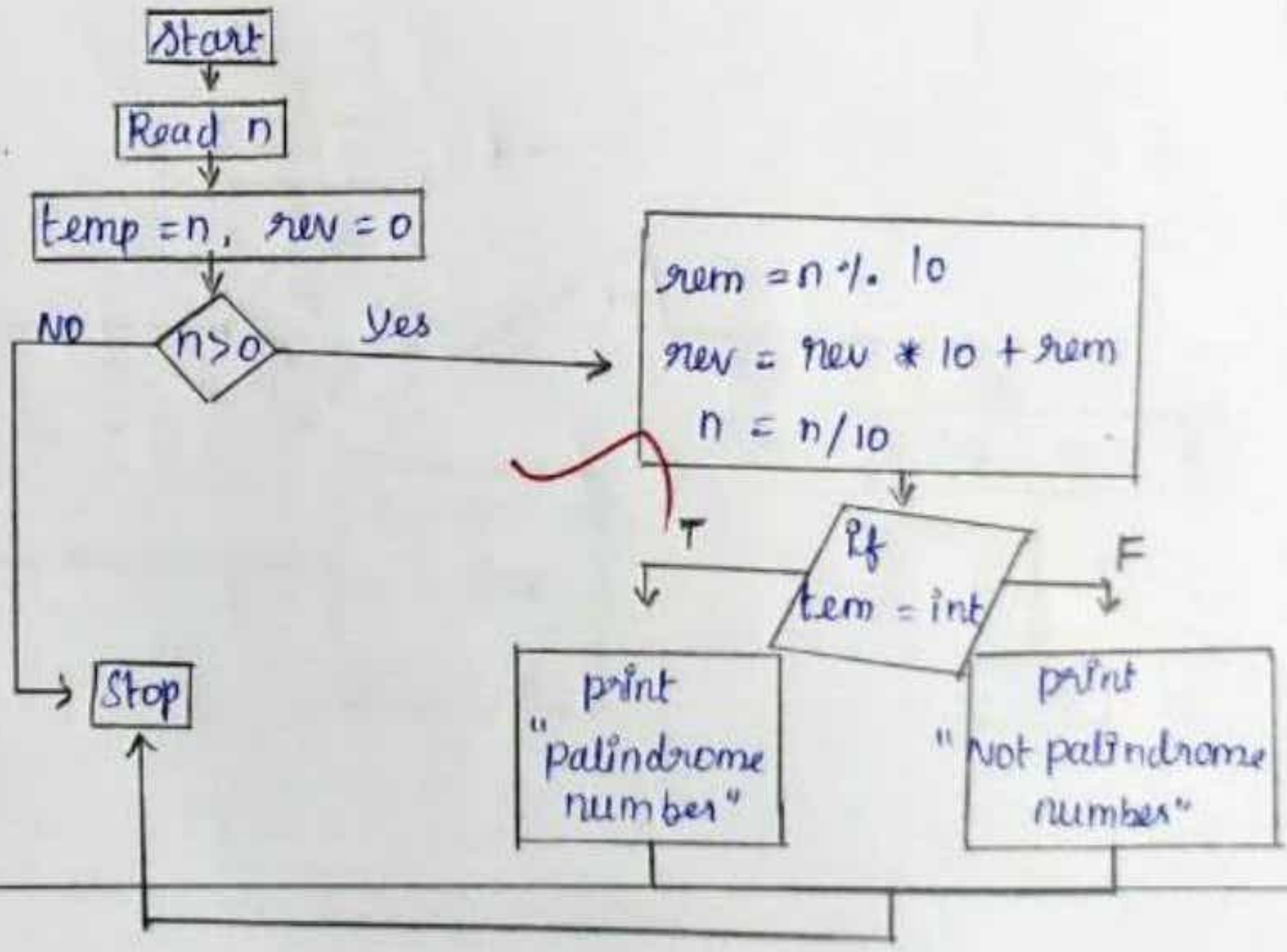
Step 4 : rem = n % 10
rev = rev * 10 + rem
n = n/10

Step 5 : if (n>0), goto step 4 to 6

Step 6 : if temp = = rev, then
print ("palindrome Number")
else
print ("Not palindrome number")

Step 7 : Stop

Flowchart:



Scanned with OKEN Scanner

Ex. No.: 6 **Date:**

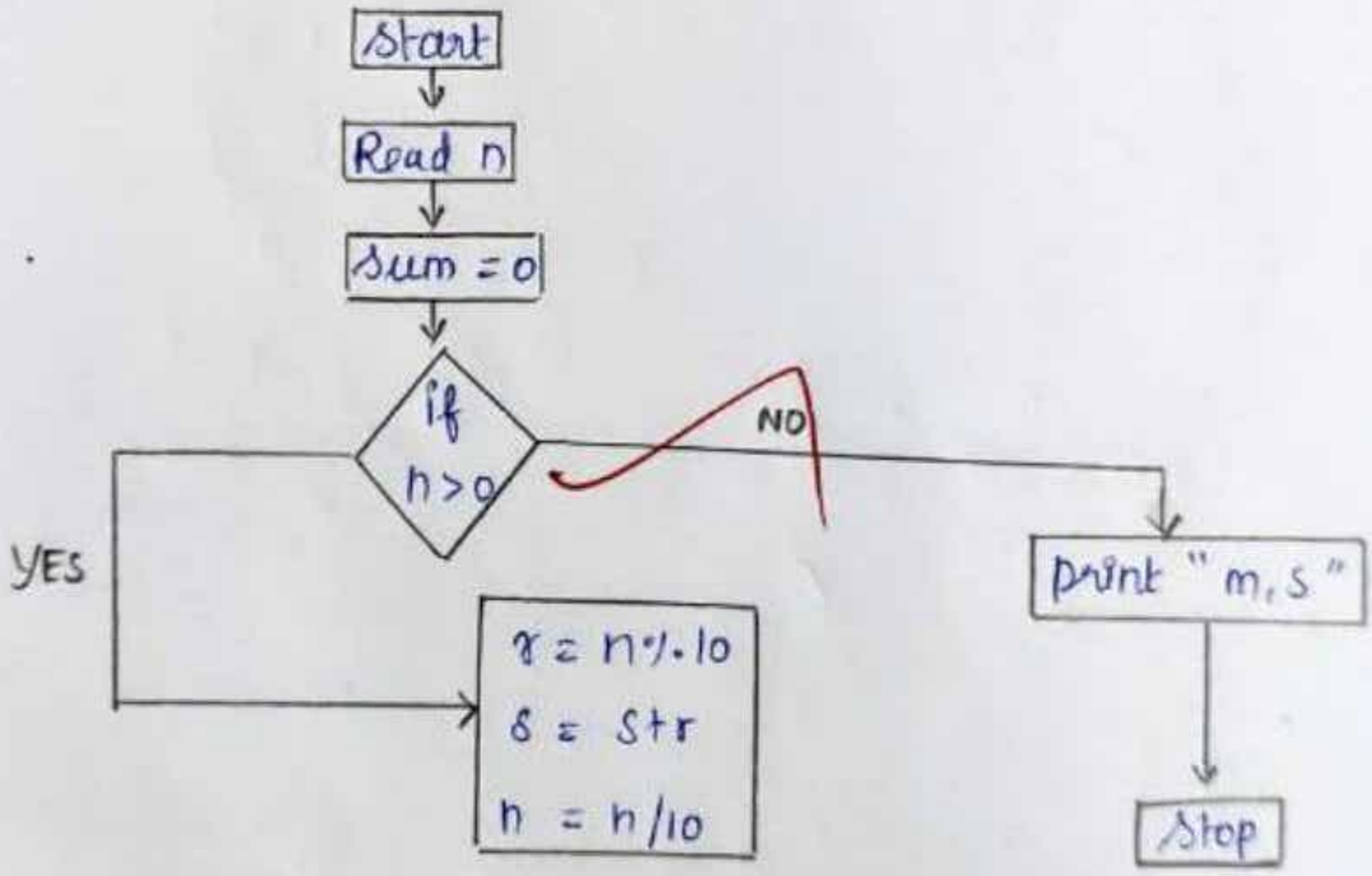
Sum of Digits

Write an Algorithm and draw a Flowchart to calculate the sum of digits in the given number.

Algorithm:

Step 1 : start
step 2 : Read n
Step 3 : s = 0, m = n
Step 4 : loop limit n=0
r = n %. 10
s = s+r
n = n/10
Step 5 : if n! =0 , goto step 4
else print
Step 6 : print sum
Step 7 : stop

Flowchart:



Scanned with OKEN Scanner

Overview of C, Constants, Variables and Data Types

Scanned with OKEN Scanner

Ex. No.: 1 **Date:**

Say "Hello, World!" With C

Problem Statement:

This is a simple challenge to help you practice printing to stdout.
We're starting out by printing the most famous computing phrase of all time! In the editor below, use either printf or cout to print the string Hello, World! to stdout.

Input Format
You do not need to read any input in this challenge.

Output Format
Print ***Hello, World!*** to stdout.

Sample Output 1
Hello, World!

Program:

```
#include <stdio.h>
int main()
{
printf("Hello, World!");
return 0;
}
```

Output:

Expected	Got
Hello, World!	Hello, World!

Result: Simple C program is implemented and executed Successfully.

Scanned with OKEN Scanner

Ex. No.: 2 **Date:**

Playing with Characters

Problem Statement:

This challenge will help you to learn how to take a character, a string and a sentence as input in C. To take a single character **ch** as input, you can use scanf("%c", &ch); and printf("%c", ch) writes a character specified by the argument char to stdout:

```
char ch;
scanf("%c", &ch);
printf("%c", ch);
```

This piece of code prints the character **ch**. You can take a string as input in C using scanf("%s", s). But it accepts string only until it finds the first space.
In order to take a line as input, you can use scanf("%[^\n] %*c", s); where **s** is defined as chars [**MAX_LEN**] where MAX_LEN is the maximum size of **s**. Here, [] is the scanset character. ^\n stands for taking input until a newline isn't encountered. Then, with this %*c, it reads the newline character and here, the used * indicates that this newline character is discarded.
Note: After inputting the character and the string, inputting the sentence by the above mentioned statement won't work. This is because, at the end of each line, a new line character(\n) is present. So, the statement: scanf("%[^\n] %*c", s); will not work because the last statement will read a newline character from the previous line. This can be handled in a variety of ways and one of them being: scanf("\n"); before the last statement.
Task: You have to print the character, **ch**, in the first line. Then print **s** in next line. In the last line print the sentence, **sen.**

Input Format
First, take a character, **ch** as input. Then take the string, **s** as input. Lastly, take the sentence **sen** as input

Output Format
Print three lines of output. The first line prints the character, **ch**. The second line prints the string, **s**. The third line prints the sentence, **sen.**

Sample Input 1
```
C
program
Programming using C
```
Sample Output 1
```
C
program
Programming using C
```


Scanned with OKEN Scanner

Program:

```
# Include < Stdio.h >

Int main ()
 {
  char ch;
  scanf ("%c", & ch);
  printf ("%c",  ch);
  return 0
 }
```

Output :

Input	Expected	Got
c	c	c

Result :

c program is implemented and Executed successfully.

Scanned with OKEN Scanner

Ex. No.: 3 **Date:**

Sum and Difference of Two Numbers

Problem Statement:

The fundamental data types in c are int, float and char. Today, we're discussing int and float data types.

The printf() function prints the given statement to the console. The syntax is printf("format string",argument_list);. In the function, if we are using an integer, character, string or float as argument, then in the format string we have to write %d (integer), %c (character), %s (string), %f (float) respectively.

The scanf() function reads the input data from the console. The syntax is scanf("format string",argument_list);. For ex: The scanf("%d",&number) statement reads integer number from the console and stores the given value in variable ***number***.
To input two integers separated by a space on a single line, the command is scanf("%d %d", &n, &m), where ***n*** and ***m*** are the two integers.

Task
Your task is to take two numbers of int data type, two numbers of float data type as input and output their sum:

1. Declare ***4*** variables: two of type int and two of type float.
2. Read ***2*** lines of input from stdin (according to the sequence given in the 'Input Format' section below) and initialize your ***4*** variables.
3. Use the + and - operator to perform the following operations:
   - Print the sum and difference of two int variable on a new line.
   - Print the sum and difference of two float variable rounded to one decimal place on a new line.

Input Format
The first line contains two integers. The second line contains two floating point numbers.
Constraints: $1 \leq$ ***integer variables*** $\leq 10^4$, $1 \leq$ ***float variables*** $\leq 10^4$

Output Format
Print the sum and difference of both integers separated by a space on the first line, and the sum and difference of both float (scaled to ***1*** decimal place) separated by a space on the second line.

Sample Input
10 4
4.0 2.0

Sample Output
14 6
6.0 2.0

Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main ()
{
  int a, b;
  float c, d;
  scanf ("%d %d", &a, &b);
  scanf ("%f %f", &c, &d);
  printf ("%d %d \n", a+b, a-b);
  printf ("%.1f %.1f", c+d, c-d);
  return 0;
}
```

Output :

Input		Expected		Got	
10	4	14	6	14	6
4.0	2.0	6.0	2.0	6.0	2.0
20	8	28	12	28	12
8.0	4.0	12.0	4.0	12.0	4.0

Result : C program implemented and executed Successfully.

Scanned with OKEN Scanner

Ex. No.: 4 **Date:**

Average Marks

Problem Statement

Write a program to input a name (as a single character) and marks of three tests as m1, m2, and m3 of a student considering all the three marks have been given in integer format.

Now, you need to calculate the average of the given marks and print it along with the name as mentioned in the output format section.

All the test marks are in integers and hence calculate the average in integer as well. That is, you need to print the integer part of the average only and neglect the decimal part.

Input Format :

Line 1 : Name(Single character)
Line 2: Marks scored in the 3 tests separated by single space.

Output Format:
First line of output prints the name of the student. Second line of the output prints the average mark.

Constraints
Marks for each student lie in the range 0 to 100 (both inclusive)

Sample Input 1 :
```
A
3 4 6
```

Sample Output 1 :
```
A
4
```


Scanned with OKEN Scanner

Program:

```
# Include <StdIo.h>
int main ()
 { char name;
int m1, m2, m3;
int average;
Scanf ("%c", &name);
Scanf ("%d %d %d", &m1, &m2, &m3);
Average = (m1+m2+m3) /3;
printf ("%c\n", name);
printf ("%d \n", average);
 return 0;
 }
```

Output

Input	Expected	Got
A	A	A
34 b	4	4

Result: C program Implemented and Executed Successfully.

Scanned with OKEN Scanner

Ex. No.: 5 **Date:**

Basic Data Types

Problem Statement:

Some C data types, their format specifiers, and their most common bit widths are as follows:

- *Int ("%d")*: 32 Bit integer
- *Long ("%ld")*: 64 bit integer
- *Char ("%c")*: Character type
- *Float ("%f")*: 32 bit real value
- *Double ("%lf")*: 64 bit real value

Reading

To read a data type, use the following syntax: scanf("`format_specifier`", &val)
For example, to read a *character* followed by a *double*: char ch;

```
double d;
scanf("%c %lf", &ch, &d);
```

For the moment, we can ignore the spacing between format specifiers.

Printing

To print a data type, use the following syntax: printf("`format_specifier`", val)
For example, to print a *character* followed by a *double*: char ch = 'd';

```
double d = 234.432;
printf("%c %lf", ch, d);
```

Note: You can also use *cin* and *cout* instead of *scanf* and *printf*; however, if you are taking a million numbers as input and printing a million lines, it is faster to use *scanf* and *printf*.

Input Format

Input consists of the following space-separated values: *int*, *long*, *char*, *float*, and *double*, respectively.

Output Format

Print each element on a new line in the same order it was received as input. Note that the floating-point value should be correct up to 3 decimal places and the double to 9 decimal places.

Sample Input

```
3
12345678912345
a
334.23
14049.30493
```

Sample Output

```
3
12345678912345
a
334.230
14049.304930000
```


Scanned with OKEN Scanner

Program:

```
# include <Stdio.h>
int main () {
    int a;
    long b;
    char c;
    float d;
    double e;
    scanf ("%d %ld %c %f %lf", &a, &b, &c, &d, &e);
    printf ("%d\n", a);
    printf ("%ld\n", b);
    printf ("%c\n", c);
    printf ("%.3f\n", d);
    printf ("%.9lf\n", e);
    return 0;
}
```

Output :

Input	Expected	Got
3	3	3
12345678912345	12345678912345	12345678912345
a	a	a
334.23 14049.30493	334.230	334.230
	14049.304930000	14049.304930000

Result : C program implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No.: 6 **Date:**

ASCII Value and Adjacent Characters

Problem Statement:

Write a program to print the ASCII value and the two adjacent characters of the given character.

Input Format: Reads the character

Output Format: First line prints the ascii value, second line prints the previous character and next character of the input character

Sample Input 1:

E

Sample Output 1:

69
D F

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
  char ch;
  scanf ("%c", & ch);
  int a = (int)ch;
  printf ("%d\n", a);
  Char b = ch-1;
 Char  c = ch+1;
 printf (" %c  %-c \n", b, c);
 return 0;
 }
```

Output :

Input	Expected	Got
E	69	69
	D F	DF

Result : C program implemented and executed successfully.

Scanned with OKEN Scanner

Operators and Expressions, Managing Input and Output Operations

Scanned with OKEN Scanner

Ex. No.: 7 **Date:**

Height Units

Problem Statement:

Many people think about their height in feet and inches, even in some countries that primarily use the metric system. Write a program that reads a number of feet from the user, followed by a number of inches. Once these values are read, your program should compute and display the equivalent number of centimeters.

Hint: One foot is 12 inches. One inch is 2.54 centimeters.

Input Format

First line, read the number of feet.

Second line, read the number of inches.

Output Format

In one line print the height in centimeters.

Note: All of the values should be displayed using two decimal places.

Sample Input 1

5

6

Sample Output 1

167.64

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
    int a ;
    int b;
    scanf ("%d", &a );
    scanf ("%d", & b);
    printf ("%.2f", (a* 12* 2.54)+(b* 2.54));
    return 0;
}
```

Output

Input	Got
5	167.64
6	

Result : C program implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No.: 8 Date:

Arithmetic

Problem Statement:

Create a program that reads two integers, a and b, from the user. Your program should compute and display:

- The sum of a and b
- The difference when b is subtracted from a
- The product of a and b
- The quotient when a is divided by b
- The remainder when a is divided by b

Input Format
First line, read the first number.
Second line, read the second number.

Output Format
First line, print the sum of a and b
Second line, print the difference when b is subtracted from a
Third line, print the product of a and b
Fourth line, print the quotient when a is divided by b
Fifth line, print the remainder when a is divided by b

Sample Input 1
100
6

Sample Output1
106
94
600
16
4

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
  int a, b;
  Scanf ("%d %d", &a, &b);
  printf ("%d\n", a+b);
  printf ("%d\n", a-b);
  printf ("%d\n", a*b);
  printf ("%d\n", a/b);
  printf ("%d\n", a%b);
  return 0;
 }
```

Output :

Input	Got
100	106
	94
	600
6	16
	4

Result : C - program is implemented and executed Successfully.

Scanned with OKEN Scanner

Ex. No.: 9 **Date:**

Day Old Bread

Problem Statement:

A bakery sells loaves of bread for $3.49 each. Day old bread is discounted by 60 percent. Write a program that begins by reading the number of loaves of day-old bread being purchased from the user. Then your program should display the regular price for the bread, the discount because it is a day old, and the total price. Each of these amounts should be displayed on its own line with an appropriate label. All of the values should be displayed using two decimal places.

Input Format
Read the number of day old loaves.

Output Format
First line, print Regular price: price Second line, print Discount: discount Third line, print Total: total

Note: All of the values should be displayed using two decimal places.

Sample Input 1
10

Sample Output 1
Regular price: 34.90 Discount: 20.94 Total: 13.96

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
   int a
   float b, c, d;
   scanf ("%d", &a);
   b = a* 3.49
printf ("Regular price : %.2f\n", b);
c = b* 0.6;
printf (" Discount : %.2f\n ", c);
d = b - c;
printf (" Total : %.2f\n ", d);
   return 0;
}
```

Output :

Input	Got
10	Regular price : 34.90
	Discount : 20.94
	Total : 13.96

Result :

c program implemented and executed successfully

Scanned with OKEN Scanner

Ex. No. : 10 **Date :**

Goki and his Breakup

Problem Statement:

Goki recently had a breakup, so he wants to have some more friends in his life. Goki has **N** people who he can be friends with, so he decides to choose among them according to their skills set **Yi**(1<=i<=n). He wants atleast **X** skills in his friends.
Help Goki find his friends.

Input Format
First line contains a single integer **X** - denoting the minimum skill required to be Goki's friend. Next line contains one integer **Y** - denoting the skill of the person.

Output Format
Print if he can be friend with Goki. **'YES' (without quotes)** if he can be friends with Goki else **'NO' (without quotes)**.

Constraints
1<=N<=1000000 1<=X,Y<=1000000

SAMPLE INPUT 1
100
110

SAMPLE OUTPUT 1
YES

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
  int x,y;
scanf ("%d %d", & x, & y);
if ( y>=x)
{
  printf ("YES\n");
}
else
 {
  printf ("NO\n");
  }
return 0;
}
```

Output :

Input	Got
100 110	YES
100 90	NO

Result : C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 11 **Date :**

Say no to Handshakes!!!

Problem Statement:

Before the outbreak of corona virus to the world, a meeting happened in a room in Wuhan. A person who attended that meeting had COVID-19 and no one in the room knew about it! So, everyone started shaking hands with everyone else in the room as a gesture of respect and after meeting unfortunately everyone got infected! Given the fact that any two persons shake hand exactly once, can you tell the total count of handshakes happened in that meeting?

Say no to shakehands. Regularly wash your hands. Stay Safe.

Input Format
Read an integer N, the total number of people attended that meeting.

Output Format
Print the number of handshakes.

Constraints
$0 < N < 106$

SAMPLE INPUT 1
1

SAMPLE OUTPUT
0

Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main()
{
  int n;
  Scanf ("%d", &n);
  int hand_shake = (n * (n-1)) /2 ;
  printf ("%d", hand_shake );
  return 0;
}
```

Output :

Input	Got
1	0
2	1

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 12 **Date :**

Back to School

Problem Statement:

In our school days, all of us have enjoyed the Games period. Raghav loves to play cricket and is Captain of his team. He always wanted to win all cricket matches. But only one last Games period is left in school now. After that he will pass out from school.

So, this match is very important to him. He does not want to lose it. So he has done a lot of planning to make sure his teams wins. He is worried about only one opponent - Jatin, who is very good batsman.

Raghav has figured out 3 types of bowling techniques, that could be most beneficial for dismissing Jatin. He has given points to each of the 3 techniques.

You need to tell him which is the maximum point value, so that Raghav can select best technique.

3 numbers are given in input. Output the maximum of these numbers.

Input Format:
Three space separated integers.

Output Format:
Maximum integer value

SAMPLE INPUT
8 6 1

SAMPLE OUTPUT
8

Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main()
{
  int a, b, c;
scanf("%d %d %d", &a, &b, &c);
if(a>b && a>c)
{
 printf("%d", a);
}
else if(b>c && b>a)
{
 printf("%d", b);
}
else {
     printf("%d", c);
    }
}
```

Output:

Input	Got
81 26 15	81

Result:

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Decision Making and Branching – if, if...else, nested if...else, if...else if, Switch-Case

Scanned with OKEN Scanner

Ex. No. : 13 **Date :**

Same Digit

Problem Statement:

Write a program to read two integer values and print true if both the numbers end with the same digit, otherwise print false.
Example: If 698 and 768 are given, program should print true as they both end with 8.

Sample Input 1
25 53

Sample Output 1
false

Sample Input 2
27 77

Sample Output 2
true

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
Int main () {
    Int a, b, c, d ;
    scanf ("%d %d", &a, &b);
    c = a % 10 ;
    d = b % 10 ;
    if (c == d)
    {
      printf ("true");
    }
    else
      {
       printf ("false");
      }
      return 0;
  }
```

Output :

Input	Got
25 53	false
27 77	true

Result :

C program is Implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 14 **Date :**

Intro to Conditional Statements

Problem Statement:

In this challenge, we're getting started with conditional statements.
Task
Given an integer, ***n***, perform the following conditional actions:

- If ***n*** is odd, print Weird
- If ***n*** is even and in the inclusive range of ***2*** to ***5***, print ***Not Weird***
- If ***n*** is even and in the inclusive range of ***6*** to ***20***, print ***Weird***
- If ***n*** is even and greater than ***20***, print ***Not Weird***

Complete the stub code provided in your editor to print whether or not ***n*** is weird.

Input Format
A single line containing a positive integer, ***n***.

Constraints
• $1 < n < 100$

Output Format
Print Weird if the number is weird; otherwise, print Not Weird.

Sample Input 0
3

Sample Output 0
Weird

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ( ) {
    int n;
    scanf ("%d", &n);
    if (n%2 == 0 && 2 <= n && n <= 5)
    {
      printf ("Not Weird");
    }
    else if (n%2 == 0 && 6 <= n && n <= 20)
    {
    printf ("Weird");
    }
    else if (n%2 == 0 && n >= 20)
    { printf ("Not Weird");
    }else {
        printf ("Weird");
        }
    }
```

Output :

Input	Got
3	Weird
24	Not Weird

Result :

C program is implemented and executed succesfully.

Scanned with OKEN Scanner

Ex. No. : 15 **Date :**

Pythagorean Triples

Problem Statement:

Three numbers form a Pythagorean triple if the sum of squares of two numbers is equal to the square of the third.

For example, 3, 5 and 4 form a Pythagorean triple, since 3*3 + 4*4 = 25 = 5*5 .

You are given three integers, a, b, and c. They need not be given in increasing order. If they form a Pythagorean triple, then print "yes", otherwise, print "no". Please note that the output message is in small letters.

Sample Input 1
3
5
4

Sample Output 1
yes

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int a, b, c ;
    scanf ("%d %d %d", &a, &b, &c);
if ((a*a +b*b ==c*c) || (b*b+c*c ==a*a) ||
   (c*c+a*a ==b*b))
  { printf ("yes");
  }
  else
  { printf ("no");
  }
  return 0;
}
```

Output :

Input	Got
3 5 4	Yes
5 8 2	no

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 16 **Date :**

Name That Shape

Problem Statement:

Write a program that determines the name of a shape from its number of sides. Read the number of sides from the user and then report the appropriate name as part of a meaningful message. Your program should support shapes with anywhere from 3 up to (and including) 10 sides. If a number of sides outside of this range is entered then your program should display an appropriate error message.

Sample Input 1
3

Sample Output 1
Triangle

Sample Input 2
7

Sample Output 2
Heptagon

Sample Input 3
11

Sample Output 3
The number of sides is not supported.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{ int a;
  scanf ("%d", &a);
  switch (a)
{ case 1 : {
    printf ("Monoangle");
    break;
    }
    case 2 : {
    printf ("Diangle");
    break;
    }
    case 3 : {
    printf ("Triangle");
    break;
    }
    case 4 : {
    printf ("Quadrangle");
    break;
    }
    case 5 : {
    printf ("pentagon");
    break;
    }
```


Scanned with OKEN Scanner

```
case 6 : {
   printf ("Hexagon");
   break;
   }
case 7 : {
   printf ("Heptagon");
   break;
   }
case 8 : {
   printf ("Octagon");
   break; }
case 9 : {
   printf ("Nonagon");
   break; }
case 10 : {
   printf ("Decagon");
   break ; }
case 11 : {
   printf ("The number of sides is not supported.");
   break ;
   }
}
```

Output :

Input	Got
3	Triangle
7	Heptagon
11	The number of sides is not supported.

Result :

C program is implemented and executed sucessfully.

Scanned with OKEN Scanner

Ex. No. : 17 Date :

Chinese Zodiac

Problem Statement:

The Chinese zodiac assigns animals to years in a 12-year cycle. One 12-year cycle is shown in the table below. The pattern repeats from there, with 2012 being another year of the Dragon, and 1999 being another year of the Hare.

Year	Animal
2000	Dragon
2001	Snake
2002	Horse
2003	Sheep
2004	Monkey
2005	Rooster
2006	Dog
2007	Pig
2008	Rat
2009	Ox
2010	Tiger
2011	Hare

Write a program that reads a year from the user and displays the animal associated with that year. Your program should work correctly for any year greater than or equal to zero, not just the ones listed in the table.

Sample Input 1
2004

Sample Output 1
Monkey

Sample Input 2
2010

Sample Output 2
Tiger

Scanned with OKEN Scanner

Program:

```
# include <Stdio.h>
int main()
{ int year;
  scanf ("%d", & year);
  if (year % 12 == 0) {
    printf ("Monkey");
    }
else if (year %-12 ==1){
    printf ("Rooster");
    }
else if (year %12 == 2){
    printf ("Dog");
    }
else if (year%12 ==3){
    printf ("pig");
    }
else if (year%-12 == 4){
    printf ("Rat");
   }
else if (year %-12 == 5){
   printf ("Ox");
   }
 else if (year %-12 == 6){
  printf ("Tiger");
 }
else if (year %12 == 7){
```


Scanned with OKEN Scanner

```
        printf (" Hare ");
    }
    else if ( year %.12 == 8 ){
        printf ( " Donkey " );
    }
    else if ( year %.12 == 9 ){
        printf ( " Snake " );
    }
    else if ( year %12 == 10) {
        printf ( " Horse " );
    }
    else if ( year %12 == 11) {
        printf ( " Sheep " );
    }
    return 0;
}
```

Output :

Input	Got
2004	Monkey
2010	Tiger

Result :

C - program is implemented and executed successfully.

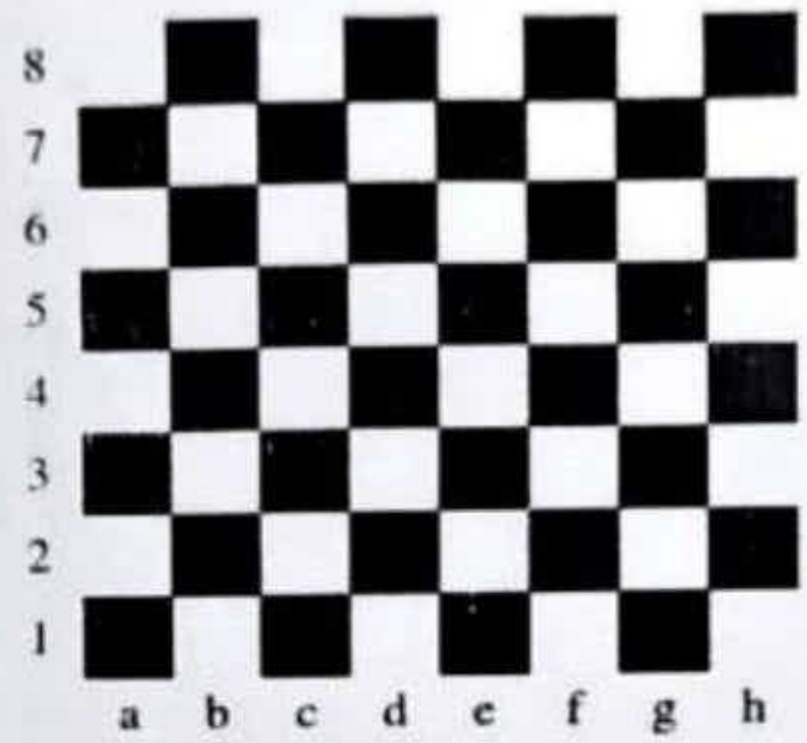
Scanned with OKEN Scanner

Ex. No. : 18 **Date :**

What Color Is That Square?

Problem Statement:

Positions on a chess board are identified by a letter and a number. The letter identifies the column, while the number identifies the row, as shown below:



Write a program that reads a position from the user. Use an if statement to determine if the column begins with a black square or a white square. Then use modular arithmetic to report the color of the square in that row. For example, if the user enters a1 then your program should report that the square is black. If the user enters d5 then your program should report that the square is white. Your program may assume that a valid position will always be entered. It does not need to perform any error checking.

Sample Input 1
a 1

Sample Output 1
The square is black.

Sample Input 2
d 5

Sample Output 2
The square is white.

Scanned with OKEN Scanner

Program:

```
# Include <Stdio.h>
int main () {
   Int a, Sum ;
    Char c;
    Scanf ("%c %d", &c, &a);
    Sum = c+a ;
    if (sum %.2 == 0)
{
 printf ("The square is black.");
}
else
 {
  printf ("The square is White.");
 }
 return 0;
}
```

Output:

Input	Got
a 1	The square is black.
d 5	The square is White.

Result :

C program is Implemented and executed Successfully.

Scanned with OKEN Scanner

Ex. No. : 19 **Date :**

Day of Year

Problem Statement:

Some data sets specify dates using the year and day of year rather than the year, month, and day of month. The day of year (DOY) is the sequential day number starting with day 1 on January 1st.

There are two calendars - one for normal years with 365 days, and one for leap years with 366 days. Leap years are divisible by 4. Centuries, like 1900, are not leap years unless they are divisible by 400. So, 2000 was a leap year.

To find the day of year number for a standard date, scan down the Jan column to find the day of month, then scan across to the appropriate month column and read the day of year number. Reverse the process to find the standard date for a given day of year.
Write a program to print the Day of Year of a given date, month and year.

Sample Input 1
18
6
2020

Sample Output 1
170

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main() {
    int d, m, y, sum = 0;
    scanf("%d %d %d", &d, &m, &y);
int month[100] = {31, 28, 31, 30, 31, 30, 31, 31, 30,
                  31, 30, 31, };
for (int i=0 ; i<m-1; i++)
{ sum = sum + month[i];
}
 int tot = sum+d;
if (y%4==0 && y%400==0)
{ int ntot = tot+1 ;
   printf("%d", ntot);
}
else if (y%100]=0)
{
  int ntot = tot+1;
  printf("%d", ntot);
}
else
  { printf("%d", tot);
  }
return 0;
}
```


Scanned with OKEN Scanner

Output

Input	Got
18	170
6	
2020	

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 20 **Date :**

Suppandi & Areas

Problem Statement:

Suppandi is trying to take part in the local village math quiz. In the first round, he is asked about shapes and areas. Suppandi, is confused, he was never any good at math. And also, he is bad at remembering the names of shapes. Instead, you will be helping him calculate the area of shapes.

- When he says rectangle, he is actually referring to a square.
- When he says square, he is actually referring to a triangle.
- When he says triangle, he is referring to a rectangle
- And when he is confused, he just says something random. At this point, all you can do is say 0.

Help Suppandi by printing the correct answer in an integer.

Input Format

- Name of shape (always in upper case R --> Rectangle, S --> Square, T --> Triangle)
- Length of 1 side
- Length of other side
  Note: In case of triangle, you can consider the sides as height and length of base

Output Format

- Print the area of the shape.

Sample Input 1

T
10
20

Sample Output 1

200

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
# include <ctype.h>
int main () {
 char a;
scanf ("%c", &a);
a = toupper(a);
int b, c, d, e, f;
scanf ("%d %d", &e, &b);
switch (a)
{
  case 'R' :
   c = e * b;
   printf ("%d", c);
   break;
  case 'S' :
  d = 0.5 * e * b;
  printf ("%d", d);
   break;
case 'T' :
f = e * b;
printf ("%d", f);
 break;
default : printf ("0");
 break; }
 return 0;
}
```


Scanned with OKEN Scanner

Output :

Input	Got
T	
10	200
20	
S	600
30	
40	
B	0
2	
11	
R	
10	300
30	
S	1000
40	
50	

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 21 **Date :**

Superman's Encounter

Problem Statement:

Superman is planning a journey to his home planet. It is very important for him to know which day he arrives there. They don't follow the 7-day week like us. Instead, they follow a 10-day week with the following days:

Day	Number Name of Day
1	Sunday
2	Monday
3	Tuesday
4	Wednesday
5	Thursday
6	Friday
7	Saturday
8	Kryptonday
9	Coluday
10	Daxamday

Here are the rules of the calendar:

- The calendar starts with Sunday always.
- It has only 296 days. After the 296th day, it goes back to Sunday.

You begin your journey on a Sunday and will reach after n. You have to tell on which day you will arrive when you reach there.

Input format:

- Contain a number n (0 < n)

Output format:
Print the name of the day you are arriving on

Sample Input
7

Sample Output
Kryptonday

Sample Input
1

Sample Output
Monday

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
        int num;
        scanf ("%d", & num);
        num = num % 296;
        int b;
        b = (num % 10) + 1;
        switch (b)
{
  case 1 :
   printf (" Sunday ");
   break;
  case 2 :
   printf (" Monday");
   break;
  case 3 :
   printf (" Tuesday");
   break;
  case 4 :
   printf (" Wednesday ");
   break;
  case 5 :
   printf (" Thursday ");
   break;
  case 6 :
   printf (" Friday");
```


Scanned with OKEN Scanner

```
        break ;
    Case 7 :
        printf (" Saturday ") ;
        break ;
    Case 8 :
        printf (" kryptonday");
        break ;
    Case 9 :
        printf (" Coluday");
        break ;
    case 10 ;
        printf (" Daxamday ");
        break ;
    }
    return 0 ;
}
```

Output :

Input	Got
7	kryptonday
1	Monday

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Decision Making and Looping – while and do...while, for

Scanned with OKEN Scanner

Ex. No. : 22 **Date :**

Stone Game-One Four

Problem Statement:

Alice and Bob are playing a game called "Stone Game". Stone game is a two-player game. Let N be the total number of stones. In each turn, a player can remove either one stone or four stones. The player who picks the last stone, wins. They follow the "Ladies First" norm. Hence Alice is always the one to make the first move. Your task is to find out whether Alice can win, if both play the game optimally.

Input Format
First line starts with T, which is the number of test cases. Each test case will contain N number of stones.

Output Format
Print "Yes" in the case Alice wins, else print "No".

Constraints 1<=T<=1000 1<=N<=10000

Sample Input
3
1
6
7

Sample Output
Yes
No

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
int main () {
    int T, I = 0, n, t ;
    scanf ("%d", &T );
    while ( I < T)
{  scanf ("%d", & n);
   t = n/4 ;
   If ( t %. 2 == =0  && n%.2 = =0)
   {  printf ("No \n");
    }
   else if ( t %. 2 = =1  && n %. 2 = =1)
   {  printf ("No\n");
     }
   else
     {  printf ("Yes \n");
     }
     I++;
   }
   return 0;
}
```

Output :

Input	Got
3	Yes
1	Yes
6	No
7	

Result : C program is Implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 29 **Date :**

Holes in a Number

Problem Statement:

You are designing a poster which prints out numbers with a unique style applied to each of them. The styling is based on the number of closed paths or holes present in a given number.

The number of holes that each of the digits from 0 to 9 have are equal to the number of closed paths in the digit. Their values are:

1, 2, 3, 5, 7 = 0 holes.
0, 4, 6, 9 = 1 hole.
8 = 2 holes.

Given a number, you must determine the sum of the number of holes for all of its digits. For example, the number 819 has 3 holes.
Complete the program, it must return an integer denoting the total number of holes in num.

Constraints
$1 \leq num \leq 109$

Input Format For Custom Testing
There is one line of text containing a single integer num, the value to process.

Sample Input
630

Sample Output
2

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int  a, b, n = 0;
    scanf ("%d", &a);
while (a > 0)
{
  b = a % 10;
if ( b == 0 || b == 6 || b == 9 || b == 4 )
{  n = n + 1;
 }
else if (b == 8)
{ n = n + 2;
 }
   a = a/10 ;
 }
printf ("%d", n);
 return 0;
}
```

Output :

Input	Got
630	2
1288	4

Result :

C program is implemented and executed succesfully.

Scanned with OKEN Scanner

Ex. No. : 2A Date :

Philaland Coin

Problem Statement:

The problem solvers have found a new Island for coding and named it as Philaland. These smart people were given a task to make a 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 the 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 till $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 a minimum number of denominations for any arbitrary max price in Philaland.

Input Format
Contains an integer N denoting the maximum price of the item present on Philaland.

Output Format
Print a single line denoting the minimum number of denominations of coins required.

Constraints
1<=T<=100 1<=N<=5000

Sample Input 1:
10

Sample Output 1:
4

Scanned with OKEN Scanner

Program:

```
# include<stdio.h>
int main() {
    int n, r;
    scanf("%d %d", &n, &r);
    while (n != 0)
    {
      n = n/2;
      r = r+1;
    }
    printf("%d", r);
    return 0;
}
```

Output :

Input	Got
10	4
5	3
20	5
500	9
1000	10

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 25 Date :

Number Count

Problem Statement:

A set of N numbers (separated by one space) is passed as input to the program. The program must identify the count of numbers where the number is odd number.

Input Format:
The first line will contain the N numbers separated by one space.

Boundary Conditions:
3 <= N <= 50
The value of the numbers can be from -99999999 to 99999999

Output Format:
The count of numbers where the numbers are odd numbers.

Sample Input:
5 10 15 20 25 30 35 40 45 50

Sample Output:
5

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{ int n, x = 0;
  while (scanf ("%d", &n) == 1)
  { if (n%2 != 0)
    { x++ ;
    }
  }
  printf ("%d", x);
  return 0;
}
```

Output :

Input	Got
5 10 15 20 25 30 35 40 45 50	5

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 26 **Date :**

Confusing Number

Problem Statement:

Given a number N, return true if and only if it is a *confusing number*, which satisfies the following condition:
We can rotate digits by 180 degrees to form new digits. When 0, 1, 6, 8, 9 are rotated 180 degrees, they become 0, 1, 9, 8, 6 respectively. When 2, 3, 4, 5 and 7 are rotated 180 degrees, they become invalid. A *confusing number* is a number that when rotated 180 degrees becomes a **different** number with each digit valid.

Example 1:

6 → 9

Input: 6
Output: true
Explanation: We get 9 after rotating 6, 9 is a valid number and 9 != 6.

Example 2:

89 → 68

Input: 89
Output: true
Explanation: We get 68 after rotating 89, 86 is a valid number and 86 != 89.

Example 3:

11 → 11

Input: 11
Output: false
Explanation: We get 11 after rotating 11, 11 is a valid number but the value remains the same, thus 11 is not a confusing number.

Example 4:

25 → 25

Input: 25
Output: false
Explanation: We get an invalid number after rotating 25.

Note:.

1. 0 <= N <= 10^9
2. After the rotation we can ignore leading zeros, for example if after rotation we have 0008 then this number is considered as just 8.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int n, x, y = 1 ;
    scanf ("%d", &n );
    while (n! = 0 && y == 1)
    {
    x = n %. 10 ; n = n/10;
    if (x == 2 || x == 3 || x == 4 || x == 7)
    { y++ ;
    }
    }
    if (y == 1)
    { printf ("true");
    }else
    { printf ("false");
    }
    return 0;
}
```

Output:

Input	Got
6	true
89	true
25	false

Result : C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 27 **Date :**

Nutrition Value

Problem Statement:

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration: Given *4* food items (hence value: *1,2,3* and *4*), and the unhealthy sum being *6* macronutrients, on choosing items *1, 2, 3* -> the sum is *6*, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- *2 + 3 + 4 = 9*
- *1 + 3 + 4 = 8*
- *1 + 2 + 4 = 7*

Since *2 + 3 + 4 = 9*, allows for maximum number of macronutrients, 9 is the right answer. Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo *1000000007 (109 + 7)*.

It has the following:

n: an integer that denotes the number of food items

k: an integer that denotes the unhealthy number

Constraints

- $1 \leq n \leq 2 \times 109$
- $1 \leq k \leq 4 \times 1015$

Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of food items. The second line contains an integer, *k*, that denotes the unhealthy number.

Sample Input 0

2
2

Sample Output 0

3

Scanned with OKEN Scanner

Program:

```
# include <Stdio.h>
int main() {
    long long int n, t, i, num = 0;
    scanf ("%lld %lld", &n, &t);
    for (i=1 ; i<=n; i++)
    {
      num = num + i;
      if (num == t)
      {
        num = num - 1;
      }
    }
    printf ("%lld", num % 1000000007);
    return 0;
}
```

Output:

Input	Got
2 2	3
2 1	2
3 3	5

Result:

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Nested Loops - while and for, Jumps in Loops

Scanned with OKEN Scanner

Ex. No. : 28 Date :

Simple Chessboard

Problem Statement:

Write a program that prints a simple chessboard.

Input format:
The first line contains the number of inputs T.
The lines after that contain a different value for size of the chessboard

Output format:
Print a chessboard of dimensions size * size.
Print W for white spaces and B for black spaces.

Sample Input:
2
3
5

Sample Output:
WBW
BWB
WBW
WBWBW
BWBWB
WBWBW
BWBWB
WBWBW

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
int main () {
    Int T, d, I=0, i1, i2, 0;
     Char c ;
    Scanf ("%.d", & d);
    While (i<T)
{  Scanf ("%.d ", & d);
    i1 =0;
    While (i1 <d)
{   0 =1 ;
    i2 = 0;
    If ( i1 % 2 ==0)
    {
      0 =0;
    }
While (i2 <d)
{
    c = 'B';
    If (i2 %.2 ==0)
    {
      c ='W';
    }
    printf ("%.c ", c);
    i2++ ;
}
i1 + =1 ;
```


Scanned with OKEN Scanner

```
            printf("\n");
        }
        i = i+1;
    }
}
```

Output :

Input	Got
2	WBW
3	BWB
5	WBW
	WBWBW
	BWBWB
	WBWBW
	BWBWB
	WBWBW

Result :

C - program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 29 **Date :**

Print Our Own Chessboard

Problem Statement:

Let's print a chessboard!

Write a program that takes input:
The first line contains T, the number of test cases
Each test case contains an integer N and also the starting character of the chessboard

Output Format
Print the chessboard as per the given examples

Sample Input:
2
2 W
3 B

Sample Output:
WB
BW
BWB
WBW
BWB

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
int main() {
    int T, d, l, i1, i2, o, z;
    char c, s;
    scanf ("%d", &T);
    for (i=0; i<T; i++)
    {
     scanf ("%d %c", &d, &s);
     for (i1 = 0; i1 <d; i1++)
     {
      z = (s == 'W') ? 0:1;
      o = (i1 %2 == z) ? 0:1;
      for(i2 =0; i2 <d; i2 ++)
       {
        c = (i2 %2 == 0) ? 'W' : 'B';
        printf ("%c ", c);
       }
       printf ("\n");
     }
    }
    return 0;
}
```

Output:

Input	Got
2	WB
2 W	BW
2 B	BWB
	WBW
	BWB

Result: C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 30 **Date :**

Pattern Printing

Problem Statement:

Decode the logic and print the Pattern that corresponds to given input.

If N= 3 then pattern will be:
102030100l1012
**4050809
****607
If N= 4, then pattern will be:
10203040170180l9020
**506070140l5016
****809012013
******10011

Constraints: 2 <= N <= 100
Input Format
First line contains T, the number of test cases, each test case contains a single integer N

Output Format
First line print Case #i where i is the test case number, In the subsequent line, print the pattern

Sample Input
3
3
4
5

Sample Output
Case #1
102030100l1012
**4050809
****607
Case #2
10203040170180l9020
**506070140l5016
****809012013
******10011
Case #3
1020304050260270280290 30
**6070809022023024025
****10011012019020021
******13014017018
********15016

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main() {
    int n, v, p3, c, in, i, i1, i2, t, ti;
    scanf("%d", &t);
    for(ti=0; ti<t; ti++){
        v=0;
    scanf("%d", &n);
    printf("case #%d\n", ti+1);
    for(i=0; i<n; i++){
      c=0;
      if(i>0){
          for(i1=0; i1<i; i1++) printf("**");
       }
    for(i1=i; i1<n; i1++){
      if(i>0) c++;
      printf("%d0", ++v);
    }
    if(i==0){
      p3 = v+(v*(v-1))+1;
       in= p3;
    }
    in = in-c;
    p3 = in;
    for(i2=i; i2<n; i2++){
       printf("%d", p3++);
       if(i2!=n-1) printf("0");
    } } printf("\n");
}
```

Result: C-program is Implemented and executed Successfully.

Scanned with OKEN Scanner

Ex. No. : 31 **Date :**

Armstrong Number

Problem Statement:

The k-digit number N is an Armstrong number if and only if the k-th power of each digit sums to N.

Given a positive integer N, return true if and only if it is an Armstrong number.

Note: 1 <= N <= 10^8

Hint: 153 is a 3-digit number, and 153 = 1^3 + 5^3 + 3^3.

Sample Input:
153

Sample Output:
true

Sample **Input**:
123

Sample **Output**:
false

Sample **Input**:
1634

Sample **Output**:
true

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
# include <math.h>
int main () {
    int n;
     Scanf ("%d", &n);
     int x=0 , n2 =n;
while (n2 != 0)
{
   x++;
   n2 = n2/10;
 }
 int sum =0;
 int n3=n , n4;
 while (n3 != 0)
 {  n4 = n3 % 10;
   sum = sum + pow( n4 , x);
   n3 = n3/10;
 }
 if (n == sum)
 {
   printf ("true");
 }
else
 { printf ("false");
 }
  return 0;
 }
```

Output:

Input	Got
3	true
123	false

Result:

C program is Implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 32 Date :

Reverse and Add Until Get a Palindrome

Problem Statement:

Take a number, reverse it and add it to the original number until the obtained number is a palindrome.

Constraints
1<=num<=99999999

Sample Input 1
32

Sample Output 1
55

Sample Input 2
789

Sample Output 2
66066

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int rn, n, nt =0, i=0;
    scanf ("%d", &n);
    do {
        nt = n; rn =0;
    while (n! =0)
    {
      rn = rn * 10 + n%10;
       n = n/10;
    }
    n = nt +rn;
     i++;
}
while (rn ! = nt || i ==1);
printf ("%d", rn);
return 0;
}
```

Output:

Input	Got
32	55
789	66066

Result :

program is implemented and executed succesfully.

Scanned with OKEN Scanner

Ex. No. : 33 Date :

Lucky Number

Problem Statement:

A number is considered lucky if it contains either 3 or 4 or 3 and 4 both in it. Write a program to print the nth lucky number. Example, 1st lucky number is 3, and 2nd lucky number is 4 and 3rd lucky number is 33 and 4th lucky number is 34 and so on. Note that 13, 40 etc., are not lucky as they have other numbers in it.

The program should accept a number 'n' as input and display the nth lucky number as output.

Sample Input 1:
3

Sample Output 1:
33

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int n=1 , i=0, nt, co=0, e ;
    scanf ("%d", &e);
    while (i<e)
    {
      nt=n;
    While (nt !=0)
    {
      co = 0;
      if (nt %10 != 3 && nt%10 != 4 )
      {
          co =1 ;
          break ;
      }
      nt = nt /10;
    }
    if (co ==0)
      {  i++ ;
      }
      n++ ;
    }
    printf ("%d", --n);
    return 0;
}
```

Output:

Input	Got
34	33344

Result : C program is implemented and executed successfully.

Scanned with OKEN Scanner

One-Dimensional Arrays

Scanned with OKEN Scanner

Ex. No. : 31 Date :

Check pair with difference k

Problem Statement:

Given an array A of sorted integers and another non negative integer k, find if there exists 2 indices i and j such that A[i] - A[j] = k, i != j.

Input Format
1. First line is number of test cases T. Following T lines contain:
2. N, followed by N integers of the array
3. The non-negative integer k

Output format
Print 1 if such a pair exists and 0 if it doesn't.

Sample Input:
1
3 1 3 5
4

Sample Output:
1

Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int a[n];
        for(int i=0;i<n;i++){
            scanf("%d",&a[i]);
        }
    int k;
    scanf("%d",&k);
    int flag=0;
    for(int i=0;i<n;i++){
        for(int j=i+1;j<n;j++){
        if(a[i]-a[j]==k||a[j]-a[i]==k)
                {flag=1;break;}
        }
        if(flag)break;}
        printf("%d\n",flag);
    }
}
```

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 35 Date :

Chocolates

Problem Statement:

Sam loves chocolates and starts buying them on the 1st day of the year. Each day of the year, x, is numbered from 1 to Y. On days when x is odd, Sam will buy x chocolates; on days when x is even, Sam will not purchase any chocolates.

Complete the code in the editor so that for each day Ni (where $1 \leq x \leq N \leq Y$) in array arr, the number of chocolates Sam purchased (during days 1 through N) is printed on a new line. This is a function-only challenge, so input is handled for you by the locked stub code in the editor.

Input Format
The program takes an array of integers as a parameter.

The locked code in the editor handles reading the following input from stdin, assembling it into an array of integers (arr), and calling calculate(arr).

The first line of input contains an integer, T (the number of test cases). Each line i of the T subsequent lines describes the ith test case as an integer, Ni (the number of days).

Constraints
$1 \leq T \leq 2 \times 105$
$1 \leq N \leq 2 \times 106$
$1 \leq x \leq N \leq Y$

Output Format
For each test case, Ti in arr, your calculate method should print the total number of chocolates Sam purchased by day Ni on a new line.

Sample Input 0
3
1
2
3

Sample Output 0
1
1
4

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int t ;
    scanf ("%d", &t);
    while (t--) {
        int n, c = 0;
        scanf ("%d", &n);
        for (int i=0; i<=n; i++) {
            if (i%2 != 0)  c = c+i;
        } printf ("%d\n", c);
    }
}
```

Output :

Input	Got
3	1
1	1
2	4
3	

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 36 **Date :**

Football Scores

Problem Statement:

The number of goals achieved by two football teams in matches in a league is given in the form of two lists. Consider:

- Football team A, has played three matches, and has scored { 1 , 2 , 3 } goals in each match respectively.
- Football team B, has played two matches, and has scored { 2, 4 } goals in each match respectively.
- Your task is to compute, for each match of team B, the total number of matches of team A,
- where team A has scored less than or equal to the number of goals scored by team B in that match.

In the above case:

- For 2 goals scored by team B in its first match, team A has 2 matches with scores 1 and 2.
- For 4 goals scored by team B in its second match, team A has 3 matches with scores 1, 2 and 3. Hence, the answer: {2, 3}.

Complete the code in the editor below. The program must return an array of m positive integers, one for each maxes[i] representing the total number of elements nums[j] satisfying nums[j] ≤ maxes[i] where $0 \le j < n$ and $0 \le i < m$, in the given order.

It has the following:

nums[nums[0],...nums[n-1]]: first array of positive integers
maxes[maxes[0],...maxes[n-1]]: second array of positive integers

Constraints:

$2 \le n, m \le 105$, $1 \le nums[j] \le 109$, where $0 \le j < n$, $1 \le maxes[i] \le 109$, where $0 \le i < m$.

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 of elements in nums.
The next n lines each contain an integer describing nums[j] where $0 \le j < n$.
The next line contains an integer m, the number of elements in maxes.
The next m lines each contain an integer describing maxes[i] where $0 \le i < m$.

Sample Input

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

Sample Output

```
2
4
```


Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main() {
    int s1, s2, ans;
    scanf("%d", &s1);
    int ta[s1];
    for(int i=0; i<s1; i++)
    scanf("%d", &ta[i]);
    scanf("%d", &s2);
    int tb[s2];
    for(int i=0; i<s2; i++)
    scanf("%d", &tb[i]);
    for(int j=0; j<s2; j++)
    {
        ans=0;
        for(int i=0; i<s1; i++){
            if(tb[j]>=ta[i])
            ans++;
        }printf("%d\n", ans);
    }
}
```

Output :

Input	Got
4	2
1	
4	4
2	
4	
2	
3	
5	

Result : C program is implemented and executed successfully.

Scanned with OKEN Scanner

Searching Algorithms – Linear and Binary

Scanned with OKEN Scanner

Ex. No. : 37 Date :

Ice Cream Parlor

Problem Statement:

Sunny and Johnny like to pool their money and go to the ice cream parlor. Johnny never buys the same flavor that Sunny does. The only other rule they have is that they spend all of their money.

Given a list of prices for the flavors of ice cream, select the two that will cost all of the money they have.

For example, they have m = 6 to spend and there are flavors costing cost = [1, 2, 3, 4, 5, 6]. The two flavors costing 1 and 5 meet the criteria. Using 1-based indexing, they are at indices 1 and 4.

Complete the code in the editor below. It should return an array containing the indices of the prices of the two flavors they buy, sorted ascending.

It has the following:

- m: an integer denoting the amount of money they have to spend
- cost: an integer array denoting the cost of each flavor of ice cream

Input Format

The first line contains an integer, t, denoting the number of trips to the ice cream parlor. The next t sets of lines each describe a visit. Each trip is described as follows:

1. The integer m, the amount of money they have pooled.
2. The integer n, the number of flavors offered at the time.
3. n space-separated integers denoting the cost of each flavor: cost[cost[1], cost[2], . . . , cost[n]].

Note: The index within the cost array represents the flavor of the ice cream purchased.

Constraints

- $1 \leq t \leq 50$
- $2 \leq m \leq 104$
- $2 \leq n \leq 104$
- $1 \leq cost[i] \leq 104, \forall i \in [1, n]$
- There will always be a unique solution.

Output Format

For each test case, print two space-separated integers denoting the indices of the two flavors purchased, in ascending order.

Sample Input

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

Sample Output 1 4

```
1 2
```


Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int t, m, n, c = 0;
    scanf ("%d", &t);
    for (int i=0 ; i<t; i++) {
    c=0;
    scanf ("%d \n %d", & m, & n);
    int arr [n];
    for (int j=0; j<n; j++) {
        scanf ("%d", & arr [j]);
    }
    for (int a=0 ; a < n-1 ; a++) {
        for (int b=a+1 ; b<n ; b++) {
            if (arr [a] + arr [b] == m) {
            printf ("%d %d \n", a+1 , b+1);
            c=1; break;
            }
        }
    if (c==1) break;
    }
return 0; }
```

Output:

Input	Got
2	
4	1 4
5	1 2
14 5 3 2	
4	
4	
2 2 4 3	

Result : C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 38 Date :

Missing Numbers

Problem Statement:

Numeros the Artist had two lists that were permutations of one another. He was very proud. Unfortunately, while transporting them from one exhibition to another, some numbers were lost out of the first list. Can you find the missing numbers?

As an example, the array with some numbers missing, arr = [7, 2, 5, 3, 5, 3]. The original array of numbers brr = [7, 2, 5, 4, 6, 3, 5, 3]. The numbers missing are [4, 6].

Notes

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- You have to print all the missing numbers in ascending order.
- Print each missing number once, even if it is missing multiple times.
- The difference between maximum and minimum number in the second list is less than or equal to 100.

Complete the code in the editor below. It should return a sorted array of missing numbers. It has the following:

- arr: the array with missing numbers
- brr: the original array of numbers

Input Format

There will be four lines of input:
n - the size of the first list, arr
The next line contains n space-separated integers arr[i]
m - the size of the second list, brr
The next line contains m space-separated integers brr[i]

Constraints

$1 \leq n, m \leq 2 \times 105$, $n \leq m$, $1 \leq brr[i] \leq 2 \times 104$, $Xmax - Xmin < 101$

Output Format

Output the missing numbers in ascending order.

Sample Input

```
10
203 204 205 206 207 208 203 204 205 206
13
203 204 204 205 206 207 205 208 203 206 205 206 204
```

Sample Output

```
204 205 206
```


Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int n, m, c, c1 = 0, co;
    scanf ("%d", &n);
    int arr [n];
    for (int a = 0 ; a < n ; a++) {
    scanf ("%d", & arr [a]);
    }
    scanf ("%d", & m);
    int brr [m], ans [m];
    for (int b = 0 ; b < m ; b++) {
        scanf ("%d", & brr [b]);
    }
    for (int j = 0; j < m ; j++)
    {
    c = 0;
    for (int i = 0 ; i < n ; i++) {
        if (arr [i] == brr [j]) {
            c = 1 ;
            arr [i] = -1 ;
            break ;
        }
    }
    if (c == 0) {
        ans [c1] = brr [j] ;
        c1++ ;
    }
    }
}
```


Scanned with OKEN Scanner

```
for (int a=0 ; a< c1 ; a++) {
    co =0;
     for (int b=0; b<c1 ; b++){
         if (ans [b] < ans [a])
          co++;
      }
      int temp = ans [a];
      ans [a] = ans [co];
      ans[co] = temp;
    }
    for (int i=0 ; i< c1 ; i++ )
    printf ("%d", ans [i]);
    return 0;
}
```

Output : Input

```
10
203 204 205 206 207 208 203
204 205 206
```

Got

```
204 205 206
```

```
13
203 204 204 205 206 207 205
208 203 206 205 206 204
```

Result :

C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 39 **Date :**

Sherlock and Array

Problem Statement:

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right. For instance, given the array arr = [5, 6, 8, 11], 8 is between two subarrays that sum to 11. If your starting array is [1], that element satisfies the rule as left and right sum to 0. You will be given arrays of integers and must determine whether there is an element that meets the criterion.

Complete the code in the editor below. It should return a string, either YES if there is an element meeting the criterion or NO otherwise. It has the following: arr: an array of integers

Input Format
The first line contains T, the number of test cases.
The next T pairs of lines each represent a test case.
- The first line contains n, the number of elements in the array arr.
- The second line contains n space-separated integers arr[i] where $0 \leq i < n$.

Constraints: $1 \leq T \leq 10$, $1 \leq n \leq 105$, $1 \leq arr[i] \leq 2 \times 104$, $0 \leq i \leq n$

Output Format
For each test case print YES if there exists an element in the array, such that the sum of the elements on its left is equal to the sum of the elements on its right; otherwise print NO.

Sample Input 0
2
3
1 2 3
4
1 2 3 3

Sample Output 0
NO
YES

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
int main () {
        int t, n, Is, rs, m;
        scanf ("%d", &t);
        for (int i=0; i<t; i++){
            Is =0;
            rs =0;
            scanf ("%d", &n);
            int arr[n];
            for( int j=0 ; j<n ; j++)
            scanf ("%d", &arr[j]);
            m= n/2;
            if (arr[m]==0){
        for (m=0; arr[m]==0 && m<n ; m++);
            }
            for (int j=0 ; j<=m; j++)
             Is = Is + arr[j];
            for (int j=m; j<n; j++)
            rs = rs + arr[j];
            printf("%s\n", (Is==rs)? "YES" : "NO");
        }
        return 0;
}
```

Output :

Input	Got
2 3 1 2 3 4 1 2 3 3	NO yes

Result : C program is Implemented and executed successfully.

Scanned with OKEN Scanner

Sorting Algorithms – Bubble and Selection

Scanned with OKEN Scanner

Ex. No. : 40 **Date :**

Easy Going

Problem Statement:

Coders here is a simple task for you, you have given an array of size N and an integer M. Your task is to calculate the difference between maximum sum and minimum sum of N-M elements of the given array.

Constraints:
1<=t<=10
1<=n<=1000
1<=a[i]<=1000

Input Format:
First line contains an integer T denoting the number of testcases.
First line of every testcase contains two integer N and M.
Next line contains N space separated integers denoting the elements of array

Output:
For every test case print your answer in new line

Sample Input
1
5 1
1 2 3 4 5

Sample Output
4

Explanation

M is 1 and N is 5 so you have to calculate maximum and minimum sum using (5-1 =) 4 elements.
Maximum sum using the 4 elements would be (2+3+4+5=)14.
Minimum sum using the 4 elements would be (1+2+3+4=)10.
Difference will be 14-10=4.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
   int t;
   scanf ("%d", &t);
   while (t--)
{
  int n, m, d , min, temp;
  scanf ("%d %d", &n, &m);
  d = n-m;
  int arr[n];
  for (int i=0; i<n; i++)
   scanf ("%d", &arr[i]);
   for (int j=0; j<n; j++)
  {
     min=j;
     for (int k=j ; k<n; k++)
     { if (arr[k] < arr [min])
        min = k;
     }
    temp = arr[min];
   arr[min] = arr[j];
   arr[j] = temp;
}int maxsum=0; min sum = 0;
for (int a=0 ; a<d; a++)
```


Scanned with OKEN Scanner

```
minsum += arr[a];
for( int b=n-1; b>m-1; b--)
Maxsum += arr[b];
printf("%d\n", Maxsum - minsum );
}
```

Result:

Thus the C-program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 41 Date :

Sort it out!

Problem Statement:

You are given an array A of non-negative integers of size m. Your task is to sort the array in nondecreasing order and print out the original indices of the new sorted array.

Example:
A={4,5,3,7,1}
After sorting the new array becomes A={1,3,4,5,7}.
The required output should be "4 2 0 1 3"

Input Format:
The first line of input consists of the size of the array
The next line consists of the array of size m

Output Format:
Output consists of a single line of integers

Constraints:
1<=m<=106
0<=A[i]<=106
NOTE: The indexing of the array starts with 0.

Sample Input
5
4 5 3 7 1

Sample Output
4 2 0 1 3

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main (){
    int n;
    scanf ("%d", &n);
    int arr[n];
for (int i=0 ; i<n; i++)
 scanf ("%d", & arr[i]);
 int max = arr [0];
 for (int i=1 ; i<n; i++)
 { if (arr[i] > max)
     max = arr[i];
 }
max ++;
int min =0;
 for (int a=0 ; a<n; a++)
 {
     for (int b=0; b<n; b++)
     {
       if (arr[b] < arr[min])
       min =b;
     }
 printf ("%d ", min);
 arr [min] = max;
 }
}
```

Result : Thus the c program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 42 Date :

Save Patients

Problem Statement:

A new deadly virus has infected large population of a planet. A brilliant scientist has discovered a new strain of virus which can cure this disease. Vaccine produced from this virus has various strength depending on midichlorians count. A person is cured only if midichlorians count in vaccine batch is more than midichlorians count of person. A doctor receives a new set of report which contains midichlorians count of each infected patient. Practo stores all vaccine doctor has and their midichlorians count. You need to determine if doctor can save all patients with the vaccines he has. The number of vaccines and patients are equal.

Input Format
First line contains the number of vaccines - N. Second line contains N integers, which are strength of vaccines. Third line contains N integers, which are midichlorians count of patients.

Output Format
Print a single line containing 'Yes' or 'No'.

Input Constraint
1 < N < 10
Strength of vaccines and midichlorians count of patients fit in integer.

Sample Input
5
123 146 454 542 456
100 328 248 689 200

Sample Output
No

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
 int n, min1, min2, temp, flag = 1;
  scanf ("%d", & n);
 int vac[n], pat[n];
for (int i=0 ; i <n ; i++)
 scanf (" %d", & vac[i]);
 for(int i=0; i<n ; i++)
 scanf ("%d", & pat [i]);
 for (int j =0; j< n-1 ; j++)
 {
    min 1 =j, min 2 = j;
    for (int k =j ; k<n; k++)
    {  if (vac[k] < vac [min1])
       min1 =k;
      if (pat [k] < pat [min 2])
      min2 = k;
    }
  temp = vac [min1];
  vac[min1] = vac [j];
   vac [j] = temp ;
    temp = pat [min 2];
  pat [min 2] = pat [j];
   pat [j] = temp;
}
```

```
for (int i=0 ; i<n; i++)
{
    if (vac[i] <= pat[i])
    {
      flag = 0;
      break;
    }
  }
  if (flag==1)
  printf ("Yes");
  else
  printf ("No");
}
```

Result :

Thus the c program is implemented and Executed successfully.

Scanned with OKEN Scanner

Ex. No. : 43 Date :

Shubham and Xor

Problem Statement:

You are given an array of n integer numbers a1, a2, ... an. Calculate the number of pair of indices (i, j) such that $1 \le i < j \le n$ and ai xor aj = 0.

Input format
- First line: n denoting the number of array elements
- Second line: n space separated integers a1, a2, ... an.

Output format
Output the required number of pairs.

Constraints
$1 \le n \le 10^6$
$1 \le ai \le 10^9$

Sample Input
5
1 3 1 4 3

Sample Output
2

Explanation
The 2 pair of indices are (1, 3) and (3,5)

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main () {
    int n, count =0;
    scanf ("%d", &n);
    int arr [n];
    for (int i=0; i<n ; i++)
    scanf ("%d", & arr [i]);
    for (int i=0 ; i<n-1 ; i++)
     {
       for (int j=i+1 ; j<n; j++)
        {
          if ((arr [i] ^ arr [j]) == =0)
            count ++;
        }
    }
    printf ("%.d", count );
}
```

Result:

Thus the C program is implemented and executed succesfully.

Scanned with OKEN Scanner

Two-Dimensional and Multi-Dimensional Arrays

Scanned with OKEN Scanner

Ex. No. : 44 **Date :**

Add Alternate Elements of 2-Dimensional Array

Problem Statement:

You are given a two-dimensional 3*3 array starting from A [0][0]. You should add the alternate elements of the array and print its sum. It should print two different numbers the first being sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2 and A 0 1, A 1 0, A 1 2, A 2 1.

Input Format
First and only line contains the value of array separated by single space.

A00	A01	A02
4	6	9
A10	**A11**	**A12**
2	5	8
A20	**A21**	**A22**
1	3	7

Output Format
First line should print sum of A 0 0, A 0 2, A 1 1, A 2 0, A 2 2
Second line should print sum of A 0 1, A 1 0, A 1 2, A 2 1

Sample Input
1 2 3 4 5 6 7 8 9

Sample Output
25
20

Scanned with OKEN Scanner

Program:

```
#include <stdio.h>
int main()
{
  int arr[3][3];
  for(int i=0; i<3; i++)
  {
   for(int j=0; j<3; j++)
  {
   scanf("%d", &arr[i][j]);
   }
   }
int  odd=0, even=0;
for(int i=0; i<3; i++)
{
   for(int j=0; j<3; j++)
   {
    if ((i+j)%2 != 0)
      oddt = arr[i][j];
    else
      event = arr[i][j];
    }
}
printf("%d\n%d", even, odd);
}
```

Result : Thus the c program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 45 Date :

The Wealthy Landlord

Problem Statement:

Shyam Lal, a wealthy landlord from the state of Rajasthan, being an old fellow and tired of doing hard work, decided to sell all his farmland and to live rest of his life with that money. No other farmer is rich enough to buy all his land so he decided to partition the land into rectangular plots of different sizes with different cost per unit area. So, he sold these plots to the farmers but made a mistake. Being illiterate, he made partitions that could be overlapping. When the farmers came to know about it, they ran to him for compensation of extra money they paid to him. So, he decided to return all the money to the farmers of that land which was overlapping with other farmer's land to settle down the conflict. All the portion of conflicted land will be taken back by the landlord.

To decide the total compensation, he has to calculate the total amount of money to return back to farmers with the same cost they had purchased from him. Suppose, Shyam Lal has a total land area of 1000 x 1000 equal square blocks where each block is equivalent to a unit square area which can be represented on the co-ordinate axis. Now find the total amount of money, he has to return to the farmers. Help Shyam Lal to accomplish this task.

Input Format:The first line of the input contains an integer N, denoting the total and pieces he had distributed. Next N line contains the 5 space separated integers (X1, Y1), (X2, Y2) to represent a rectangular piece of land, and cost per unit area C.

(X1, Y1) and (X2, Y2) are the locations of first and last square block on the diagonal of the rectangular region.

Output Format:

Print the total amount he has to return to farmers to solve the conflict.

Constraints:

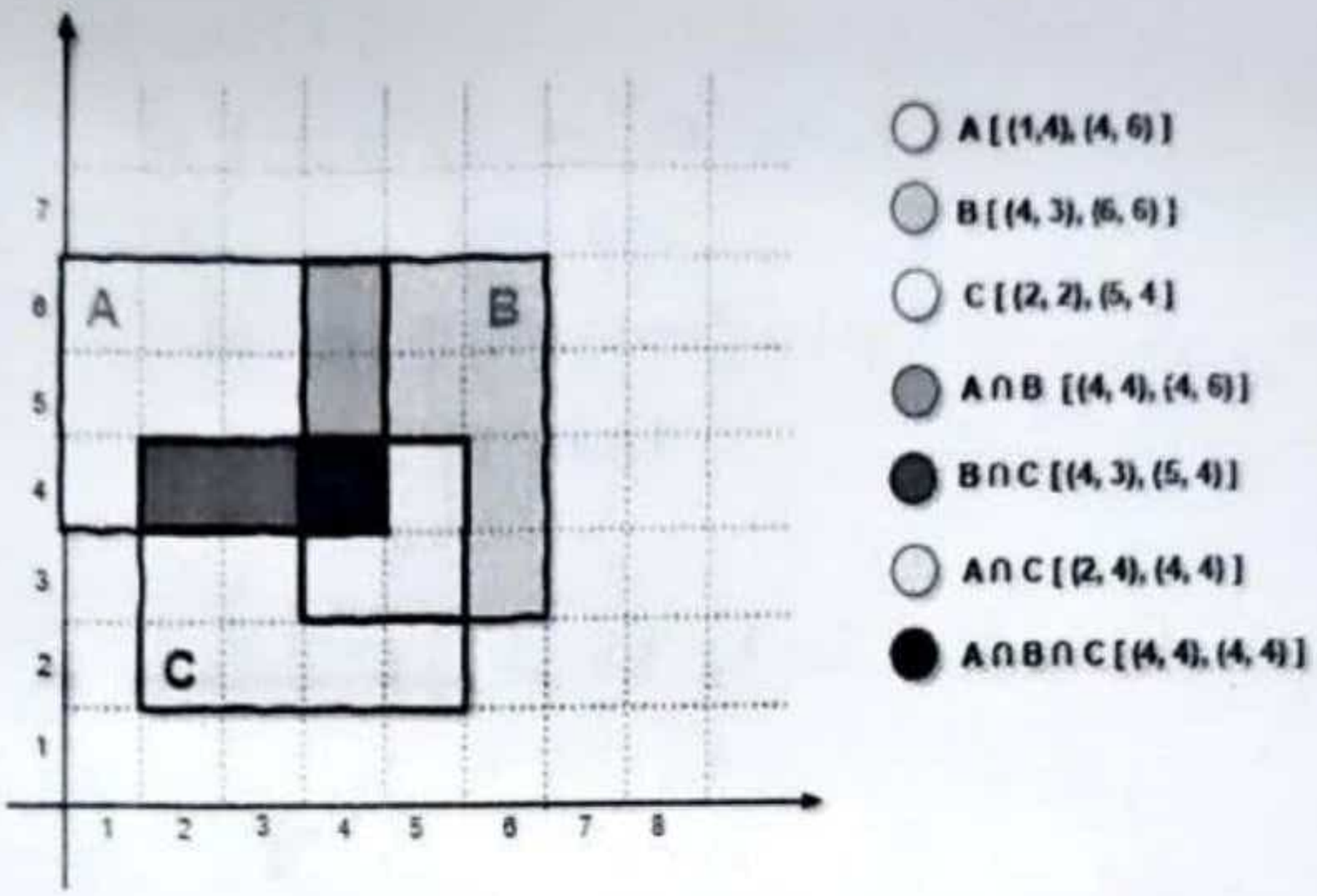
$1 \le N \le 100$
$1 \le X1 \le X2 \le 1000$
$1 \le Y1 \le Y2 \le 1000$
$1 \le C \le 1000$

Sample Input
3
1 4 4 6 1
4 3 6 6 2
2 2 5 4 3

Sample Output
35

Scanned with OKEN Scanner

Explanation



Simple Illustration of Distribution of Land

For given sample input (see given graph for reference), compensation money for different
farmers is as follows:

Farmer with land area A: C1 = 5 * 1 = 5
Farmer with land area B: C2 = 6 * 2 = 12
Farmer with land area C: C3 = 6 * 3 = 18

Total Compensation Money = C1 + C2 + C3 = 5 + 12 + 18 = 35

Scanned with OKEN Scanner

Program:

```
# Include <stdio.h>
int main () {
    int i, j, n, x1, x2, y1, y2, t=0;
     long long total =0;
     int arr [1001][1001] ={0};
      scanf ("%d", &n);
while (n--)
{ scanf ("%d %d %d %d %d", &x1, &x2, &y1,
          & y2, &t );
 for (i=x1; i<=x2; i++)
  {
      for (j=y1 ; j<=y2 ; j++)
      {
        if (arr [i][j] ==0)
        arr [i][j] += t;
        else if (arr [i][j] >0)
        arr [i][j] = (-1) * (arr [i] [j] + t);
        else if (arr [i][j] <0)
        arr [i] [j] -=t ;
  }
}
for (i=1; i<1001 ; i++)
{
```


Scanned with OKEN Scanner

```
for (j=1 ; j< 1001 ; j++)
 {
   if (arr[i][j] <0)
   total += arr[i][j];
 }
} printf ("%.lld\n", (-1) * total );
  return 0;
}
```

Result:

The c-program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : 46 Date :

Priority Interview

Problem Statement:

Microsoft has come to hire interns from your college. N students got shortlisted out of which few were males and a few females. All the students have been assigned talent levels. Smaller the talent level, lesser is your chance to be selected. Microsoft wants to create the result list where it wants the candidates sorted according to their talent levels, but there is a catch. This time Microsoft wants to hire female candidates first and then male candidates. The task is to create a list where first all-female candidates are sorted in a descending order and then male candidates are sorted in a descending order.

Input Format
The first line contains an integer N denoting the number of students. Next, N lines contain two space-separated integers, ai and bi. The first integer, ai will be either 1(for a male candidate) or 0(for female candidate). The second integer, bi will be the candidate's talent level.
Constraints: 1 <= N <= 105, 0 <= ai <= 1, 1 <= bi <= 109

Output Format
Output space-separated integers, which first contains the talent levels of all female candidates sorted in descending order and then the talent levels of male candidates in descending order.

Sample Input
5
0 3
1 6
0 2
0 7
1 15
Sample Output
7 3 2 15 6

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
   struct data
{  int gen; int tal;
};
int main ()
{  int n;
   scanf ("%d", & n);
   struct data a[n];
    for (int i=0; i<n; i++)
    scanf ("%d %d", & a[i]. gen, & a [i]. tal);
    for (int i=0; i< n-1; i++)
    {
        for ( int j=0; j< n-i-1; ++j)
        {
            if (a[j]. tal < a [j+1]. tal)
            {
              struct data temp = a[j];
               a[j] = a[j+1];
              a[j+1]= temp;
            }
        }
}
  for (int i=0; i<n; i++)
{
   if (a[i]. gen == 0)
```


Scanned with OKEN Scanner

```
printf ("%d", a[i].tal);
}
for (int i=0; i<n; ++i)
{ if (a[i].gen==1)
    printf ("%d", a[i].tal);
}
}
```

Result :

Thus the C-program is implemented and executed succesfully.

Scanned with OKEN Scanner

Character Arrays

Scanned with OKEN Scanner

Ex. No. : **Date :**

Strings

Problem Statement:

Input Format
You are given two strings, a and b, separated by a new line. Each string will consist of lower-case Latin characters ('a'-'z').

Output Format
In the first line print two space-separated integers, representing the length of a and b respectively.
In the second line print the string produced by concatenating a and b (a + b).
In the third line print two strings separated by a space, a' and b'. a' and b' are the same as a and b, respectively, except that their first characters are swapped.

Sample Input
abcd
ef

Sample Output
4 2
abcdef
ebcd af

Explanation
a = "abcd"
b = "ef"
|a| = 4
|b| = 2
a + b = "abcdef"
a' = "ebcd"
b' = "af"

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
char str1[10] , str2[10], t ;
int i=0, j=0;
int count1 =0 , count2 =0;
scanf ("%s", str1 );
scanf ( "%s", str2 );
while (str1[i] != '\0')
{ count1 ++ ;
   i++;
 }
while (str2[j] != '\0')
{ count2 ++ ;
   j++;
 }
printf ("%d"%d\n", count1, count2 );
printf (" %s %s\n", str1, str2 );
  t = str1[0];
str1[0] = str2[0];
str2[0] = t;
printf ("%s %s", str1 , str2 );
return 0;
}
```

2.

Result: Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Printing Tokens

Problem Statement:

Given a sentence, s, print each word of the sentence in a new line.

Input Format
The first and only line contains a sentence, s.

Constraints
$1 \leq len(s) \leq 1000$

Output Format
Print each word of the sentence in a new line.

Sample Input
This is C

Sample Output
This
is
C

Explanation
In the given string, there are three words ["This", "is", "C"]. We have to print each of these words in a new line.

Hint
Here, once you have taken the sentence as input, we need to iterate through the input, and keep printing each character one after the other unless you encounter a space. When a space is encountered, you know that a token is complete and space indicates the start of the next token after this. So, whenever there is a space, you need to move to a new line, so that you can start printing the next token.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
  int main ()
{
  char s[1000];
scanf ("%[^\n]s", s);
for (int i=0 ; s[i]! = '\0' ; i++)
{
   if (s[i]! = ' ')
    printf ("%c", s[i]);
   else
    printf ("\n");
  }
  return 0;
}
```

Result :

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Digit Frequency

Problem Statement:

Given a string, s, consisting of alphabets and digits, find the frequency of each digit in the given string.

Input Format
The first line contains a string, num which is the given number.

Constraints
$1 \leq len(num) \leq 1000$
All the elements of num are made of English alphabets and digits.

Output Format
Print ten space-separated integers in a single line denoting the frequency of each digit from 0 to 9.

Sample Input 0
a11472o5t6

Sample Output 0
0 2 1 0 1 1 1 1 0 0

Explanation 0
In the given string:

- 1 occurs two times.
- 2, 4, 5, 6 and 7 occur one time each.
- The remaining digits 0, 3, 8 and 9 don't occur at all.

Hint:

- Declare an array, freq of size 10 and initialize it with zeros, which will be used to count the frequencies of each of the digit occurring.
- Given a string, s, iterate through each of the character in the string. Check if the current character is a number or not.
- If the current character is a number, increase the frequency of that position in the freq array by 1.
- Once done with the iteration over the string, s, in a new line print all the 10 frequencies starting from 0 to 9, separated by spaces.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
    char str [1000];
    scanf ("%s", str);
int hash [10] = {0,0,0,0,0,0,0,0,0,0};
int temp;
for (int i=0; str[i] != '\0'; i++)
{
    temp = str[i] - '0';
    if (temp <=9 && temp >=0)
      { hash [temp] ++;
      }
}
 for ( int i=0; i<=9; i++)
 { printf ("%d ", hash[i]);
 }
 return 0;
}
```

Result :

Thus the c - program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Monk Takes a Walk

Problem Statement:

Today, Monk went for a walk in a garden. There are many trees in the garden and each tree has an English alphabet on it. While Monk was walking, he noticed that all trees with vowels on it are not in good state. He decided to take care of them. So, he asked you to tell him the count of such trees in the garden.

Note: The following letters are vowels: 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o' and 'u'.

Input Format:
The first line consists of an integer T denoting the number of test cases.
Each test case consists of only one string, each character of string denoting the alphabet (may be lowercase or uppercase) on a tree in the garden.

Output Format:
For each test case, print the count in a new line.

Constraints:
$1 \leq T \leq 10$
$1 \leq$ length of string ≤ 105

Sample Input
2
nBBZLaosnm
JHkIsnZtTL

Sample Output
2
1

Explanation
In test case 1, a and o are the only vowels. So, count=2
Brief Description: Given a string S you have to count number of vowels in the string.

Solution 1:
For each vowel, count how many times it is appearing in the string S. Final answer will the sum of frequencies of all the vowels.

Solution 2:
Iterate overall all the characters in the string S and use a counter (variable) to keep track of number of vowels in the string S. While iterating over the characters, if we encounter a vowel, we will increase the counter by 1.

Time Complexity: O(N) where N is the length of the string S. Space Complexity: O(N)

Scanned with OKEN Scanner

Program:

```
# Include <Stdio.h>
 Int main()
{ Int t;
 scanf("%d",&t);
 while (t--)
 {
  Char Str[100000];
  int Count =0;
  Scanf("%s",Str);
  for(Int I=0; Str[i]!='\0'; i++)
   {
   Char C = Str[i];
if((C=='a')||(c=='e')||(C=='i')||(C=='o')||(C=='u')||
   (C=='A')||(C=='E')||(C=='I')||(C=='O')||(C=='U'))
   Count ++;
  }
   printf("%d\n", count);
 }
 return 0;
}
```

Result :

Thus the c program is implemented and executed successfully.

Scanned with OKEN Scanner

String Handling Function

Scanned with OKEN Scanner

Ex. No. : **Date :**

What is your mobile number?

Problem Statement:

These days Bechan Chacha is depressed because his crush gave him list of mobile number some of them are valid and some of them are invalid. Bechan Chacha has special power that he can pick his crush number only if he has valid set of mobile numbers. Help him to determine the valid numbers.

You are given a string "S" and you have to determine whether it is Valid mobile number or not. Mobile number is valid only if it is of length 10 , consists of numeric values and it shouldn't have prefix zeroes.

Input Format:
First line of input is T representing total number of test cases.
Next T line each representing "S" as described in in problem statement.

Output Format:
Print "YES" if it is valid mobile number else print "NO".
Note: Quotes are for clarity.

Constraints:
1<= T <= 103
sum of string length <= 105

Sample Input
3
1234567890
0123456789
0123456.87

Sample Output
YES
NO
NO

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
# include <string.h>
int main () {
    int t;
    scanf ("%d", &t);
    while (t--)
{ int flag = 1;
  char S[100000];
  scanf ("%S", S);
  int k = strlen (S);
   if (K == 10)
   {
      for (int i=0; i<10; i++ )
        {
           if (S[0] == '0')
           {
             flag = 0;
              break;
           }
        if (S[i] < '0' || S[i] > '9')
        { flag = 0;
          break;
        }
      }
}
else
flag = 0;
if (flag == 1)
printf ("YES\n");
```


Scanned with OKEN Scanner

```
    else
     printf ("NO\n");
    }
     return 0;
    }
```

Result :

Thus the C-program is implemented and executed successfully.

Scanned with OKEN Scanner

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.

Scanned with OKEN Scanner

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;
}}}
```


Scanned with OKEN Scanner

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

Result :

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

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

Scanned with OKEN Scanner

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]);
 }
}
```


Scanned with OKEN Scanner

```
Printf ("%s", ans);
return 0;
}
```

Result :

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

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

Scanned with OKEN Scanner

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", word [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 = reverse [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 ;
}
```


Scanned with OKEN Scanner

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

Result:

Thus the C-program is implemented and executed successfully.

Scanned with OKEN Scanner

User-defined Functions & Recursive Functions

Scanned with OKEN Scanner

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.

Scanned with OKEN Scanner

Program:

```
/*
 * Complete the 'pthfactor' function below.
 *
 * The function is expected to return a
  LONG_ INTEGER.
 * The function accepts following parameters:
 * 1. LONG_ INTEGER n
 * 2. LONG_ INTEGER p
 */
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;
}
```

Result : Thus the c program is implemented and executed successfully

Scanned with OKEN Scanner

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.

Scanned with OKEN Scanner

Program:

```
/*
 * Complete the 'fourthBit' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts INTEGER number as
   parameter.
 */
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;
 }
```

Result :

Thus the c-program is implemented and executed successfully.

Scanned with OKEN Scanner

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.

Scanned with OKEN Scanner

Program: */

```
* Complete the 'powerSum' function below.
*
* The function is expected to return an Integer.
* The function accepts following parameters:
* 1. INTEGER x
* 2. INTEGER n
*/
int powerSum (int x, int m, int n)
{
   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 ;
}
```

Result : Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

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

Scanned with OKEN Scanner

Program:

```
int myfunc (int n)
{
  if (n==1)
  return 1;
  if (n %.10==0)
  if (myFunc (n/10)==1)
   return 1;
  if (n %.20 ==0)
 if (myFunc(n/20)==1)
  return 1;
  return 0;
  int t;
  scanf ("%d", &t);
  while (t--)
  {
    int n;
    scanf ("%d ", &n);
     int x = myFunc (n);
     if (x)
     printf ("Yes\n");
     else
     printf ("No\n");
  }
}
```

Result : Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Passing Arrays and Strings to Functions

Scanned with OKEN Scanner

Ex. No. : **Date :**

Balanced Array

Problem Statement:

Given an array of numbers, find the index of the smallest array element (the pivot), for which the sums of all elements to the left and to the right are equal. The array may not be reordered.

Example: arr=[1,2,3,4,6]

- the sum of the first three elements, 1+2+3=6. The value of the last element is 6.
- Using zero based indexing, arr[3]=4 is the pivot between the two subarrays.
- The index of the pivot is 3.

Function Description: Complete the function balancedSum in the editor below.

balancedSum has the following parameter(s): int arr[n]: an array of integers

Returns: int: an integer representing the index of the pivot

Constraints

- $3 \leq n \leq 105$
- $1 \leq arr[i] \leq 2 \times 104$, where $0 \leq i < n$
- It is guaranteed that a solution always exists.

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 size of the array arr. Each of the next n lines contains an integer, arr[i], where $0 \leq i < n$.

Sample Input

STDIN		Function Parameters
4	→	arr[] size n = 4
1	→	arr = [1, 2, 3, 3]
2		
3		
3		

Sample Output 0
2

Explanation 0

- The sum of the first two elements, 1+2=3. The value of the last element is 3.
- Using zero based indexing, arr[2]=3 is the pivot between the two subarrays.
- The index of the pivot is 2.

Scanned with OKEN Scanner

Program:

```
int balancedSum(int arr_count, int* arr)
{
    int totalsum=0, leftsum=0;
    for(int i=0; i<arr_count; i++)
    {
        totalsum+=arr[i];
    }
for(int i=0; i<arr_count; i++)
{   totalsum-=arr[i];
    if(leftsum==totalsum)
    {
        return i;
    }
    leftsum+=arr[i];
}
    return 1;
}
```

Result :

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Sum Them All

Problem Statement:

Calculate the sum of an array of integers.

Example
numbers = [3, 13, 4, 11, 9]
The sum is 3 + 13 + 4 + 11 + 9 = 40.

Function Description
Complete the function arraySum in the editor below.

arraySum has the following parameter(s):
int numbers[n]: an array of integers

Returns
int: integer sum of the numbers array

Constraints
$1 \leq n \leq 104$
$1 \leq numbers[i] \leq 104$

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 size of the array numbers.
Each of the next n lines contains an integer numbers[i] where $0 \leq i < n$.

Sample Input

STDIN		Function
5	→	numbers[] size n = 5
1	→	numbers = [1, 2, 3, 4, 5]
2		
3		
4		
5		

Sample Output
15

Explanation
1 + 2 + 3 + 4 + 5 = 15.

Scanned with OKEN Scanner

Program:

```
int arraySum (int numbers_count, int *numbers)
{
    int sum = 0;
    for (int i=0 ; i< numbers_count ; i++)
      {
       sum+ = numbers[i];
      }
    return sum;
}
```

Result :

Thus the C-program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Minimum Difference Sum

Problem Statement:

Given an array of n integers, rearrange them so that the sum of the absolute differences of all adjacent elements is minimized. Then, compute the sum of those absolute differences.

Example
n = 5, arr = [1, 3, 3, 2, 4]
If the list is rearranged as arr' = [1, 2, 3, 3, 4], the absolute differences are |1 - 2| = 1, |2 - 3| = 1, |3 - 3| = 0, |3 - 4| = 1. The sum of those differences is 1 + 1 + 0 + 1 = 3.

Function Description
Complete the function minDiff in the editor below.

minDiff has the following parameter:
arr: an integer array

Returns:
int: the sum of the absolute differences of adjacent elements

Constraints
$2 \leq n \leq 105$
$0 \leq arr[i] \leq 109$, where $0 \leq i < n$

Input Format For Custom Testing
The first line of input contains an integer, n, the size of arr.
Each of the following n lines contains an integer that describes arr[i] (where $0 \leq i < n$).

Sample Input For Custom Testing

STDIN		Function
5	→	arr[] size n = 5
5	→	arr[] = [5, 1, 3, 7, 3]
1		
3		
7		
3		

Sample Output
6

Explanation
n = 5, arr = [5, 1, 3, 7, 3]
If arr is rearranged as arr' = [1, 3, 3, 5, 7], the differences are minimized.
The final answer is |1 - 3| + |3 - 3| + |3 - 5| + |5 - 7| = 6.

Scanned with OKEN Scanner

Program:

```
int compare ( const void* a , const void * b )
{
  return (* (int *)a - * (int *)b);
 }
int minDiff (int arr_count, int * arr)
{
  qsort (arr, arr_count , sizeof (int), compare);
   int min_ sum = 0;
    for (int i=1 ; i < arr_ count ; i++ )
      {
       min. sum += abs (arr [i] - arr[ i-1] );
      }
        return min_ sum;
    }
```

Result :

Thus the a program is implemented and executed successfully.

Scanned with OKEN Scanner

Structures and Unions

Scanned with OKEN Scanner

Ex. No. : **Date :**

Boxes through a Tunnel

Problem Statement:

You are transporting some boxes through a tunnel, where each box is a parallelepiped, and is characterized by its length, width and height.
The height of the tunnel 41 feet and the width can be assumed to be infinite. A box can be carried through the tunnel only if its height is strictly less than the tunnel's height. Find the volume of each box that can be successfully transported to the other end of the tunnel.

Note: Boxes cannot be rotated.

Input Format
The first line contains a single integer n, denoting the number of boxes.
n lines follow with three integers on each separated by single spaces - $length_i$, $width_i$ and $height_i$ which are length, width and height in feet of the i-th box.

Constraints

$1 \leq n \leq 100$

$1 \leq length_i, width_i, height_i \leq 100$

Output Format
For every box from the input which has a height lesser than 41 feet, print its volume in a separate line.

Sample Input
4
5 5 5
1 2 40
10 5 41
7 2 42

Sample Output
125
80

Explanation
The first box is really low, only 5 feet tall, so it can pass through the tunnel and its volume is 5 x 5 x 5 = 125.

The second box is sufficiently low, its volume is 1 x 2 x 4= = 80.

The third box is exactly 41 feet tall, so it cannot pass. The same can be said about the fourth box.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
int main ()
{
  int n,i;
  int length, width, height;
  scanf ("%d", &n);
   for (i=0; i<n; i++)
{
scanf ("%d %d %d", &length, &width, &height);
 if (height <41)
 {
  printf ("%d\n", length,* width * height);
 }
    }
     return 0;
  }
```

Result:

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Small Triangles, Large Triangles

Problem Statement:

You are given n triangles, specifically, their sides a_i, b_i and c_i. Print them in the same style but sorted by their areas from the smallest one to the largest one. It is guaranteed that all the areas are different.

The best way to calculate a volume of the triangle with sides a, b and c is Heron's formula:

$S = \sqrt{p * (p - a) * (p - b) * (p - c)}$ where $p = (a + b + c) / 2$.

Input Format
First line of each test file contains single integer n. n lines follow with ai, bi and ci on each separated by single spaces.

Constraints
$1 \le n \le 100$
$1 \le a_i, b_i, c_i \le 70$
$a_i + b_i > c_i$, $a_i + c_i > b_i$ and $b_i + c_i > a_i$

Output Format
Print exactly n lines. On each line print 3 integers separated by single spaces, which are ai, bi and ci of the corresponding triangle.

Sample Input
3
7 24 25
5 12 13
3 4 5

Sample Output
3 4 5
5 12 13
7 24 25

Explanation
The square of the first triangle is 84. The square of the second triangle is 30. The square of the third triangle is 6. So, the sorted order is the reverse one.

Scanned with OKEN Scanner

Program:

```
# include <stdio.h>
# include <stdlib.h>
# include <math.h>
struct triangle
{ int a, b, c;
  double area;
};
double calculate area (int a, int b, int c)
{ double p = (a+b+c)/2.0;
  return sqrt(p*(p-a)*(p-b)*(p-c));
}
int compare triangles (const void *x, const void *y)
{ struct triangle *t1 = (struct triangle *)x;
  struct triangle *t2 = (struct triangle *)y;
  return (t1->area > t2->area) - (t1->area < t2->area);
}
int main()
{ int n, i;
  scanf("%d", &n);
  struct triangle triangles[n];
  for (i=0; i<n; i++)
  { scanf("%d%d%d", &triangles[i].a, &triangles[i].b &
      triangles[i].c);
    triangles[i].area = calculatearea (triangles[i].a, triangles
      [i].b, triangles[i].c);
  }
```


Scanned with OKEN Scanner

```
qsort (triangles, n, sizeof(struct triangle), compareTriangles);
 for (i=0; i<n ; i++)
 { printf ("%.d %.d %.d \n", triangles [i].a, triangles[i].b,
                              triangles [i].c);
 }
 return 0;
}
```

Result :

Thus, the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Pointers

Scanned with OKEN Scanner

Ex. No. : **Date :**

Reverse a List

Problem Statement:

Given an array of integers, reverse the given array in place using an index and loop rather than a built-in function.

Example
arr = [1, 3, 2, 4, 5]
Return the array [5, 4, 2, 3, 1] which is the reverse of the input array.

Function Description
Complete the function reverseArray in the editor below.

reverseArray has the following parameter(s):
int arr[n]: an array of integers

Return
int[n]: the array in reverse order

Constraints
$1 \leq n \leq 100$
$0 < arr[i] \leq 100$

Input Format For Custom Testing
The first line contains an integer, n, the number of elements in arr.
Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer, arr[i].

Sample Input For Custom Testing
5
1
3
2
4
5

Sample Output
5
4
2
3
1

Explanation
The input array is [1, 3, 2, 4, 5], so the reverse of the input array is [5, 4, 2, 3, 1].

Scanned with OKEN Scanner

Program:

```
int * reverseArray (int arr_ count, int * arr, int *
                    result_ count)
{
  * result_ count = arr_ count ;
  int * reversedarray = (int *) malloc (arr_ count *
                        size of (int));
  for (int  i =0 ; i< arr_ count  ; i++)
{
  reversed array [i] = arr [arr- Count -1 -i ];
 }
 return  reversedarray;
}
```

Result :

Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner

Ex. No. : **Date :**

Cut Them All

Problem Statement:

An automated cutting machine is used to cut rods into segments. The cutting machine can only hold a rod of minLength or more, and it can only make one cut at a time. Given the array lengths [] representing the desired lengths of each segment, determine if it is possible to make the necessary cuts using this machine. The rod is marked into lengths already, in the order given.

Example
n = 3
lengths = [4, 3, 2]
minLength = 7

The rod is initially sum(lengths) = 4 + 3 + 2 = 9 units long. First cut off the segment of length 4 + 3 = 7 leaving a rod 9 - 7 = 2. Then check that the length 7 rod can be cut into segments of lengths 4 and 3. Since 7 is greater than or equal to minLength = 7, the final cut can be made. Return "Possible".

Example
n = 3
lengths = [4, 2, 3]
minLength = 7

The rod is initially sum(lengths) = 4 + 2 + 3 = 9 units long. In this case, the initial cut can be of length 4 or 4 + 2 = 6. Regardless of the length of the first cut, the remaining piece will be shorter than minLength. Because n - 1 = 2 cuts cannot be made, the answer is "Impossible".

Function Description

Complete the function cutThemAll in the editor below.

cutThemAll has the following parameter(s):

int lengths[n]: the lengths of the segments, in order
int minLength: the minimum length the machine can accept

Returns

string: "Possible" if all n-1 cuts can be made. Otherwise, return the string "Impossible".

Constraints

- $2 \leq n \leq 105$
- $1 \leq t \leq 109$
- $1 \leq lengths[i] \leq 109$
- The sum of the elements of lengths equals the uncut rod length.

Scanned with OKEN Scanner

Input Format For Custom Testing

The first line contains an integer, n, the number of elements in lengths.
Each line i of the n subsequent lines (where $0 \leq i < n$) contains an integer, lengths[i].
The next line contains an integer, minLength, the minimum length accepted by the machine.

Sample Input For Custom Testing

STDIN		Function
-----		--------
4	→	lengths[] size n = 4
3	→	lengths[] = [3, 5, 4, 3]
5		
4		
3		
9	→	minLength= 9

Sample Output
Possible

Explanation

The uncut rod is 3 + 5 + 4 + 3 = 15 units long. Cut the rod into lengths of 3 + 5 + 4 = 12 and 3.
Then cut the 12-unit piece into lengths 3 and 5 + 4 = 9.
The remaining segment is 5 + 4 = 9 units and that is long enough to make the final cut.

Scanned with OKEN Scanner

Program:

```
char * CutThemAll (int lengths_count, long * lengths,
                   long minLength)
{
  long sum = 0
  for (int i=0; i< lengths_count; i++)
  {
    sum+ = lengths [i];
  }
for (int i=0; i< lengths_ count ; i++)
{   if (sum < min Length)
    { break;
    }
    sum - = lengths [i];
    if ( sum >= minLength)
    {  static char result [] = "possible";
       return result;
    }
}
static char result [] = "Impossible";
return result;
}
```

Result: Thus the C program is implemented and executed successfully.

Scanned with OKEN Scanner