[প্যালিনড্রোম (palindrome)](http://alavolacoder.blogspot.com/2013/06/palindrome.html)

শব্দ বা বাক্য নিয়ে যত ধরনের মজার মজার ব্যাপার হতে পারে তার মধ্যে প্যালিনড্রোম([palindrome](http://en.wikipedia.org/wiki/Palindrome" \t "_blank)) অন্যতম। আমরা মোটামুটি সবাই জানি যে, কোনো একটা শব্দ বা বাক্যকে উভয়দিক থেকে পড়লে যদি একই জিনিস পাওয়া যায় তাহলেই সেটা একটা প্যালিনড্রোম, যেমন- 'MADAM' বা উপরের বাক্যটি।

এখন যদি একজন প্রোগ্রামারকে কেউ বলে আপনাকে একটি বাক্য দেয়া হবে, আপনাকে বলতে হবে বাক্যটি প্যালিনড্রোম কিনা? তাহলে প্রোগ্রামার করবে কি বাক্যটার একটা কপি তৈরি করে, সেটাকে উলটে

দুটো বাক্যকে মিলিয়ে দেখবে। যদি দুটো বাক্য মিলে যায় তাহলে সেটা একটা প্যালিনড্রোম। আর একটু 'চাল্লু প্রোগ্রামার' হলে সে এতো কষ্ট করতে যাবে না। সে করবে কি যেই শব্দ বা বাক্য দেয়া আছে সেটার প্রথম অক্ষরের সাথে শেষ অক্ষরকে মিলিয়ে দেখবে, ২য় অক্ষরের সাথে শেষেরটার আগেরটা মিলিয়ে দেখবে, এভাবে শব্দ বা বাক্যটার মাঝখান পর্যন্ত গেলেই বলে দেয়া যাবে যে শব্দ বা বাক্যটা প্যালিনড্রোম কিনা।

শুধু শব্দ বা বাক্যই নয় একটা নাম্বারও প্যালিনড্রোম হতে পারে। যেমন- ১২৩৪৪৩২১, এটা একটা প্যালিনড্রোমিক নাম্বার।

কোনো একটা শব্দ বা বাক্য বা নাম্বার আসলেই প্যালিনড্রোম কিনা সেটা চেক করা আসলে 'চা' তে 'টোস্ট বিস্কুট' চুবিয়ে খাওয়ার মতই সহজ কাজ। কিন্তু যদি বলা হয়, একটা নাম্বার দেয়া হলো তার পরের প্যালিনড্রোমিক নাম্বার কি হবে?? তখন একটু মাথাটা চুলকানোর প্রয়োজন হয় বৈকি। কিন্তু আসলেই কি সেটা খুব কঠিন কাজ?? এই নাম্বারের পরের নাম্বার একটা একটা করে নিয়ে সেটা প্যালিনড্রোম কিনা চেক করে করেই বের করে ফেলা যায়, তাই না??

হ্যাঁ, কাজটা এভাবেও করা যায়, কিন্তু অনেক বড় নাম্বারের জন্য এভাবে করলে অনেকবার লুপ ঘুরবে, মানে অনেক সময় লাগবে।

এই কাজটাই সহজভাবে করা যায়।

মনে করি আমাদেরকে বলা হলো '134567329807541' এই নাম্বারের পরের প্যালিনেড্রোমিক নাম্বারটা বের করতে হবে।

এখন আমরা নাম্বারটিকে 'head', 'mid' এবং 'tail' এ ভাগ করবো।  
  
**head-> 1345673**  
**mid-> 2**  
**tail-> 9807541**  
  
এখন 'head'  কে উলটে 'tail' এর সাথে compare করবো।  
সেক্ষেত্রে ২ টা ব্যাপার হতে পারে-  
  
**if reverse(head) <= tail**  
(যদি প্রদত্ত নাম্বারটাই একটা প্যালিনড্রোম হয়, তখন reverse(head)= tail হবে, তখন পরের প্যালিনড্রোমটা খুঁজতে হবে)

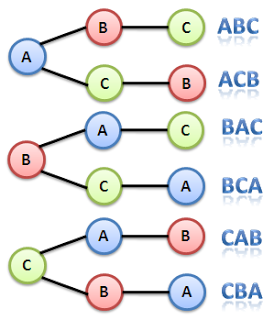
* **if mid<9, increment mid**
* **else increment 'head' part and set mid = 0**

এখন রেজাল্ট পাওয়া যাবে-

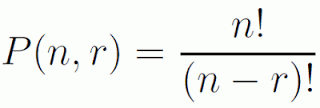
**Result := head mid reverse(head)**

তারমানে আমাদের নাম্বারের জন্য আমরা পাচ্ছি-  
**1345673 3 reverse(1345673) => 134567333765431**  
  
দেখা যাচ্ছে যে, যদি reverse(head) > tail হয়, তখন কোনো কিছু চেঞ্জ না করে শুধু head mid reverse(head) করলেই রেজাল্ট পাওয়া যাবে।

[N-th permutation (N-তম পারমুটেশন)](http://alavolacoder.blogspot.com/2013/06/n-th-permutation-n.html)

[](http://3.bp.blogspot.com/-if4xRuqmj7g/UcxXav8b89I/AAAAAAAAALI/9F6LcjkNGlE/s349/figure13.png)

পারমুটেশন (permutation) বা বিন্যাস প্রোগ্রামারদের খুব পরিচিত এক বিষয়। প্রোগ্রামিংয়ে হাতেখড়ি হবার আগেই আসলে আমাদের পারমুটেশনের সাথে পরিচয় হয়ে যায়। এজন্য পারমুটেশন কি বা পারমুটেশনের সূচনার কথা বলার আপাতত প্রয়োজন মনে করছি না। যদি কেউ পারমুটেশন বের করার সেই বিখ্যাত সূত্র ভুলে গিয়ে থাকো শুধু তাদের মনে করিয়ে দিচ্ছি।

[](http://1.bp.blogspot.com/-5sAlJvKeJAE/UcxXaSfaYkI/AAAAAAAAALM/9229q9EznDY/s431/perm.GIF)

"১-৯ পর্যন্ত অংকগুলি(digit) ব্যবহার করে ৩ অংকের কতগুলি সংখ্যা বানানো যায়?"- এরকম কাগুজে সমস্যার সমাধান পরীক্ষার খাতায় 'ধুম-ধাম' লিখে দিয়ে আসার পর যখন একজন প্রোগ্রামার কীবোর্ড টিপে পারমুটেশন সমস্যার সমাধান করতে বসেন তখন তার সামনে প্রথম চ্যালেঞ্জই হয় 'পারমুটেশন জেনারেট(permutation generate)' করা। যেমন- উপরের ছবির মত করে যদি বলা হয় 'ABC' দিয়ে যতগুলো বিন্যাস তৈরি করা সম্ভব সবগুলি বিন্যাস তৈরি করার জন্য একটা কোড কর। তখন তো প্রোগ্রামার বগল বাজাতে বাজাতে ৩ টা লুপ মেরে দিয়ে সমস্যাটার সমাধান করে দিলেন। কিন্তু যখন ৫টা কিংবা ১০টা অংকের জন্য একই কাজ করতে বলা হবে তখন নিশ্চয়ই একইভাবে ৫টা-১০টা লুপ মেরে দিয়ে এই সমস্যার সমাধান করা যাবে না, তাই না?

কিভাবে তাহলে পারমুটেশন বের করা যায়?

পারমুটেশন বের করার সহজ ২টা বুদ্ধির কথা বলবো।

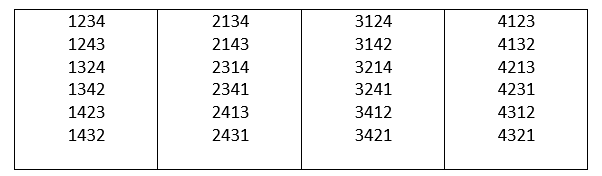
প্রথম বুদ্ধি হলো, [STL](http://en.wikipedia.org/wiki/Standard_Template_Library) এর [next\_permutation()](http://www.cplusplus.com/reference/algorithm/next_permutation/) এবং [prev\_permutation()](http://www.cplusplus.com/reference/algorithm/prev_permutation/) ব্যবহার করে খুব সহজেই পারমুটেশন বের করা যায়।

আরেকটা বুদ্ধি হলো, ব্যাকট্র্যাকিং করে পারমুটেশন জেনারেট করা। এটা ভালোভাবে বুঝতে হলে [এখানে](http://www.shafaetsplanet.com/planetcoding/?p=1266) (ব্যাকট্র্যাকিং : পারমুটেশন জেনারেটর) দেখতে পারো।

যাই হোক, উপরের ২টা বুদ্ধিই হলো All permutation generate এর বুদ্ধি। আমি এই পোষ্টে আরেকটা বুদ্ধি নিয়ে বলবো যেটা হলো, **"আগের পারমুটেশনগুলো জেনারেট না করেই N-তম পারমুটেশন কিভাবে বের করা যায়"**।

সবচাইতে ভালো বুদ্ধি হলো, এই পর্যন্ত পড়ার পর নীচের টুকু পড়ার আগে নিজে নিজে একটু চিন্তা করা, কাজটা আসলেই কিভাবে করা যায়? যেমন, 'abc' দিয়ে বলা হলো এটার 5-তম বিন্যাসটা কি হবে? আগের গুলো বের না করেই কি বলে দেয়া যায় যে 5-তম বিন্যাস হবে 'cab'?

মনে করি, উদাহরণ হিসেবে আমাদের হাতে আছে, '1234'. এখন বোঝার সুবিধার জন্য আমরা দেখি যে, '1234' এর সবগুলি পারমুটেশন কি কি?

[](http://3.bp.blogspot.com/-GNVBkMOYhY8/UcxkIIBaOtI/AAAAAAAAALs/qYTSZ5mO9qo/s602/2013-06-27_220756.png)

লক্ষ্যণীয় যে, আমাদের উদাহরণে **4** টা ডিজিট- **'1', '2', '3', '4'**. আর  আমরা আমাদের মোট **4! = 24** টা বিন্যাসকে **4** ভাগে ভাগ করেছি যেখানে প্রতিভাগে **(4-1)! = 3! = 6** টা করে বিন্যাস আছে।

তারমানে আমরা সাধারণভাবে বলতে পারি,

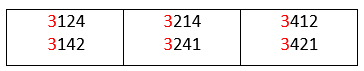
***যদি N টা ডিজিট দেয়া থাকে তাহলে তার N! বিন্যাসকে আমরা N ভাগে ভাগ করতে পারবো, যেখানে প্রতি ভাগে (N-1)! টা বিন্যাস থাকবে।***

 এখন এই **(N-1)! বিন্যাসের N টা ভাগ** কি আমাদের কোনো কাজে লাগবে?

উপরের টেবিলটা দেখলেই উত্তরটা পাওয়া যাবে। দেখা যাচ্ছে, প্রথম ভাগে সবগুলো বিন্যাসের প্রথম অক্ষরটা '1' তারমানে আমাদের প্রথম ডিজিট, তারপরের ভাগের প্রথম অক্ষর '2' মানে আমাদের দ্বিতীয় ডিজিট, এভাবে বাকিগুলো। তারমানে হলো, মনে করি আমাদের '1234' এর 17-তম বিন্যাস বের করতে বলা হয়েছে। এখন আমরা দেখবো ১৭ সংখ্যার বিন্যাসটা কত-তম ঘরে পড়বে। তাহলেই আমরা আমাদের কাঙ্খিত বিন্যাসের প্রথম অক্ষর কি হবে বলে দিতে পারবো।

তাহলে, ১৭-তম বিন্যাস কোনঘরে পড়বে আমরা বের করতে পারি, **(17 / 3!) = 3 (after ceiling)** . তারমানে ৩য় ঘরে। সুতরাং আমাদের উত্তরের প্রথম সংখ্যাটা হবে '3'.

এখন আমরা জানি যে আমাদের কাঙ্খিত বিন্যাসটি ৩য় ঘরে আছে। এবার আমরা শুধু ৩য় ঘরের দিকেই নজর দেবো আর এটাকেও আগের মত ভাগ করবো। যেহেতু আমরা প্রথম সংখ্যাটা কি হবে জেনে গেছি, তারমানে আমাদের কাজ হবে শেষের ৩ সংখ্যা ( '1', '2', '4') নিয়ে। তাহলে আমাদের টেবিলটি হবে এরকম-

[](http://1.bp.blogspot.com/-QbSYncS8OFE/UcxppYkRoLI/AAAAAAAAAL8/aOZPHsSbdu8/s364/2013-06-27_223413.png)

এখানে, মোট (**N-1)! = 3! = 6** টা বিন্যাস আছে **(6/(N-2)!) = 3** ভাগে, এবং প্রতিভাগে আছে **(N-2)! = 2** টি বিন্যাস। আমরা যেহেতু আগের ২ ভাগে মোট **2\* 3! = 12** টি বিন্যাস বাদ দিয়েছে, তারমানে এখন আমাদের **17-12 = 5** তম বিন্যাসটি খুঁজতে হবে। (এটাকে **17 mod 6 = 5** করেও পাওয়া যায়)

এখানেও আমরা দেখেবো যে, প্রতিটি ভাগে '২য়' সংখ্যাটি একই। তারমানে আমরা আগের পদ্ধতিতে, **5 / 2! = 3 (after ceiling)** , মানে ৩য় ঘরে আমাদের উত্তরটি পাবো।

তারমানে আমাদের ২য় সংখ্যাটি হবে '1', '2', '4' এর মধ্যে ৩য়টি অর্থাৎ '4'। (যেহেতু, '3' ইতোমধ্যে নেয়া হয়ে গেছে।

এখন বাকি ২টি সংখ্যার জন্য আমরা পাবো-

[http://1.bp.blogspot.com/-ec-9HIFAljg/UcxstGRZXWI/AAAAAAAAAMM/a3P1kp9DH7U/s252/2013-06-27_224732.png](http://1.bp.blogspot.com/-ec-9HIFAljg/UcxstGRZXWI/AAAAAAAAAMM/a3P1kp9DH7U/s252/2013-06-27_224732.png)

একইভাবে, হিসেব করতে থাকলে আমরা দেখবো আমাদের ৩য় সংখ্যাটি '1' এবং ৪র্থ সংখ্যাটি '2'।

তারমানে আমাদের উত্তরটি দাঁড়ায় '3412' , যেটি 17-তম বিন্যাস।

ব্যস হয়ে গেলো আমাদের N-তম বিন্যাস।

এখানে সবগুলি বিন্যাস দেখানোর কারণ হচ্ছে, সাধারণ পদ্ধতিটা বোঝানো। এখন খুব সহজেই N-তম পারমুটেশন বের করার কোড করে ফেলা যায়, উপরের বুদ্ধিতে, যেখানে বাকি বিন্যাসগুলো বের করার দরকার পড়ে না।

নাহ, আমি কোড করে দিচ্ছি না। ওটা তোমার কাজ। :)

আচ্ছা যদি এরকম দেয়া থাকে - **একটা পারমুটেশন 'cba' দেয়া আছে, এটা কততম পারমুটেশন?**

তাহলে কি করবো??? একটু মাথা খাটালেই **কইয়ের তেল দিয়ে কইটা ভাজা যাবে**। :)

মানে হলো, আমরা একটু আগে যে বুদ্ধি দিয়ে N-তম পারমুটেশন বের করেছি, এখনও সেই বুদ্ধিটাকে উলটে দিয়ে আমরা একাজটা করে ফেলতে পারি।

কিভাবে???

এখানে দেখা যাচ্ছে, ১ম অক্ষর 'c'. তার মানে আমরা বুঝতে পারছি, 3 টি সংখ্যার জন্য 3 টি ভাগে আমরা যে টেবিলটি বানাতাম (আগের বুদ্ধির প্রথম ধাপ) তার ৩য় ঘরে আছে আমাদের পারমুটেশনটি। তারমানে আগের ২ ঘরে আমাদের বিন্যাস আছে**2\*(3-1)! = 4** টি।

এখন, পরের অক্ষর 'b' এর জন্যও আমরা বলতে পারি,  বাকি ২ অক্ষরের জন্য যে ২ ভাগে একটি টেবিল বানানো হবে তার ২য় ঘরে থাকবে 'b'. তারমানে এর আগে একটি ঘরে **1\*(3-2)! = 1** টি বিন্যাস আছে। তারমানে এখন পর্যন্ত আমরা বিন্যাস অতিক্রম করেছি (মানে বর্তমান বিন্যাসের আগে আছে)  ৪+১=৫ টি।

অবশিষ্ট অক্ষর 'a'। এবং এটির পূর্বে আর কোন বিন্যাস নেই।

সুতরাং, আমাদের প্রদত্ত বিন্যাস 'cba' এর আগে বিন্যাস আছে ৪+১=৫ টি। তারমানে এটি ৬ষ্ঠ বিন্যাস।

তারমানে আমাদেরকে পারমুটেশন দেয়া হলেও আমরা খুব সহজে বলে দিতে পারবো বিন্যাসটি কততম।

পারমুটেশন নিয়ে আজকে এ পর্যন্তই।

বি:দ্র: ধরেই নিচ্ছি এ পোষ্টটির কিছুই তোমার মাথায় ঢুকেনি। সুতরাং, এখন একটা কলম নাও, আর একটা কাগজ নাও। তারপর সেখানো নিজের মত করে হিসেব করে বোঝার চেষ্টা করো, আসলেই ব্যাপারটা কি হয়? আশা করি তখন আরো সহজে বুঝতে পারবে। :)

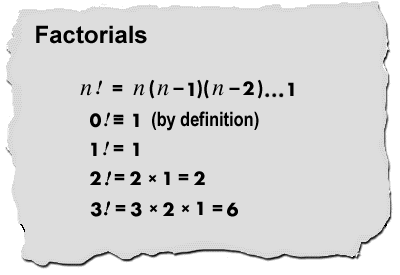
[ফ্যাক্টোরিয়াল ফ্যাক্টস (Factorial Facts)-১](http://alavolacoder.blogspot.com/2013/04/factorial-facts.html)

Factorial সম্পর্কে আমরা মোটামুটি সবাই বেশ ভালোই জানি। যারা জানি না বা ভুলে গেছি তাদের জন্য একটু অ, আ , ক, খ বলে নেয়া ভালো।

কোনো একটা নাম্বারের Factorial হলো, ঐ নাম্বার থেকে ছোট বা সমান যতগুলো পজিটিভ নাম্বার আছে তাদের সবগুলোর গূণফল। যেটাকে প্রকাশ করা হয় এভাবে N!

আবার, কোনো একটা নাম্বারের Factorial বের করার জন্য নাম্বারটিকে তার আগের নাম্বারের Factorial এর সাথে গুণ করেও পারা যায়। অর্থাৎ-

N! = N\* (N-1)!

[](http://4.bp.blogspot.com/-xqX2a6hXzTo/UWBFTGTUHiI/AAAAAAAAAJE/qhhbJNw9cVo/s1600/Factorial+(1).gif)

Factorial এর (!) এই চিহ্ন প্রথম ব্যবহার করেন ফরাসী গণিতবিদ [Christian Kramp](http://en.wikipedia.org/wiki/Christian_Kramp).

এখন কথা হলো Factorial কোথায় কোথায় প্রয়োজন পড়ে তা কি আমরা জানি?  
Factorial এর সবচাইতে বেশি ব্যবহার Combinatorics এ। যেমন, আমরা যদি বলি N টি বিভিন্ন বস্তুকে আমরা কতভাবে বিন্যস্ত (permutation) করতে পারি, সেক্ষেত্রে উত্তরটি হবে N!.

যাই হোক, ফ্যাক্টরিয়াল কি তাতো খুব সহজে বোঝা গেল, এখন যে কাউকে ধরে বসিয়ে দিলেই সে একটা একটা করে গুণ করে যে কোনো নাম্বারের Factorial খুব সহজেই বের করতে পারবে। তাই না?

আচ্ছা, আমরা একটু দেখি তো 20! কত?

20!= 2432902008176640000

ঘটনা কি? এত বড় কেন?

আচ্ছা, 50! কত?

50!=30414093201713378043612608166064768844377641568960512000000000000

তাহলে ১০০ এর Factorial বের করতে হলে অবস্থা তো পুরাই শ্যাষ!!!!!!!!

Factorial হচ্ছে সেই নোটেশন যার মান খুব দ্রুত বাড়তে থাকে। সুতরাং, ইচ্ছে করলেই আমরা আমাদের বাসার চাল ডালের হিসেব করা ক্যালকুলেটরে ১০০ এর factorial বের করতে পারবো না, কারণ সেটা খুব বড় একটা সংখ্যা হয়ে যাবে।  
  
 আমরা যদি কোনো একটা নাম্বারের Factorial বের করতে চাই, কোড করে সেটা খুব দ্রুতই বের করে ফেলা যায়।

এই পদ্ধতিতে খুব বড় নাম্বারের Factorial বের করা যায় না। সেক্ষেত্রে আমরা অন্য কোনো পদ্ধতি ব্যবহার করবো। সেক্ষেত্রে আমরা, JAVA এর Big Integer ব্যবহার করতে পারি, যেখানে বড় নাম্বার নিয়ে খুব সহেজই কাজ করা যায়, আর যদি C/C++ দিয়েই করতে চাই সেক্ষেত্রে নাম্বারগুলোকে Character String হিসেবে বিবেচনা করে স্ট্রিং এর Addition,Multiplication করে করা যায়, যদিও সেটা বেশ বড় প্রসেস হয়ে যায়। তাই, সেটা নিয়ে এখন আলোচনা করতে চাচ্ছি না। নিজেরাই চেষ্টা করে দেখতে পারো। :)

একটা নাম্বারের Factorial যেহেতু খুব বড় হয়ে যায় তাই Factorial  বের না করেও আমরা ঐ বড় নাম্বারের আরো কিছু বৈশিষ্ট্য নিয়ে কাজ করতে পারি। সেরকমই কয়েকটি বিষয় নীচে বলার চেষ্টা করলাম-

**Number of Digits in N! :**

আমরা জানি কোনো একটা নাম্বার N এ কতগুলো ডিজিট আছে সেটা বের করার পদ্ধতি হলো-

num\_of\_digit= ceiling(log(N))

কিন্তু নাম্বারটা যদি ১০ এর perfect power হয় সেক্ষেত্রে আমরা এক(১) কম পাবো। এজন্য ভালো বুদ্ধি হলো-

num\_of\_digit= floor(log(N))+1

নাম্বারটা যদি 10-base এর হয়, তাহলে এ পদ্ধতি ঠিক আছে। কিন্তু নাম্বারের base যদি অন্য কিছু হয়, যেমন আমরা যদি জানতে চাই বাইনারী (base-2) তে এই নাম্বারের কতগুলো ডিজিট থাকবে, তাহলে আমরা এভাবে করতে পারি-

num\_of\_digit= floor(log(N)/log(b))+1

এখানে b=base, বাইনারীর জন্য, b=2 , ডেসিমাল এর জন্য b=10 হবে, ইত্যাদি ইত্যাদি।

এখন এই একই বুদ্ধিতে আমরা N! এ কতগুলো ডিজিট আছে সেটা বের করতে পারি।  
যদি N! অনেক বড় নাম্বারও হয়, সেক্ষেত্রে আমাদের দুশ্চিন্তার কিছু নেই। কারণ আমরা N! = ? সেটা বের না করেই বলে দিতে পারবো যে, N! এ কতগুলো ডিজিট আছে।  
এখন আমরা যদি উপরের নিয়মে N! এর জন্য করতে চাই তাহলে প্রক্রিয়াটা কি হবে বোঝা যাচ্ছে নিশ্চয়ই?

digit\_of\_fact= floor(log(N!))+1

                  = floor(log(N\*(N-1)\*(N-2)\*........\*2\*1))+1

এখন, যদি আমরা N! বের করে তারপর তার log  করতে চাই তাহলে তো সমস্যা। এজন্য আমরা ছোটোবেলায় শেখা log এর ফর্মুলা ব্যবহার করবো। আমরা জানি,

log(a\*b\*c) = loga+logb+logc

তারমানে আমাদের কাজ সহজ হয়ে গেলো-

digit\_of\_fact = floor(logN+log(N-1)+log(N-2)+..........+log1)+1

এখন আর কোনো সমস্যা নেই, কারণ আমরা যখন নাম্বারগুলোর log ব্যবহার করবো তখন নাম্বারগুলো অনেক ছোটো হয়ে যাচ্ছে আর মোটের ওপর log গুলোর যোগফল আমাদের int এর লিমিটের মধ্যেই থাকবে।

উপরের ফর্মুলা তো base-10 এর জন্য, কিন্তু আমরা যদি অন্য base এর জন্য করতে চাই তাহলে কি হবে? হ্যাঁ, আগের মতই, অর্থাৎ-

digit\_of\_fact = floor((logN+log(N-1)+log(N-2)+..........+log1) / logB)+1

তো, কোড করা তো এখন খুবই সহজ হয়ে গেলো, করে ফেলো ধুপধাপ। :)

**An Interesting Fact:**

আমরা তো দেখলাম যে, 1 থেকে N পর্যন্ত নাম্বারগুলোকে গুণ করলে আমরা Factorial of N পাই।

Factorial(N)= 1\*2\*3\*.................\*(N-1)\*N

কিন্তু আমরা যদি নাম্বারগুলোকে গূণ না করে যোগ করি সেক্ষেত্রে আমরা একটা মজার জিনিস দেখবো।

1=1

1+2=3

1+2+3=6

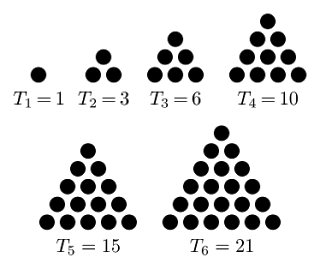
1+2+3+4=10

1+2+3+4+5=15

1+2+3+4+5+6=21

...............................

এই যে নাম্বারগুলো পাওয়া গেলো, 1,3,6,10,15,21................এই নাম্বারগুলোকে বলা হয় Triangular Number.  
Triangular Number বলা হয় সেই সব নাম্বারকে যাদেরকে দিয়ে Triangle বানানো যায়, অর্থাৎ ৩ টি ডট(.) কে বসিয়ে একটা ত্রিভুজ বানানো যায়, ৬টি কে বসিয়ে বানানো যায়, ১০ টি কে বসিয়ে বানানো যায়, কিন্তু কখনোই ৪,৫,৭,৮...... এগুলোকে দিয়ে বানানো যাবে না। এরকম-

[](http://1.bp.blogspot.com/-j3f0GDkOlOs/UWBlEYeEHHI/AAAAAAAAAJo/NhLOlzcTez4/s1600/374px-First_six_triangular_numbers.svg.png)

তারমানে, আমরা 1 থেকে N পর্যন্ত যোগ করলে N-তম Triangular Number পাবো। Triangular number সম্পর্কে আরো জানতে [এখানে](http://www.mathsisfun.com/algebra/triangular-numbers.html) দেখতে পারো।

http://www.mathsisfun.com/algebra/triangular-numbers.html

তো Factorial নিয়ে আজকে এ পর্যন্তই, আরো কিছু জিনিস বলার আছে। আগামী পোষ্টে বলার ইচ্ছে আছে।

[ফ্যাক্টোরিয়াল ফ্যাক্টস(Factorial Facts)-২](http://alavolacoder.blogspot.com/2013/04/factorial-facts_9.html)

**ফ্যাক্টরিয়াল ফ্যাক্টরস(Factorial Factors):**

কোনো একটা নাম্বার এর প্রাইম ফ্যাক্টর গুলো যদি বের করতে হয়, তাহলে আমরা সেই নাম্বার থেকে ছোট যে প্রাইম নাম্বারগুলো([prime number](http://en.wikipedia.org/wiki/Prime_number" \t "_blank)) আছে সেগুলো দিয়ে দিয়ে ভাগ করে করে ঐ নাম্বারটাকে ফ্যাক্টরাইজ করতে পারি। আমরা কিন্তু ছোটোবেলায় একাজটা করেছি, হয়তো অনেকের মনে নেই। যেমন, আমরা ছোটোবেলায় অংক করার সময় যখন গ.সা.গু আর ল.সা.গু বের করতাম, তখনও কিন্তু আমরা প্রথমে ২ টা নাম্বারের প্রাইম ফ্যাক্টরগুলো বের করে নিয়ে তারপর গ.সা.গু. বা ল.সা.গু. বের করতাম। তারমানে, আমরা যদি ৬০ এর প্রাইম ফ্যাক্টরগুলো বের করতে চাই তাহলে পাবো-

৬০= ২\*২\*৩\*৫

এভাবে, প্রতিটা নাম্বারের ফ্যাক্টরগুলো বের করে তারপর আমরা সেখান থেকে কমনগুলো নিয়ে গ.সা.গু বের করতাম। যাই হোক, এখানে কিভাবে একটা নাম্বারকে আমরা ফ্যাক্টরাইজ করতে পারি সেটা নিয়ে বলবো না। কাজটা খুবই সহজ, যারা পারো না কিভাবে করতে হয় তারা আগে অবশ্যই [জানে আলম জান](https://plus.google.com/110406683621006987856) ভাইয়ের একটা সুন্দর টিউটরিয়াল আছে [prime factorization](http://www.lightoj.com/article_show.php?article=1002) নিয়ে। আগে এটা শিখে নাও।  
উপরের লিংক কাজ না করলে [এখানে](https://www.dropbox.com/s/ndai0fquchmazu7/factorization.pdf) দেখতে পারো।

তবে অবশ্যই আগে একটা সাধারণ নাম্বারের প্রাইম ফ্যাক্টর কিভাবে বের করে এটা জানতে হবে। তা নাহলে আমি এখন যে ফ্যাক্টরিয়ালের প্রাইম ফ্যাক্টর বের করার কথা বলবো তার সবটাই নেটওয়ার্কের উপর দিয়ে যাবে।

তাহলে এবার কিভাবে ফ্যাক্টরিয়ালের প্রাইম ফ্যাক্টরস বের করা যায় সেটা নিয়ে বকর বকর শুরু করি।

আমরা জানি যে, খুব ছোট নাম্বারের ফ্যাক্টরিয়ালও অনেক বড় সংখ্যা হয়ে যেতে পারে। সুতরাং, আগে ফ্যাক্টরিয়াল বের করে তারপর তাকে প্রাইম দিয়ে ভাগ করে করে আগের বু্দ্ধিতে........................... নাহ, এটা খুবই বাজে একটা বুদ্ধি। কারণ, অতবড় একটা সংখ্যাকে এভাবে ফ্যাক্টরাইজ করতে গেলে আমাদের মোটামুটি খবরই আছে। তাহলে, কি করা যায়????? কোনো বুদ্ধি????  
হুম...... "Always there is a solution". :)  
  
আচ্ছা, আমরা দেখি যে 10! মানে কি?

10!= 1\*2\*3\*4\*5\*6\*7\*8\*9\*10

ঠিক আছে, তাই যদি হয়, তাহলে আমরা একটু বুদ্ধি খাটাই এখানে, এই সংখ্যাগুলোকে আরো একটু ভাঙ্গি, এরকম করে-

10!= 1\* 2 \* 3 \* (2\*2) \* 5 \* (2\*3) \* 7 \* (2\*2\*2) \* (3\*3) \* (2\*5)

বুদ্ধিমান ব্যক্তিবর্গ এতক্ষণে ধরে ফেলার কথা, আমি এখানে কি করেছি। হ্যাঁ, আমি এখানকার নাম্বারগুলোকে ফ্যাক্টর করে তাদের বদলে ফ্যাক্টর গুলো বসিয়েছি। এটার সাথে কিন্ত আগে লেখা (1\*2\*3\*4......) এটার কোনো পার্থক্য নেই। কিন্তু লাভ যেটা হয়েছে সেটা হলো আমি এখন শুধুমাত্র প্রাইম নাম্বারের গুণফল হিসেবে 10! কে দেখাতে পারছি। তারমানে আমরা কিন্তু এরিমধ্যে 10! কে ফ্যাক্টরাইজ করে ফেলেছি।

যদিও 10!= 3628800  বেশ বড় একটা সংখ্যা কিন্তু আমরা খুব সহজেই এটার ফ্যাক্টরগুলো বের করে ফেলেছি। তারমানে এখন 10! এর prime factorization হচ্ছে-

10! = (2\*2\*2\*2\*2\*2\*2\*2) \* (3\*3\*3\*3) \* (5\*5) \* 7

      = 28 \* 34\* 52 \* 7

এখন এখান থেকে আমরা কয়েকটা গুরুত্বপূর্ণ জিনিস নোট করতে পারি-

১. N! বের করার জন্য N এর চেয়ে বড় প্রাইম আমাদের লাগবে না।

২. আমরা যদি শুধু বের করতে পারি যে, ফ্যাক্টরাইজেশনে এই প্রাইমগুলোর ফ্রিকোয়েন্সি (মানে, প্রাইমগুলো কতগুলো করে আছে) কত তাহলেই আমাদের ফ্যাক্টরাইজেশন করা হয়ে যাবে।

তারমানে, আমাদের N পর্যন্ত প্রাইম গুলো লাগবে আর প্রাইমগুলোর পাওয়ার কত হবে সেটা লাগবে।

আমরা যদি আবার একটু ১০ এর ফ্যাক্টরিয়ালের দিকে তাকাই, প্রথমে আছে  28 ।

আচ্ছা, এটা কিভাবে আসলো একটু দেখি।

10!= 1\*2\*3\*4\*5\*6\*7\*8\*9\*10

10!= 1\* 2 \* 3 \* (2\*2) \* 5 \* (2\*3) \* 7 \* (2\*2\*2) \* (3\*3) \* (2\*5)

এখানে আমরা দেখি কোন নাম্বারগুলো 2 দিয়ে বিভাজ্য বা কোথায় কোথায় আমরা 2 পাবো।

২,৪,৬,৮,১০ এই ৫ জায়গায় আমরা অন্তত একবার করে ২ পাচ্ছি। তারমানে আমরা এখন পর্যন্ত ২ পেলাম (১০/২ ) = ৫ বার।

আবার ২\*২=৪ এর কারণে আমরা আরো একবার করে ২ পাচ্ছি- ৪ এবং ৮ এ। তারমানে আরো পেলাম, (১০/৪) = ২ বার। মোট হলো ৫+২=৭।

আবার ২\*২\*২ = ৮ এর জন্য আরো একবার ২ পাচ্ছি শুধুমাত্র ৮ এ। তারমানে আরো পেলাম, (১০/৮)=১ বার। মোট হলো, ৫+২+১=৮।

আর কোনো ২ আমরা পাচ্ছি না, কারণ ২\*২\*২\*২=১৬ আমাদের নাম্বার ১০ এর চেয়ে বড়। তারমানে এরপরে কোনোটার জন্য আমরা কোনো ২ পাবো না। এখন তাহলে আমরা দেখি যে, 10! এর জন্য আমরা কতগুলো ২ পাবো সেটা আমরা পেয়ে গেছি,৫+২+১=৮, তারমানে 28।

একইভাবে আমরা এবার খুব সহজে বলে দিতে পারি আমাদের পরবর্তী প্রাইম ৩ এর পাওয়ার কত হবে, তাই না?

শুধুমাত্র ৩ পাবো আমরা ৩,৬,৯ এ। আর ৩\*৩ এর জন্য আরো একবার ৩ পাবো ৯ এ।

তারমানে ৩ এর পাওয়ার হবে, (১০/৩)+(১০/৯)=৩+১=৪

আমরা উপরে আমাদের ফ্যাক্টরাইজেশনের লাইনে দেখবো যে আসলেই আমাদের 34 আছে। তারমানে বাকি প্রাইমগুলো একইভাবে পাওয়া যাবে।

৫ হবে, (১০/৫)=২

৭ হবে, (১০/৭)=১

তারমানে,

10! = 28 \* 34 \* 52 \* 7

তারমানে মোদ্দাকথা দাঁড়ালো যে, কোনো একটা প্রাইমের পাওয়ার পাওয়া যাবে, তার যেই পাওয়ারগুলো N এর চেয়ে ছোটো, তারা কতবার করে N! এ আছে, যেমন,

(১০/২) = ৫ , ২ আছে ৫ বার

(১০/৪) = ২, ৪ আছে ২ বার

(১০/৮) = ১, ৮ আছে ১ বার

২ এর আর কোনো পাওয়ার নেই ১০ এর মধ্যে তারমানে, 10! এ ২ এর পাওয়ার হবে (৫+২+১)=৮

একইভাবে বাকি প্রাইমগুলো বের করে নিলেই আমরা 10! এর prime factorization পেয়ে যাবো।

মজার কথা হচ্ছে, আমরা কিন্তু 10! এর প্রাইম ফ্যাক্টর বের করার সময় একবারো দেখিনি, 10!=? , অথচ আমরা কিন্তু ঠিকই 10! এর prime factorization করে ফেলেছি। :) কি দরকার অত বড় একটা নাম্বারকে ঘাঁটাঘাঁটি করার? বুদ্ধি থাকলে যে উপায় হয় সেটা তো আমরা দেখলামই।

কতটুকু বুঝতে পেরেছো উপরের হাবিজাবি, সেটা বোঝার জন্য, [UVA 884 Factorial factors](http://uva.onlinejudge.org/external/8/884.html) এই প্রবলেমটা সলভ করার চেষ্টা করে দেখতে পারো। :)

[sample code.](http://alavolacodes.blogspot.com/2013/07/factorial-factorization.html)

Factorial factorization

This function calculates the factorization of the given number's factorial.

int factors(int num)

{

int i,tmp,cnt;

for(i=0;i<max\_prime && prime[i]<=num;i++)

{

cnt=0;

tmp=num;

cout<< prime[i] << " ";

while(tmp>0)

{

cnt+=floor(tmp/prime[i]);

tmp/=prime[i];

}

cout<< cnt << endl;

}

}