1. Write a C program to print prime numbers between the given ranges (both boundaries inclusive) and also display largest prime number in the series.

Note: save the file/class as “primeNo” with appropriate extension

**Input:**

The first line of each test case is lower and upper limit, L and U which are positive integers, on the same line separated by a single space respectively.

**Output:**

For each test case, the output should be the list of prime numbers lying between lower and upper limit (both boundaries inclusive) followed by the largest prime number in the series.

**Constraints:**

0<L<U<=1000

**Example:**

**Input:**1 10  
11 20

**Output:**

2 3 5 7 **7**  
  
11 13 17 19 **19**

2. Generate first N terms of Fibonacci series.

**Note:** save the file/class as “fibonacci” with appropriate extension

**Input:**

The first line of each test case is first N terms of Fibonacci series.

**Output:**

For each test case, there is one line of output i.e. first N terms of Fibonacci series.

**Constraints:**

5<=N<=100

**Example:**

**Input:**

5

10

**Output:**

0 1 1 2 3

0 1 1 2 3 5 8 13 21 34

3. Check whether the given number is an Armstrong number or not

**Note:** save the file/class as “**armstrongNo**” with appropriate extension

**Note:** It should accept number upto 4-digits.

**Input:**

The first line of each test case is a number N upto 4 digit.

**Output:**

For each test case, there is one line of output i.e. whether it is Armstrong number or not an Armstrong number.

**Constraints:**

1<=N<=9999

**Example**

**Input:**

370

1500

**Output:**

Armstrong number

Not an Armstrong number

4. Write a C-program to check whether the given number is a Trendy number or not

**Note:** save the file/class as “trendyNo” with appropriate extension

**Note:** In a 3 digit number if middle number divisible by 3 then it is trendy number.

**Input:**

The first line of each test case is a number N with 3 digits.

**Output:**

For each test case, there is one line of output i.e. whether it is Trendy number or not a Trendy number.

**Constraints:**

100<=N<=999

**Example**

**Input:**

464

675

**Output:**

Trendy Number

Not a trendy Number

5. Write a C-program to check whether the given number is Special number or not.

**Note:** save the file/class as “specialNo” with appropriate extension

**Note:** split no -->Add and multiply the no and add it to get the same number

**Example:** if the input is 19 ====>>1+9 = 10 and 1\*9=9 =====>> 10+9=19 which is equal to the input so the given number is a special number.

**Input:**

The first line of each test case is a number N.

**Output:**

For each test case, there is one line of output i.e. whether it is Special number or not a special number.

Constraints:

10<=N<=99

**Example**

**Input:**

19

11

**Output:**

Special Number

Not a Special Number

6. Write a C-program to check whether the given string is Palindrome or not.

**Note:** save the file/class as “palindrome” with appropriate extension

**Input:**

The first line of each test case is a String S.

**Output:**

For each test case, there is one line of output i.e. whether it is Palindrome or not a palindrome.

**Constraints:**

2<=Length of String S <=50

**Example**

**Input:**

malayalam

hello

**Output:**

Palindrome

Not a Palindrome

7. Find LCM and GCD of two numbers

**Note:** save the file/class as “lcmGcd” with appropriate extension

**Input:**

The first line of each test case is two integers, N1 and N2 which are positive integers, on the same line separated by a single space respectively.

**Output:**

For each test case, there are two lines of output. First line is LCM of two given integers N1 and N2. Second line is the GCD of two given integers N1 and N2.

**Constraints:**

0<N1, N2<=1000

**Input:**

24 18

**Output:**

72

6

8. Check whether the given number is perfect square or not.

Note: save the file/class as “perfectSquare” with appropriate extension

**Input:**

The first line of each test case is a number N.

**Output:**

For each test case, there is one line of output i.e. whether it is Perfect square or not a Perfect square.

**Constraints:**

1<=N<=1000

**Example**

**Input:**

64

11

**Output:**

Perfect square

Not a perfect square

9. C program to compare if two strings are equal or not

Note: save the file/class as “compareStrings” with appropriate extension

**Input:**

The test case is two input strings S1 on the first line and S2 on the second line.

**Output:**

For each test case, there is one line of output i.e. whether the Strings are equal or Strings are not equal

**Constraints:**

S1 and S2 length may not exceed 50 characters.

**Input:**

**Example**

codevita

codevita

Hackerrank

Hackerearth

**Output:**

Strings are equal

Strings are not equal

10. Print the following pattern

**Note:** save the file/class as “pattern” with appropriate extension

**Input:**

The first line of each test case is a number of rows N.

**Output:**

For each test case, there is an output which generates the given pattern with the given number of rows.

**Constraints:**

1<=N<=99

**Example**

**Input:**

3

4

**Output:**

1  
2 3  
4 5 6

1  
2 3  
4 5 6

7 8 9 10

11. Write a C program to input amount from user and print minimum number of notes (Rs. 500, 100, 50, 20, 10, 5, 2, and 1) required for the amount. How to the minimum number of notes required for the given amount in C programming. Program to find minimum number of notes required for the given denomination. Logic to find minimum number of denomination for a given amount in C program.

Note: save the file/class as “minDenomination” with appropriate extension

**Input:**

The first line of each test case is an amount N.

**Output:**

For each test case, there is an output i.e. minimum number of denomination for a given amount.

**Constraints:**

1<=N<=10000

**Example**

**Input:**

575

**Output:**

500: 1

100: 0

50: 1

20: 1

10: 0

5: 1

2: 0

1: 0

12. Write a C program to find sum of digits

Note: save the file/class as “sumOfDigits” with appropriate extension

Input:

The first line of each test case is a number N whose sum of digits has to be calculated.

Output:

For each test case, there is one line of output i.e. sum of digits of a given number.

Constraints:

2<=N<=100000

Example

Input:

12345

123

Output:

15

6

13. Finding grade of a student

Note: save the file/class as “gradeOfStud” with appropriate extension

0-40 U  
41-50 D  
51-60 C  
61-70 B+  
71-80 B  
81-90 A  
91-100 S

Input:

The first line of each test case is an integer N.

Output:

For each test case, there is one line of output i.e. grade of a student with respective to the given question.

Constraints:

0<=N<=100

Example

Input:

89

32

Output:

A

U

14. Write a C-Program to find the factorial of a given number.

Note: save the file/class as “factorialNo” with appropriate extension

Input:

The first line of each test case is a integer N.

Output:

For each test case, there is one line of output i.e. factorial of a given number.

Constraints:

0<=N<=1000

Example

Input:

5

3

Output:

120

6

15.Write a c program if given number is divisible by 3, print "foo" , if it divisible by 5, print "Bar", if it is divisible by 3 and 5, print "FooBar" and if it is not divisible by 3 and 5,print "None".

**Note:** save the file/class as “fooBar” with appropriate extension

**Input:**

The first line of each test case is an integer N

**Output:**

For each test case, there is one line of output i.e. Foo or Bar or FooBar or None.

**Constraints:**

0<=N<=1000

**Example**

**Input:**

21

25

30

4

**Output:**

Foo

Bar

FooBar

None