

# Numbers And Programming

Collected by Rudra Kaiser

Reference : Google, Wikipedia, ChatGPT, YouTube

A handwritten signature in black ink that reads "Rudra Kaiser". The signature is stylized with a large, sweeping 'R' and a long horizontal line extending from the end of the name.

## PRIME NUMBERS

গণিতের পরিভাষায় প্রাইম নাম্বার বা মৌলিক সংখ্যা হল এমন স্বাভাবিক সংখ্যা যার কেবল দুটো পৃথক উৎপাদক আছে এবং তা হল ১ এবং ঐ সংখ্যাটি নিজে। ১ এর চেয়ে বড় যে সকল সংখ্যা মৌলিক না তাদেরকে যৌগিক সংখ্যা বলে। অর্থাৎ যে সংখ্যাকে শুধুমাত্র ১ এবং নিজ সংখ্যা ছাড়া অন্য কোন সংখ্যা দ্বারা ভাগ করা যায় না, সেই সকল সংখ্যাই মূলত মৌলিক সংখ্যা / Prime Number। মনেকরি, স্বাভাবিক সংখ্যা N একটি মৌলিক সংখ্যা, অর্থাৎ এই সংখ্যা কেবল ১ এবং N ছাড়া অন্য কোনো সংখ্যা দ্বারা নিঃশেষে বিভাজ্য হবে না। (যেমন ২, ৩, ৫, ৭, ১১, ১৩, ১৭, ১৯ ... ইত্যাদি মৌলিক সংখ্যা)।

### Prime Numbers

Prime 1	Prime 2	Prime 3	Prime 4	Prime 5	Prime 6	Prime 7	Prime 8
2	3	5	7	11	13	17	19
Prime 9	Prime 10	Prime 11	Prime 12	Prime 13	Prime 14	Prime 15	Prime 16
23	29	31	37	41	43	47	53
Prime 17	Prime 18	Prime 19	Prime 20	Prime 21	Prime 22	Prime 23	Prime 24
59	61	67	71	73	79	83	89

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a prime number or not, if the number 'N' is given by the user. Print "The number N is a Prime Number" if the number is a prime number or print "The number N is Not a Prime Number" if the number is not a prime number.

**Problem :** Write a C / C++ / Java / Python program to all the prime numbers between 200 and 500.

## PERFECT NUMBERS

সংখ্যা তত্ত্বে, একটি পারফেক্ট নাম্বার বা নিখুঁত সংখ্যা হল একটি ধনাত্মক পূর্ণসংখ্যা যা তার ধনাত্মক সঠিক ভাজকের যোগফলের সমান (এখানে "সঠিক ভাজক" বলতে বোঝানো হয়েছে এমন ভাজক যা ঐ সংখ্যাটি নিজে নয়)। সহজভাবে বলতে গেলে, ধরি কোনো সংখ্যা  $N$  হল নিখুঁত সংখ্যা। এখন  $N$  এর উৎপাদক সমূহ নির্ণয় করতে গেলে তন্মধ্যে  $N$  নিজেও থাকবে, যেহেতু  $N$  নিখুঁত সংখ্যা, আমরা উৎপাদক সমূহের মধ্যে  $N$  কে উপেক্ষা করে বাকি সকল উৎপাদকের যোগফল  $N$  এর সমান পাবো। উদাহরণস্বরূপ, আমরা একটি সংখ্যা নিলাম ৬। ৬ এর উৎপাদক সমূহ ১, ২, ৩ এবং ৬। এখানে যেহেতু আমরা সংখ্যা হিসেবে ৬ কে নিয়েছি তাই এই সংখ্যার সাথে মিল উৎপাদক ৬ কে বাদ দিলে বাকি থাকে ১, ২, এবং ৩। এখন, এই সকল উৎপাদকের যোগফল হয়  $1+2+3 = 6$  এবং আমরা যে সংখ্যা নিয়েছিলাম সেটাও ৬, সুতরাং ৬ একটি নিখুঁত সংখ্যা। (পরবর্তী নিখুঁত সংখ্যা ২৮)।

6 এবং 28 কি পারফেক্ট নাম্বার ?

1	2	3	6
✓	✓	✓	×

$$= 1 + 2 + 3$$

$$= 6$$

Perfect Number

1	2	4	7	14	28
✓	✓	✓	✓	✓	×

$$= 1 + 2 + 4 + 7 + 14$$

$$= 28$$

Perfect Number

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a perfect number or not, if the number 'N' is given by the user. Print "The number N is a Perfect Number" if the number is a perfect number or print "The number N is Not a Perfect Number" if the number is not a perfect number.

**Problem :** Write a C / C++ / Java / Python program to print all the perfect numbers between 1 and 500.

## STRONG NUMBERS

স্ট্রং নাম্বার হল একটি ধনাত্মক পূর্ণসংখ্যা যার ডিজিট গুলোর ফ্যাক্টোরিয়ালের যোগফলের সমান সেই সংখ্যা নিজে। অন্য কথায়, যদি আমরা একটি সংখ্যার অঙ্কগুলি গ্রহণ করি এবং প্রতিটি অঙ্কের ফ্যাক্টোরিয়াল গণনা করে তারপর সেগুলোর যোগফল নির্ণয় করি এবং যোগফল যদি আসল সংখ্যার সমান হয়, তবে সেই সংখ্যাটিকে স্ট্রং নাম্বার বলা হয়। যেমন 145 একটি স্ট্রং নাম্বার কারণ, এই সংখ্যার মধ্যে ডিজিট গুলো হল 1, 4 এবং 5। এখন আমরা যদি এই 1, 4 এবং 5 প্রত্যেকের ফ্যাক্টোরিয়াল বের করে তারপর তাদের যোগফল নির্ণয় করি তবে যোগফল হবে 145 [ $1! + 4! + 5! = 1 + 24 + 120 = 145$ ]।

$$\begin{array}{c} 145 \\ \\ 1! + 4! + 5! \\ 1 + 24 + 120 \\ \\ 145 \end{array}$$

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a strong number or not, if the number 'N' is given by the user. Print "The number N is a Strong Number" if the number is a strong number or print "The number N is Not a Strong Number" if the number is not a strong number.

**Problem :** Write a C / C++ / Java / Python program to print all the strong numbers between 1 and 1000.

## ARMSTRONG NUMBERS

আর্মস্ট্রং নাম্বার হল এমন একটি সংখ্যা, যার মোট ডিজিটগুলোর পাওয়ারের সমষ্টি সেই সংখ্যা হয় এবং পাওয়ারের মান হবে ঐ সংখ্যায় যতগুলো ডিজিট থাকবে তত। উদাহরণস্বরূপ, নাম্বারটি 3 ডিজিটের হলে Power নেওয়া হবে 3, 4 টি ডিজিটের হলে Power নেওয়া হবে 4। যেমন, 153 একটি আর্মস্ট্রং নাম্বার কারণ এর তিনটি ডিজিট যথাক্রমে 1, 5, এবং 3। এখানে যেহেতু 3টি ডিজিট আছে, তাই প্রতিটি ডিজিটের কিউব নেওয়া হয়েছে :  $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ ।

$$153 \Rightarrow 1^3 + 5^3 + 3^3 \Rightarrow 1 + 125 + 27 \Rightarrow 153$$

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is an armstrong number or not, if the number 'N' is given by the user. Print "The number N is an Armstrong Number" if the number is an armstrong number or print "The number N is Not an Armstrong Number" if the number is not an armstrong number.

**Problem :** Write a C / C++ / Java / Python program to print all the armstrong numbers between 1 and 1000.

## FIBONACCI NUMBERS

ফিবোনাচ্চি সিরিজ হল একটি সংখ্যা সিকোয়েন্স যা 0 এবং 1 দিয়ে শুরু হয়, এবং এর প্রতিটি সংখ্যাকে পূর্ববর্তী দুটি সংখ্যার যোগফলের মাধ্যমে নির্ণয় করা হয়। এই সিরিজের প্রতিটি সংখ্যা ফিবোনাচ্চি নাম্বার নামে পরিচিত। ফিবোনাচ্চি সিরিজ : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 ...।

### First 10 Fibonacci Numbers

F <sub>1</sub>	F <sub>2</sub>	F <sub>3</sub>	F <sub>4</sub>	F <sub>5</sub>	F <sub>6</sub>	F <sub>7</sub>	F <sub>8</sub>	F <sub>9</sub>	F <sub>10</sub>
0	1	1	2	3	5	8	13	21	34

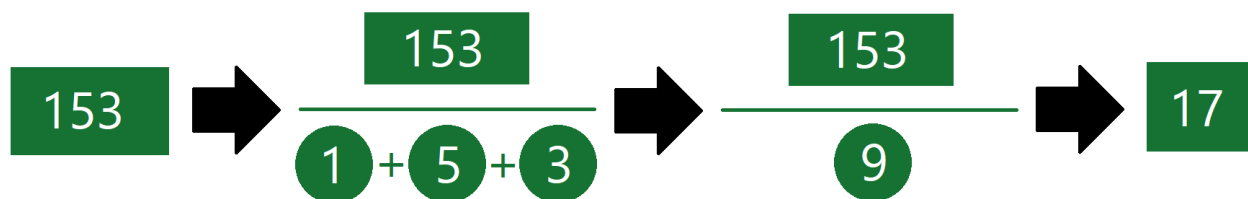
**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a fibonacci number or not, if the number 'N' is given by the user. Print "The number N is a Fibonacci Number" if the number is a fibonacci number or print "The number N is Not a Fibonacci Number" if the number is not a fibonacci number.

**Problem :** Write a C / C++ / Java / Python program to print all the fibonacci numbers between 1 and 1000.

## HARSHAD (or NIVEN) NUMBERS

হারশাদ নাম্বার হল এমন একটি সংখ্যা, যাকে ঐ সংখ্যার ডিজিটসমূহের যোগফল দ্বারা ভাগ করলে একটি পূর্ণসংখ্যা পাওয়া যায়। উদাহরণস্বরূপ, 153 একটি হারশাদ নাম্বার, কারণ :  $153 \div (1+5+3) = 153 \div 9 = 17$ । এখানে, 153 কে এর ডিজিটসমূহের যোগফল ( $1+5+3 = 9$ ) দ্বারা ভাগ করার ফলে ভাগফল 17 পাওয়া যায়, যা একটি পূর্ণসংখ্যা (Integer Number)। এছাড়াও, হারশাদ নাম্বারগুলোর মধ্যে অন্যান্য উদাহরণ হিসেবে 18, 24, 36 ইত্যাদি অন্তর্ভুক্ত।

**153 is a Harshad Number**




**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a harshad number or not, if the number 'N' is given by the user. Print "The number N is a Niven Number" if the number is a harshad number or print "The number N is Not a Niven Number" if the number is not a harshad number.


**Problem :** Write a C / C++ / Java / Python program to print all the harshad numbers between 100 and 200.

## PALINDROMIC NUMBERS

**প্যালিনড্রমিক নাম্বার** হল এমন একটি সংখ্যা, যাকে বিপরীত করলে সেই সংখ্যা অপরিবর্তিত থাকে। অন্য কথায়, সংখ্যার কোনও পরিবর্তন না ঘটলে সেটিকে প্যালিনড্রমিক নাম্বার বলা হয়। যেমন 13732 এই সংখ্যাকে বিপরীত করলে হবে 23731, এখানে যেহেতু রিভার্স করার পর সংখ্যার পরিবর্তন হয়েছে তাই এটি প্যালিনড্রমিক নাম্বার নয়। আবার, 13731 এই সংখ্যাকে বিপরীত করলে হবে 13731, এখানে যেহেতু বিপরীত বা রিভার্স করার পরেও সংখ্যার কোনো পরিবর্তন হয়নি তাই এটি একটি প্যালিনড্রমিক নাম্বার।

### Example of Palindromic Numbers

 74347 ————— 74347 ————— Palindromic Number

 13625 ————— 52631 ————— Not Palindromic Number

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a palindromic number or not, if the number 'N' is given by the user. Print "The number N is a Palindromic Number" if the number is a palindromic number or print "The number N is Not a Palindromic Number" if the number is not a palindromic number.

**Problem :** Write a C / C++ / Java / Python program to print all the palindromic numbers between 100 and 200.

## HAPPY & UNHAPPY NUMBERS

কোনো নাম্বারকে বারবার তার ডিজিটের স্কয়ার করার পরে সব ডিজিটের যোগফল নির্ণয় করতে থাকা যতক্ষণ না পর্যন্ত যোগফল একটি মাত্র সংখ্যা 1 আসে, যদি যোগফল 1 আসে তাহলে তা হ্যাপি নাম্বার আর যদি যোগফল 4 আসে তাহলে আন-হ্যাপি নাম্বার। যেমন 97 এই নাম্বারের জন্যে হ্যাপি নাকি আন-হ্যাপি যদি তা চেক করা হয় তবে 97 এর ডিজিটসমূহ 9 এবং 7 সুতরাং,  $9^2 + 7^2 = 81 + 49 = 130$  আবার, 130 এর ডিজিটসমূহ 1, 0 এবং 3 সুতরাং,  $1^2 + 0^2 + 3^2 = 1 + 0 + 9 = 10$  এখন, 10 এর ডিজিটসমূহ 1 এবং 0 সুতরাং,  $1^2 + 0^2 = 1$  (শেষ অবধি 1 এলো) অর্থাৎ 97 হলো হ্যাপি নাম্বার। আবার, 116 এই নাম্বারের জন্যে হ্যাপি নাকি আন-হ্যাপি যদি তা চেক করা হয় তবে 116 এর ডিজিটসমূহ 1, 1 এবং 6 সুতরাং,  $1^2 + 1^2 + 6^2 = 1 + 1 + 36 = 38$  আবার, 38 এর ডিজিটসমূহ 3 এবং 8 সুতরাং,  $3^2 + 8^2 = 9 + 64 = 73$  এখন, 73 এর ডিজিটসমূহ 7 এবং 3 সুতরাং,  $7^2 + 3^2 = 49 + 9 = 58$  এখন, 58 এর ডিজিটসমূহ 5 এবং 8 সুতরাং,  $5^2 + 8^2 = 25 + 64 = 89$  আবার, 89 এর ডিজিটসমূহ 8 এবং 9 সুতরাং,  $8^2 + 9^2 = 64 + 81 = 145$  এখন, 145 এর ডিজিটসমূহ 1, 4 এবং 5 সুতরাং,  $1^2 + 4^2 + 5^2 = 1 + 16 + 25 = 42$  আবার, 42 এর ডিজিটসমূহ 4 এবং 2 সুতরাং,  $4^2 + 2^2 = 16 + 4 = 20$  এখন, 20 এর ডিজিটসমূহ 2 এবং 0 সুতরাং,  $2^2 + 0^2 = 4 + 0 = 4$  (শেষ অবধি 4 এলো) অর্থাৎ 116 হলো আন-হ্যাপি নাম্বার।

### Happy Number 😊

$$97 = 9^2 + 7^2 = 130 = 1^2 + 3^2 + 0^2 = 10 = 1^2 + 0^2 = 1$$

### Unhappy Number 😞

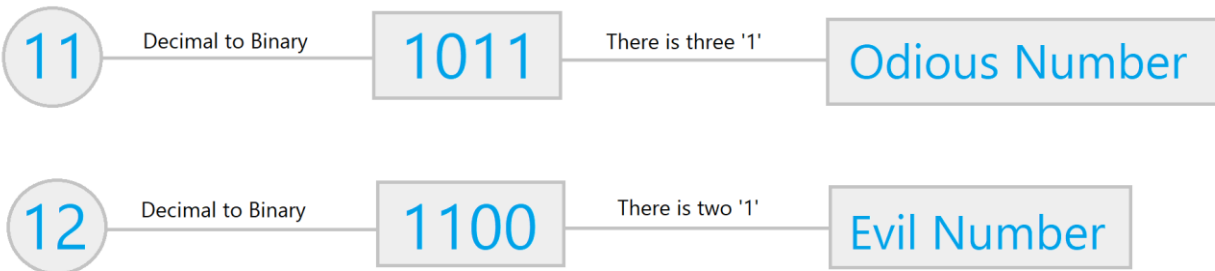
$$145 = 1^2 + 4^2 + 5^2 = 42 = 4^2 + 2^2 = 20 = 2^2 + 0^2 = 4$$

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a happy or unhappy number, if the number 'N' is given by the user. Print "The number N is a Happy Number" if the number is a happy number or print "The number N is an Unhappy Number" if the number is an unhappy number.



## EVIL & ODIUS NUMBERS

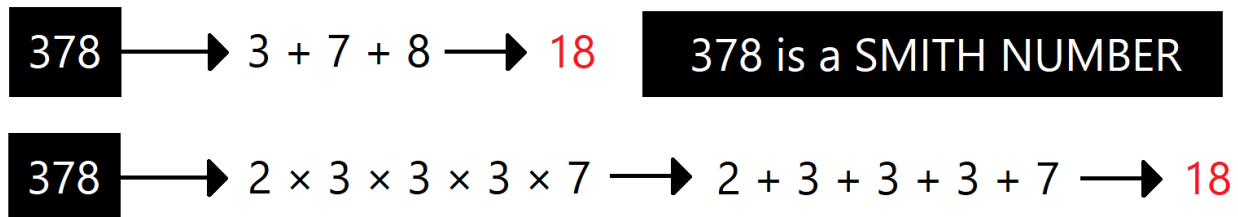
কোনো ডেসিমেল সংখ্যাকে বাইনারি তে রূপান্তর করার পরে যদি, সেই বাইনারি নাম্বারের মধ্যে জোড় সংখ্যক 1 থাকে তাহলে সেই ডেসিমেল নাম্বারকে Evil Number বলা হয় আর যদি, বিজোড় সংখ্যক 1 থাকে তাহলে সেই ডেসিমেল নাম্বারকে Odious Number বলা হয়। যেমন 5 এর বাইনারি সংখ্যা হল 100। যেহেতু 5 এর বাইনারিতে একটি মাত্র '1' রয়েছে (অর্থাৎ বিজোড় সংখ্যক 1) তাই এটি একটি Odious Number আবার, 6 এর বাইনারি সংখ্যা হল 101। যেহেতু 6 এর বাইনারিতে দুইটি '1' রয়েছে (অর্থাৎ জোড় সংখ্যক 1) তাই এটি একটি Evil Number।



**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is an evil or odious number, if the number 'N' is given by the user. Print "The number N is an Evil Number" if the number is an evil number or print "The number N is Not an Odious Number" if the number is an odious number.

## SMITH NUMBERS

যদি কোনো সংখ্যার ডিজিটগুলোর যোগফল ঐ সংখ্যার উৎপাদক গুলোর যোগফলের সমান হয় এবং উৎপাদকগুলো অবশ্যই মৌলিক সংখ্যা হতে হবে তাহলে ঐ সংখ্যাকে স্মিথ নাম্বার বলা হবে।



**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a smith number or not, if the number 'N' is given by the user. Print "The number N is a Smith Number" if the number is a smith number or print "The number N is Not a Smith Number" if the number is not a smith number.

## SPHENIC NUMBERS

Sphenic Number হল এমন একটি সংখ্যা, যার তিনটি উৎপাদক মৌলিক সংখ্যা বা প্রাইম নাম্বার। অর্থাৎ, যদি একটি সংখ্যা তিনটি আলাদা মৌলিক সংখ্যা গুনফলে তৈরি হয়, তবে সেই সংখ্যাকে Sphenic Number বলা হয়। উদাহরণস্বরূপ, 30 একটি Sphenic Number, কারণ এর উৎপাদক মৌলিক সংখ্যা গুলি হল :  $30 = 2 \times 3 \times 5$  এখানে 2, 3, এবং 5 সকলেই মৌলিক সংখ্যা।

### Checking 30, 42 and 78 For SPHENIC NUMBERS

$$30 = 2 \times 3 \times 5$$

2, 3 and 5 are prime numbers

$$42 = 2 \times 3 \times 7$$

2, 3 and 7 are prime numbers

$$78 = 2 \times 3 \times 13$$

2, 3 and 13 are prime numbers

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a sphenic number or not, if the number 'N' is given by the user. Print "The number N is a Sphenic Number" if the number is a sphenic number or print "The number N is Not a Sphenic Number" if the number is not a sphenic number.

**Problem :** Write a C / C++ / Java / Python program to print all the sphenic number between 10 and 200.

## KAPREKAR'S CONSTANT 495 & 6174

495 এবং 6174 এই দুটি নাম্বার হলো কাপ্রিকারের ম্যাজিক্যাল নাম্বার এছাড়াও এদের Kaprekar's Constant ও বলা হয়ে থাকে। Kaprekar's Constant এর কনসেপ্ট আরও ভালো ভাবে বুঝার জন্য আমরা কিছু উদাহরণ খেয়াল করি। আমরা তিন ডিজিটের যেকোনো একটা নাম্বার কল্লনা করি, যেমন 627। এই নাম্বারের ডিজিটগুলো হলো 6, 2 এবং 7 এখন, এই ডিজিটগুলোর সাহায্যে সর্বনিম্ন নাম্বার গঠন করলে তা হয় 267 এবং সর্বোচ্চ নাম্বার গঠন করলে তা হয় 762। এখন আমরা যদি 762 থেকে 267 বিয়োগ করি (সর্বোচ্চ নাম্বার থেকে সর্বনিম্ন নাম্বার বিয়োগ) তাহলে তা হবে 495 আবার, আমরা যদি এখন 495 এর ডিজিট গুলোর সাহায্যে সর্বোচ্চ এবং সর্বনিম্ন নাম্বার নির্ণয় করে, সর্বোচ্চ নাম্বার থেকে সর্বনিম্ন নাম্বার বিয়োগ করি তাহলেও আমরা 495 ই পাবো ( $954 - 459 = 495$ )। এই উদাহরণের ক্ষেত্রে (627 এই নাম্বারের) আমরা মাত্র একবার সর্বোচ্চ ও সর্বনিম্ন নাম্বার নির্ণয় করে এবং সর্বোচ্চ থেকে সর্বনিম্ন নাম্বারের বিয়োগফল বের করেই 495 (ম্যাজিক্যাল নাম্বার) পেয়ে গিয়েছি, কিন্তু এমনও নাম্বার আছে যাদের জন্য আমাদের এই একই কাজ কয়েক বার করার পরে 495 উত্তর টি আসবে। একইভাবে 6174 হলো চার ডিজিটের নাম্বারের জন্য একটি ম্যাজিক্যাল নাম্বার। উদাহরণ হিসেবে আমরা যেকোনো চার ডিজিটের একটা নাম্বার কল্লনা করি যেমন 2147 এবং এই নাম্বারের ডিজিট গুলোর সাহায্যে সর্বোচ্চ নাম্বার এবং সর্বনিম্ন নাম্বার নির্ণয় করে, সর্বোচ্চ নাম্বার থেকে সর্বনিম্ন নাম্বার বিয়োগ করলে আমরা বিয়োগফল 6147 ই পাবো ( $7641 - 1467 = 6174$ )।

<b>326 এই নম্বরের জন্য</b>	
<div><div><div>632</div><div>-236</div><div>369</div></div><div>Highest Smallest</div><div><div>936</div><div>-369</div><div>594</div></div><div>Highest Smallest</div><div><div>954</div><div>-459</div><div>495</div></div></div>	<div>Step 1 : এখানে 326 এই নম্বরের ডিজিটগুলোর সর্বোচ্চ ও সর্বনিম্ন নাম্বার নির্ণয় করে, সর্বোচ্চ নাম্বার থেকে সর্বনিম্ন নাম্বার বিয়োগ করা হয়েছে</div> <div>Step 2 : এখানে 369 (Step 1 এ প্রাপ্ত বিয়োগফল) এই নম্বরের ডিজিটগুলোর সর্বোচ্চ ও সর্বনিম্ন নাম্বার নির্ণয় করে আবারো বিয়োগ (সর্বোচ্চ - সর্বনিম্ন) করা হয়েছে</div> <div>Step 3 : এখানে 594 (Step 2 এ প্রাপ্ত বিয়োগফল) ডিজিটগুলো সাজিয়ে পূর্বের মত বিয়োগফল নির্ণয় করা হয়েছে এবং বিয়োগফল 495 (Magical Number)</div>

**Problem :** Write a C / C++ / Java / Python program to count how many steps it will be take to find out the magical number from three digit number that given by the user.

**Problem :** Write a C / C++ / Java / Python program to count how many steps it will be take to find out the magical number from four digit number that given by the user.

## LUCAS NUMBERS

লুকাস নাম্বার হল এমন একটি সংখ্যা সিরিজ, যা 2 এবং 1 দিয়ে শুরু হয়, এবং এর পরবর্তী প্রতিটি সংখ্যাকে পূর্ববর্তী দুটি সংখ্যার যোগফল দ্বারা নির্ধারণ করা হয়। অর্থাৎ, যদি  $L(n)$  লুকাস নাম্বার সিরিজের  $n$ -তম সংখ্যা হয়, তাহলে :  $L(n) = L(n-1) + L(n-2)$  [যেখানে  $n \geq 2$  এবং  $L(0) = 2$  ও  $L(1) = 1$ ] ।

### First 10 Lucas Numbers

$L_1$	$L_2$	$L_3$	$L_4$	$L_5$	$L_6$	$L_7$	$L_8$	$L_9$	$L_{10}$
2	1	3	4	7	11	18	29	47	76

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a lucas number or not, if the number 'N' is given by the user. Print "The number N is a Lucas Number" if the number is a lucas number or print "The number N is Not a Lucas Number" if the number is not a lucas number.

**Problem :** Write a C / C++ / Java / Python program to print all the lucas numbers between 100 and 200.

## MERSENNE NUMBERS

1, 3, 7, 15, 31 ... এই ধারাটি গঠন করা হয়েছে  $2^n - 1$  সূত্রের সাহায্যে, এখানে,  $n$  হল (পদ সংখ্যা) । যেমন এখানে ধারাটির ৫টি পদ দেখানো হয়েছে, আমরা সূত্র ব্যবহার করে ধারাটির ৬ষ্ঠ পদ যদি নির্ণয় করি তবে,  $n = 6$  সুতরাং,  $2^6 - 1 = 64 - 1 = 63$  হবে ধারাটির ৬ষ্ঠ পদ ।

$2^n - 1$	1, 3, 7, 15, 31, 63, 127, 255, 511, 1023, 2047 ...
-----------	--

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a mersenne number or not, if the number 'N' is given by the user. Print "The number N is a Mersenne Number" if the number is a mersenne number or print "The number N is Not a Mersenne Number" if the number is not a mersenne number.

## PELL NUMBERS

0, 1, 2, 5, 12, 29 ... এই সিরিজের প্রত্যেকটি নাম্বারকে বলা হয় Pell Number। ফিবোনাচ্চি বা লুকাস সিরিজের মতই কিন্তু একটু ভিন্ন এই সিরিজের শুরুটা হবে 0 এবং 1 দিয়ে, তারপর পর্যায়ক্রমে নতুন পদ নির্ণয় করা হবে পূর্ববর্তী দুটি পদের উপর কিছু হিসাব-নিকাশ করে। উপরোক্ত সিরিজের 12 এর জন্য যদি আমরা হিসাব করি (যে 12 কিভাবে হলো) তাহলে, 12 এর ঠিক পূর্বের পদের সাথে 2 গুণ করে যা হবে তা, এর ঠিক পূর্বের পদের সাথে যোগ করে দিতে হবে অর্থাৎ 12 এর পূর্বের পদ 5 এবং তার ঠিক পূর্বের পদ 2 সুতরাং,  $5 \times 2 = 10$  এর সাথে 2 যোগ করলে  $10 + 2 = 12$  হয়। একইভাবে, যদি সিরিজের 29 এর জন্য আমরা হিসাব করি (যে 29 কিভাবে হলো) তাহলে, 29 এর ঠিক পূর্বের পদের সাথে 2 গুণ করে যা হবে তা, এর ঠিক পূর্বের পদের সাথে যোগ করে দিতে হবে অর্থাৎ 29 এর পূর্বের পদ 12 এবং তার পূর্বের পদ 5 সুতরাং,  $12 \times 2 = 24$  এর সাথে 5 যোগ করলে  $24 + 5 = 29$  হয়।

### First 10 Pell Numbers

$P_1$	$P_2$	$P_3$	$P_4$	$P_5$	$P_6$	$P_7$	$P_8$	$P_9$	$P_{10}$
0	1	2	5	12	29	70	169	408	985

ধারাটির ৭ম ও ৯ম পদের মান  
নির্ণয়ের বিশ্লেষণ দেখানো হলো

0	1	2	5	12	29	70	169	408
---	---	---	---	----	----	----	-----	-----

$$(169 \times 2) + 70$$



$$(29 \times 2) + 12$$



**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a pell number or not, if the number 'N' is given by the user. Print "The number N is a Pell Number" if the number is a pell number or print "The number N is Not a Pell Number" if the number is not a pell number.

## KEITH NUMBERS

যেকোনো একটি নাম্বার, ধরি (তিন ডিজিটের একটি নাম্বার) 742, এখন এই নাম্বারের ডিজিটগুলো থেকে যদি একটি এমন নাম্বার সিরিজ বানানো হয় যে, সিরিজের শুরুর নাম্বার হবে তিনটি ডিজিট (যেহেতু 742 এটা তিন ডিজিটের একটা নাম্বার) এবং এই সিরিজের প্রতিটি নতুন পদ হবে পূর্ববর্তী তিনটি পদের যোগফল (এখানে পূর্ববর্তী তিনটি পদ নেয়া হয়েছে কারণ 742 নাম্বারটি তিন ডিজিটের ছিলো) সুতরাং সিরিজটি হবে 7, 4, 2, 13, 19, 34, 66, 119, 219, 404, 742। এখানে, ধারাটি ১১তম পদে এসে 742 নাম্বার গঠিত হয়েছে এবং এই 742 নাম্বার থেকেই আমরা ধারাটি গঠন করেছিলাম, সুতরাং 742 এই নাম্বারটি একটি Keith Number। এখন, অন্য একটা উদাহরণ (দুই ডিজিটের কোনো নাম্বারের) দেখা যাক, ধরি 19 এই নাম্বারটি Keith Number কিনা তা যাচাই করতে হবে (অর্থাৎ এই নাম্বার দিয়ে উপরোক্ত নিয়মে নাম্বার সিরিজ নির্ণয় করলে যদি এই নাম্বারটিই সেই সিরিজে থাকে তাহলে তা Keith Number হবে)। সিরিজের শুরুর নাম্বার হবে দুইটি ডিজিট (যেহেতু 19 এটা দুই ডিজিটের একটা নাম্বার) এবং এই সিরিজের প্রতিটি নতুন পদ হবে পূর্ববর্তী দুটি পদের যোগফল (এখানে পূর্ববর্তী দুটি পদ নেয়া হয়েছে কারণ 19 নাম্বারটি দুই ডিজিটের ছিলো) সুতরাং সিরিজটি হবে 1, 9, 10, 19। এখানে, ধারাটি ৪র্থ পদে এসে 19 নাম্বার গঠিত হয়েছে এবং এই 19 নাম্বার থেকেই আমরা ধারাটি গঠন করেছিলাম, সুতরাং 19 এই নাম্বারটিও একটি Keith Number।

### Keith Numbers

$$47 = 4, 7, 11, 18, 29, 47$$

$$61 = 6, 1, 7, 8, 15, 23, 38, 61$$

$$197 = 1, 9, 7, 17, 33, 57, 107, 197$$

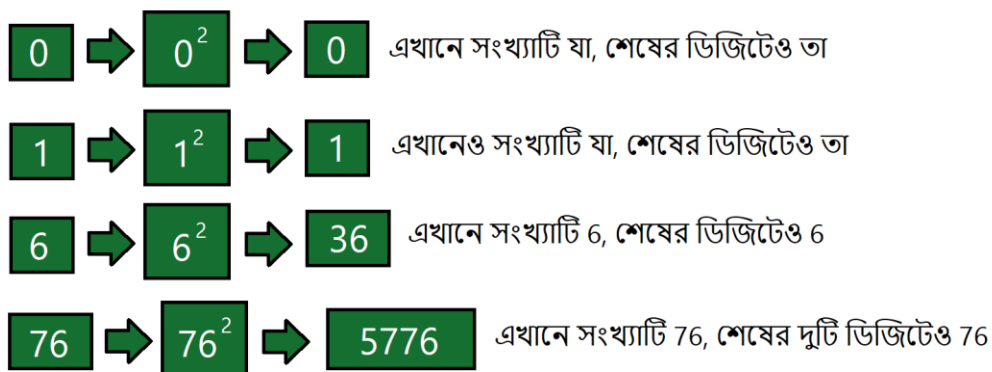
$$1537 = 1, 5, 3, 7, 16, 31, 57, 111, 215, 414, 797, 1537$$

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a keith number or not, if the number 'N' is given by the user. Print "The number N is a Keith Number" if the number is a keith number or print "The number N is Not a Keith Number" if the number is not a keith number.

**Problem :** Write a C / C++ / Java / Python program to print all the Keith numbers between 100 and 1000.

## AUTOMORPIC NUMBERS

কোনো একটি সংখ্যাকে বর্গ করলে যে বর্গসংখ্যা পাওয়া যাবে তার ডিজিটগুলোর মধ্যে যদি শেষের ডিজিটে বা শেষের ডিজিটগুলোতে (অর্থাৎ বর্গসংখ্যার শেষের ডিজিটে বা শেষের ডিজিটগুলোতে) যদি সেই সংখ্যাটি (যে সংখ্যার বর্গ করেছি) থাকে তবে তাকে Automorphic Number অথবা Circular Number বলা হবে। যেমন, ধরি কোনো একটি সংখ্যা 25, এখন, 25 কে বর্গ করলে হয় 625 (এখানে 6, 2, 5 এই ডিজিটগুলোর মধ্যে 6 এর পরে 25 আছে, আর আমরা যে সংখ্যাটি ধরেছিলাম সেটাও 25) সুতরাং 25 একটি Automorphic Number।



**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is an automorphic number or not, if the number 'N' is given by the user. Print "The number N is a Circular Number" if the number is an automorphic number or print "The number N is Not a Circular Number" if the number is not an automorphic number.

**Problem :** Write a C / C++ / Java / Python program to print all the automorphic numbers between 0 and 1000.

## LYCHREL NUMBERS

সাধারণত কোনো সংখ্যাকে ঐ সংখ্যারই বিপরীত বা রিভার্স সংখ্যা দ্বারা যোগ করা হলে প্রাপ্ত যোগফল যদি Palindromic Number না হয় তবে সেই যোগফলকে আবার যোগফলের রিভার্স নাম্বারের সাথে যোগ করলে এবং এই প্রসেস বারবার করতে থাকলে আমরা এক না এক সময় Palindromic Number (প্রাপ্ত যোগফল গুলো থেকে) পেয়ে যাবো। কিন্তু এমন কিছু নাম্বার আছে যাদের যোগফলে আমরা এই Palindromic Number পাবো না, আর সেই সকল নাম্বারগুলোকেই Lychrel Number বলা হয়। যেমন 196, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986, 1495, 1497, 1585, 1587, 1675, 1677, 1765, 1767, 1855, 1857, 1945, 1947, 1997 ইত্যাদি Lychrel Number এর কিছু উদাহরণ।

$\begin{array}{r} 56 \\ +65 \\ \hline 121 \end{array}$ <p><b>Palindromic</b></p>	$\begin{array}{r} 57 \\ +75 \\ \hline 132 \\ +231 \\ \hline 363 \end{array}$ <p><b>Palindromic</b></p>	$\begin{array}{r} 178 \\ +871 \\ \hline 1049 \\ +9401 \\ \hline 10450 \\ +05401 \\ \hline 15851 \end{array}$ <p><b>Palindromic</b></p>	$\begin{array}{r} 493 \\ +394 \\ \hline 887 \\ +788 \\ \hline 1675 \\ +5761 \\ \hline 7436 \\ +6347 \\ \hline 13783 \end{array} \dots$ <p><b>Lychrel Number</b></p> <p><b>No Palindromic</b></p>
--	--	--	--

**মন্তব্য :** Lychrel Numbers সম্পর্কে আরও গবেষণা চলছে, এবং প্রমাণিত হয়নি যে 196 এবং অন্যান্য সংখ্যা সত্যিকার অর্থে Lychrel Number কি না, কারণ বর্তমানে গবেষকরা এখনও সঠিকভাবে নিশ্চিত নয় যে এই সংখ্যা Palindromic Number উৎপন্ন করতে পারবে কি না।



## SOPHIE GERMAIN PRIMES

যদি কোনো মৌলিক সংখ্যাকে 2 দ্বারা গুণ করার পরে 1 যোগ করলে আরও একটি মৌলিক সংখ্যা পাওয়া যায় তবে সেই সংখ্যাটিকে Sophie Germain Primes বলা হবে। যেমন 11 হল Sophie Germain Primes কারণ, 11 কে 2 দ্বারা গুণ করার পর 1 যোগ করলে মানটি হবে  $(11 \times 2) + 1 = 22 + 1 = 23$  যা আরেকটি মৌলিক সংখ্যা। এখন 13 এর জন্য যদি চেক করি যে, 13 Sophie Germain Prime Number কি না তবে,  $(13 \times 2) + 1 = 26 + 1 = 27$  যা মৌলিক সংখ্যা না, সুতরাং 13 Sophie Germain Prime Number নয়।

**For 17 =  $(17 \times 2) + 1 = 34 + 1 = 35$  is Not a Prime Number**

**17 is Not a Sophie Germain Prime**

**For 23 =  $(23 \times 2) + 1 = 46 + 1 = 47$  is a Prime Number**

**23 is a Sophie Germain Prime**

**For 47 =  $(47 \times 2) + 1 = 94 + 1 = 95$  is Not a Prime Number**

**47 is Not a Sophie Germain Prime**

**For 53 =  $(53 \times 2) + 1 = 106 + 1 = 107$  is a Prime Number**

**53 is a Sophie Germain Prime**

**Problem :** Write a C / C++ / Java / Python program to check the number 'N' is a sophie germain prime number or not, if the number 'N' is given by the user. Print "The number N is a Sophie Germain Prime number" if the number is a sophie germain prime number or print " The number N is Not a Sophie Germain Prime number" if the number is not a sophie germain prime number.

**Problem :** Write a C / C++ / Java / Python program to print all the sophie germain prime numbers between 1 and 100.

## MAGIC SQUARE

$n \times n$  আকারের কোনো ম্যাট্রিক্স / স্কয়ারের ভেতরে থাকা উপাদান গুলোর মধ্যে যদি এমন কিছু বৈশিষ্ট্য থাকে যে,

- ❖ স্কয়ারের প্রত্যেক সারি বরাবর, সারিতে থাকা উপাদান গুলোর যোগফল
- ❖ প্রত্যেক কলাম বরাবর, কলামে থাকা উপাদান গুলোর যোগফল
- ❖ মূখ্য কর্ণ বরাবর উপাদান গুলোর যোগফল
- ❖ গৌণ কর্ণ বরাবর উপাদান গুলোর যোগফল

এই সকল যোগফল যদি সমান হয়, তবেই একটি স্কয়ার Magic Square হবে। নিম্নে চিত্রে  $3 \times 3$  আকারের একটা ম্যাট্রিক্স / স্কয়ারের সাহায্যে ম্যাজিক স্কয়ার দেখানো হল।

4	9	2	➡	$4+9+2 = 15$
3	5	7	➡	$3+5+7 = 15$
8	1	6	➡	$8+1+6 = 15$
↓	↓	↓		
$4+3+8$	$9+5+1$	$2+7+6$		
15	15	15		
				<b>Diagonal 1</b> = $4+5+6 = 15$
				<b>Diagonal 2</b> = $8+5+2 = 15$

**Problem :** Write a C / C++ / Java / Python program to check the matrix is a magic square or not. Elements of the matrix should be given by the user [Hint : use loop and two dimensional array].

## BELL NUMBERS TRIANGLE

N = 7	0	1	2	3	4	5	6
A	1						
B	1	2					
C	2	3	5				
D	5	7	10	15			
E	15	20	27	37	52		
F	52	67	87	114	151	203	
G	203	255	322	409	523	674	877

এখানে  $N=7$  অর্থাৎ 7 টা Row এর জন্য Bell Numbers Triangle গঠন করা হল। এখন যদি বিশ্লেষণ করি যে কিভাবে এই সংখ্যা গুলোর মাধ্যমে ত্রিভুজটি গঠন করা হল, তবে, প্রথম Row (Row A) এর **A0** তে মান রয়েছে 1, দ্বিতীয় Row (Row B) এর **B0** এবং **B1** এ যথাক্রমে মান রয়েছে 1 এবং 2, এখানে উল্লেখ্য প্রথম Row এর শুরু এবং শেষ সংখ্যা ছিলো 1। যেকোনো Row এর শেষ সংখ্যাটি পরবর্তী Row এর শুরুর সংখ্যা হবে, যেমনটা প্রথম Row এর শেষে, দ্বিতীয় Row এর শুরুও হয়েছে 1 দিয়ে। আবার দ্বিতীয় Row এর শেষ সংখ্যাটিই হল তৃতীয় Row এর শুরুর সংখ্যা। একইভাবে তৃতীয় Row এর শেষের সংখ্যাটি চতুর্থ Row এর শুরুর সংখ্যা। এখন বাকি সংখ্যাগুলো তাহলে কোথা থেকে এলো? আমরা যদি ২য় Row এর ২য় উপাদানটি খেয়াল করি তবে, এখানে **B0** এবং **A0** এর মান যোগ হয়ে **B1** এ হয়েছে ( $1+1=2$ )। আবার **C0** এবং **B0** এর মান যোগ হয়ে **C1** এ হয়েছে ( $2+1=3$ )। তারপর **C1** এবং **B1** এর মান যোগ হয়ে **C2** তে হয়েছে একইভাবে এখানে কোনো একটি নির্দিষ্ট পজিশনের মান (কোনো Row এর শুরুর উপাদান ব্যতিত) যদি আমরা বের করতে চাই, ধরি (নিম্নের চিত্রানুযায়ী) c এর মান বের করতে চাচ্ছি! তবে c এর ঠিক আগের উপাদান a এর সাথে, a এর ঠিক উপরের উপাদান b যোগ করলেই আমরা c কে পাবো ( $a+b=c$ )।

b		3		7		5		15	
a	c	7	10	20	27	10	15	37	52
a + b = c		7 + 3 = 10		20 + 7 = 27		10 + 5 = 15		15 + 37 = 52	

**Problem :** Write a C / C++ / Java / Python program to print N-th line of Bell Numbers Triangle.