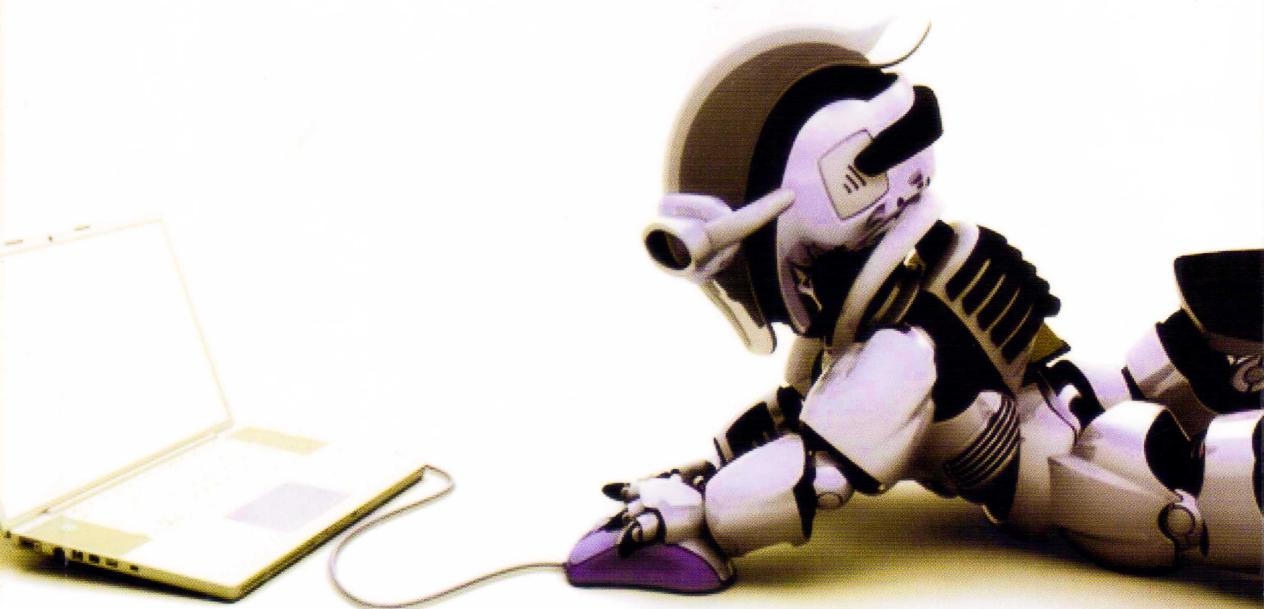


প্রোগ্রামিং কনটেক্ট সিরিজ

৫২টি প্রোগ্রামিং সমস্যা ও

সমাধান

তামিম শাহরিয়ার সুবিন
তাহমিদ রাফি
তামাঙ্গা নিশাত রিনি



দিশা প্রকাশনী

৫২টি

প্রোগ্রামিং সমস্যা ও সমাধান

তামিম শাহরিয়ার সুবিন

তাহমিদ রাফি

তামাঙ্গা নিশাত রিনি



দিবিক প্রকাশনী

৫২টি প্রোগ্রামিং সমস্যা ও সমাধান

তামিম শাহরিয়ার সুবিন, তাহমিদ রাফি ও তামান্না নিশাত রিনি

প্রথমস্থান: তামিম শাহরিয়ার সুবিন

প্রথম প্রকাশ: ফেব্রুয়ারি, ২০১৬

দ্বিতীয় সংস্করণ: মার্চ, ২০১৬

প্রকাশক,
দ্বিমিক প্রকাশনী

৭৪১/এ, বায়তুল আমান হাউজিং সোসাইটি,
আদাবর, ঢাকা-১২০৭

প্রচ্ছদ: নিয়ামুল কবির

মুদ্রণ
আলফা প্রিন্টার্স
১৬৭, গাউসুল আজম মার্কেট, নীলক্ষেত, ঢাকা

প্রাপ্তিষ্ঠান:

- | | |
|----------------------------------|----------------------------------|
| • মানিক হক লাইব্রেরী | হক লাইব্রেরী |
| বাবুপুরা মার্কেট, নীলক্ষেত, ঢাকা | বাবুপুরা মার্কেট, নীলক্ষেত, ঢাকা |
| ০১৭৩৫-৭৪২৯০৮ | ০১৮২০-১৫৭১৮১ |

মূল্য: ২৪০ টাকা

52 Programming Problems and Solutions

By Tamim Shahriar Subeen, Tahmid Rafi & Tamanna Nishat Rini

First Published: February 2016,

Second Edition Published: March 2016,

Price: 240 Taka

ISBN: 978-984-92164-0-7

উৎসর্গ

৫২-এর ভাষা সৈনিকদের, যাদের ত্যাগের কারণে বেঁচে আছে বাংলা ভাষা।

-

ভূমিকা

প্রোগ্রামিংয়ে দক্ষতা অর্জন করতে হলে প্রোগ্রামিং সমস্যা সমাধান করা বা যাকে আমরা প্রবলেম সলভিং বলি, সেই জিনিসটি করার কোনো বিকল্প নেই। কেউ যখন নতুন প্রোগ্রামিং শিখবে, তখন প্রোগ্রামিংয়ের মৌলিক ধারণাগুলো আয়ত্তে আনার পরপর কয়েকশ ঘণ্টা প্রোগ্রামিং চর্চা করা আবশ্যিক। ইন্টারনেটে হাজার হাজার প্রোগ্রামিং সমস্যা রয়েছে, যেগুলো সমাধান করে অনলাইন জাজে পরীক্ষা করা যায়। কিন্তু মুশকিল হচ্ছে ওই সমস্যাগুলো বাংলা ভাষায় লেখা নয়। তাই আমাদের অনেক শিক্ষার্থীরাই সেখানে আগ্রহ পায় না। এছাড়া প্রোগ্রামিংয়ে যারা একেবারে নতুন, তাদের জন্য আলাদাভাবে কোনো রিসোর্স আমার চোখে পড়েনি। তাই এই বইটি লেখার কথা আমার মাথায় আসে।

বইটি ষষ্ঠি শ্রেণি ও তার ওপরের যেকোনো ক্লাসের শিক্ষার্থীর জন্য উপযোগী। তবে তার আগে জানা থাকতে হবে সি প্রোগ্রামিং ভাষা। আমার "কম্পিউটার প্রোগ্রামিং ১ম খণ্ড" বইতে যতটুকু প্রোগ্রামিং দেখানো হয়েছে, এই বইয়ের সমস্যাগুলো সমাধানের জন্য সেটুকু প্রোগ্রামিং জানাই যথেষ্ট। বইটি পড়ার সময় অবশ্যই প্রতিটি সমস্যা আগে নিজে চেষ্টা করতে হবে। কয়েক ঘণ্টা একনাগাড়ে চেষ্টা করেও কেউ যদি না পারে, তবে সমাধান দেখে শিখে নিতে হবে। আর ওয়েবসাইটে গিয়ে নিজের সমাধান জমা দিয়ে যাচাই করার কাজটিও করা যাবে।

বইটি লিখতে আমার সহলেখক রিনি এবং রাফি অক্সান্ট পরিশ্রম করেছে। এছাড়া আমার ছাত্র তুষারও অনেক পরিশ্রম করে বইটি রিভিউ করেছে। আবার বইয়ের সমস্যাগুলোর কয়েকটি সিপিবুক ওয়েবসাইট থেকে নেওয়া যেগুলো তৈরি করেছেন ইকরাম মাহমুদ ফাহিম, তানভীরুল ইসলাম, শুভানন রায়িক প্রমুখ। তাদের পরিশ্রম সফল হবে যখন আমাদের স্কুল-কলেজের শিক্ষার্থীরা বিভিন্ন প্রোগ্রামিং প্রতিযোগিতায় দাপিয়ে বেড়াবে। সেই দিনের অপেক্ষায় রইলাম।

তামিম শাহরিয়ার সুবিন,
গ্র্যাব আরএন্ডডি সেন্টার, সিঙ্গাপুর।
ফেব্রুয়ারি ২০১৬।

প্রোগ্রামিং কনটেন্ট

৫২টি প্রোগ্রামিং সমস্যা ও সমাধান

১। জোড়-বিজোড় ১ ...	১৫
২। জোড়-বিজোড় ২ ...	১৯
৩। অধোগামী সংখ্যা ...	২১
৪। ভাজক	২৩
৫। বাক্স ১ ...	২৬
৬। যোগফল নির্ণয়	২৯
৭। সংখ্যা গণনা	৩১
৮। ছেট থেকে বড়	৩৪
৯। পূর্ণবর্গ সংখ্যা	৩৬
১০। রান রেট – ১ ...	৩৯
১১। গৌণিক বা ফ্যাক্টরিয়াল	৪২
১২। ফ্যাক্টরিয়াল ১০০	৪৪
১৩। টমি মিয়ার প্রোবাবিলিটি ...	৪৬
১৪। অক্ষরের ঘনঘটা ...	৪৮
১৫। অক্ষর গণনা	৫০
১৬। শব্দ বিপর্যয়	৫৩
১৭। স্বরবর্ণ গণনা ...	৫৭
১৮। স্বরবর্ণ – ব্যাঞ্জনবর্ণ	৫৮
১৯। শব্দ গণনা ১	৬১
২০। শব্দ গণনা ২	৬৩
২১। উল্টে দেখা	৬৫
২২। মৌলিক সংখ্যা ...	৬৭
২৩। বর্ণমালা থেকে সংখ্যা	৬৯
২৪। একান্তর উপাদান	৭২
২৫। লঘিষ্ঠ সাধারণ গুণিতক	৭৪
২৬। এলিয়েন গুপ্তি	৭৬
২৭। আর্মস্ট্রিং সংখ্যা	৭৯
২৮। এলোমেলো অ্যারে	৮১
২৯। চিহ্ন পরিচয়	৮৩

৩০। যোগ্য সংখ্যা ১ ...	৮৬
৩১। যোগ্য সংখ্যা ২ ...	৮৮
৩২। X-এর শুণিতক	৯১
৩৩। বিভাজনসাধ্য ১ ...	৯৪
৩৪। বিভাজনসাধ্য ২ ...	৯৬
৩৫। বৃত্তের বাইরে	৯৮
৩৬। শব্দ সাজানো	১০০
৩৭। সংখ্যা বিপর্যয়	১০২
৩৮। হীরক রাজ্য	১০৫
৩৯। প্যালিনড্রোম	১০৭
৪০। ধারার যোগফল ১	১০৯
৪১। ধারার যোগফল ২	১১২
৪২। ধারার যোগফল ৩	১১৪
৪৩। হিসাব-কিতাব ...	১১৬
৪৪। প্যাসকেলের ত্রিভুজ ১	১২০
৪৫। প্যাসকেলের ত্রিভুজ ২	১২৩
৪৬। ত্রিভুজের ক্ষেত্রফল	১২৬
৪৭। অ্যারের জোট	১২৭
৪৮। নিঁশোজ সংখ্যা	১২৯
৪৯। মৌলিক কি না	১৩৩
৫০। লেফট-রাইট	১৩৫
৫১। খোঁজ দ্য সার্চ ১ ...	১৩৬
৫২। খোঁজ দ্য সার্চ ২ ...	১৩৮
পরিশিষ্ট ...	১৪১
অনলাইন জাজ কী এবং কেন?	১৪৩
রং অ্যানসার (Wrong Answer) ...	১৪৬
প্রোগ্রামিং চর্চার কয়েকটি অনলাইন জাজ	১৪৮

কম্পিউটার বিজ্ঞানের একটি গুরুত্বপূর্ণ বিষয় হচ্ছে প্রোগ্রামিং, যা সফটওয়্যার নির্মাণ কৌশলেরও একটি গুরুত্বপূর্ণ অংশ। এই বিষয়টি অন্যান্য লেখাপড়ার মতো নয় যে বই পড়লাম, কিছু প্রশ্নের উত্তর শিখে ফেললাম, পরীক্ষা দিয়ে সব ভুলে গেলাম। প্রোগ্রামিং হচ্ছে একটি দক্ষতা (skill)। ব্যাপারটিকে সংগীতের সঙ্গে তুলনা করা যায়। সংগীত যেমন চর্চা করেই শিখতে হয়, তেমনি প্রোগ্রামিং চর্চার মাধ্যমেই ভালো প্রোগ্রামার হওয়া যায়, প্রোগ্রামিংয়ে উৎকর্ষ সাধন করা যায়।

কম্পিউটার প্রোগ্রামিং চর্চাকে উৎসাহ দেওয়ার জন্য সারা পৃথিবীতে বিভিন্ন ধরনের প্রোগ্রামিং প্রতিযোগিতার আয়োজন করা হয়। প্রাক-বিশ্ববিদ্যালয় পর্যায়ের শিক্ষার্থীদের জন্য সবচেয়ে বড় আয়োজনের নাম হচ্ছে ইনফরমেটিক্স অলিম্পিয়াড আর বিশ্ববিদ্যালয় পর্যায়ের শিক্ষার্থীদের অংশগ্রহণে সবচেয়ে বড় প্রতিযোগিতার নাম আইসিপিসি (ইন্টারন্যাশনাল কলেজিয়েট প্রোগ্রামিং কনটেক্ট)। আইসিপিসি'র মূল আয়োজক হচ্ছে এসিএম (ACM: অ্যাসোসিয়েশন অব কম্পিউটার মেশিনারিজ), তাই একসঙ্গে একে এসিএম আইসিপিসি বলা হয়ে থাকে।

এসিএম আইসিপিসি মূলত দুটি ধাপে অনুষ্ঠিত হয়, আঞ্চলিক প্রতিযোগিতা (রিজিওনাল কনটেক্ট) ও বিশ্ব চ্যাম্পিয়নশিপ (ওয়ার্ল্ড ফাইনালস)। অনেক জায়গায় আবার আঞ্চলিক প্রতিযোগিতার আগে অনলাইনে একটি বাছাই প্রতিযোগিতারও আয়োজন করা হয়। এসব প্রতিযোগিতায় বিভিন্ন বিশ্ববিদ্যালয় থেকে এক বা একাধিক দল অংশ নিতে পারে। আর হ্যাঁ, প্রতিযোগীদের কিন্তু কম্পিউটার সায়েন্সের শিক্ষার্থী হতে হবে এমন কোনো কথা নেই, বিশ্ববিদ্যালয়ের যেকোনো বিভাগের শিক্ষার্থীরা এতে অংশ নিতে পারে।

বাংলাদেশ ১৯৯৮ সাল থেকে নিয়মিত এসিএম আইসিপিসি'র চূড়ান্ত পর্ব অর্থাৎ ওয়ার্ল্ড ফাইনালসে অংশ নিয়ে আসছে। অনেকদিন ধরে অংশ নিলেও বাংলাদেশে প্রোগ্রামিং প্রতিযোগিতা এখনও খুব জনপ্রিয় হয়ে উঠতে পারেনি, সেটি প্রোগ্রামিং সম্পর্কে না জানার কারণেই হোক, কিংবা কষ্ট করে নতুন কিছু শেখার প্রতি তরুণ প্রজন্মের অনীহার কারণেই হোক। কিন্তু ডিজিটাল বাংলাদেশ বিনির্মাণে প্রোগ্রামিং প্রতিযোগিতা একটি বিশেষ গুরুত্ব বহন করে।

প্রোগ্রামিং প্রতিযোগিতায় আসলে কী হয়? এসিএম আইসিপিসি বা এ ধরনের প্রতিযোগিতায় তিনজন প্রোগ্রামার মিলে একটি দল হিসেবে অংশগ্রহণ করে। প্রতিটি দলকে দেওয়া হয় একটি কম্পিউটার, এক সেট প্রোগ্রামিং সমস্যা (৯ থেকে ১২ টি) এবং সেগুলো সমাধানের জন্য ৫ ঘণ্টা সময়। নির্দিষ্ট সময়ের মধ্যে যে দল সবচেয়ে বেশি সংখ্যক সমস্যার সমাধান করতে পারে, সে দল বিজয়ী হয়। সমান সংখ্যক সমস্যার সমাধান করলে যারা কম সময়ে করেছে এবং পেনাল্টি কম (সমাধান ভুল হলে ২০ মিনিট পেনাল্টি হয়), তারা rank-list-এ ওপরে থাকে। সমস্যাগুলোর

মধ্যে একটি সমস্যা এমন দেওয়া হয় যেন সেটি সব দলই সমাধান করতে পারে, আরেকটি থাকে এমন যা কোনো দলই সমাধান করতে না পারে।

আর স্কুল-কলেজ পর্যায়ের শিক্ষার্থীদের জন্য সবচেয়ে বড় প্রোগ্রামিং প্রতিযোগিতা হচ্ছে আইওআই (IOI) যার পূর্ণ অর্থ হচ্ছে International Olympiad in Informatics। আইওআইতে যেতে হলে বাংলাদেশ ইনফরমেটিক্স অলিম্পিয়াডে ভালো ফল করতে হবে, তাহলে ক্যাম্পে সুযোগ পাওয়া যাবে। ক্যাম্পে একাধিক প্রতিযোগিতার মাধ্যমে চূড়ান্ত দল নির্বাচন করা হয়। তবে ইনফরমেটিক্স অলিম্পিয়াডে কিন্তু দলীয় প্রতিযোগিতা নয়, একক প্রতিযোগিতা - অনেকটা গণিত অলিম্পিয়াডের মতো। এছাড়া ২০১৫ সাল থেকে বাংলাদেশে শুরু হয়েছে জাতীয় হাই স্কুল প্রোগ্রামিং প্রতিযোগিতা। এটিও ইনফরমেটিক্স অলিম্পিয়াডের মতো একক প্রতিযোগিতা। দেশের বিভিন্ন জায়গায় আঞ্চলিক পর্ব অনুষ্ঠিত হয়, তারপর চূড়ান্ত পর্ব। এসব প্রতিযোগিতায় সি ও সি প্লাস প্লাস - এই দুটি প্রোগ্রামিং ভাষা ব্যবহার করা যায়।

প্রোগ্রামিং সমস্যাগুলো তৈরি করেন কারা? যারা ইতিপূর্বে বিভিন্ন জাতীয় ও আন্তর্জাতিক প্রোগ্রামিং প্রতিযোগিতায় সাফল্য লাভ করেছেন, তাঁদের মধ্য থেকেই কয়েকজন প্রোগ্রামিং সমস্যাগুলো তৈরি করেন এবং প্রতিযোগিতার বিচারক হন। আর আমাদের শাহরিয়ার মঞ্চের তো এসিএম আইসিপিসি'র চূড়ান্ত পর্বের বিচারক, তাও গত ১২ বছরের বেশি সময় ধরে, যা আমাদের জন্য অত্যন্ত গর্বের বিষয়। প্রোগ্রামিং সমস্যা তৈরি হওয়ার পরে যিনি সমস্যাটি তৈরি করেন, তিনি সেটির সমাধান করার জন্য প্রোগ্রাম লেখেন এবং ডেটা সেট তৈরি করেন, যেগুলো দিয়ে সমাধান পরীক্ষা করা হবে। তারপর জাজ প্যানেলের আরেকজন সেই সমস্যাটির একটি বিকল্প সমাধান তৈরি করেন এবং সেটি জাজ ডেটা সেট দিয়ে পরীক্ষা করা হয়। সবকিছু ঠিকঠাক মতো হলেই সমস্যাটি প্রতিযোগিতার জন্য নির্বাচন করা হয়। এ সবই করা হয় প্রতিযোগিতা শুরু হওয়ার বেশ কিছুদিন আগে। প্রতিযোগিতার সময় কেউ যখন কোনো সমস্যা সমাধান করে জমা দেন, তখন জাজ ডেটা দিয়ে সেটি পরীক্ষা করা হয়। প্রোগ্রামটি যদি নির্দিষ্ট সময়ের মধ্যে রান করে সঠিক আউটপুট দেয়, তবেই সেটি সমাধান হয়েছে বলে গ্রহণ করা হয় এবং সঙ্গে সঙ্গে সেই দলকে সেটি জানিয়ে দেওয়া হয়। আবার সমাধান সঠিক না হলেও সেটি জানিয়ে দেওয়া হয়, যাতে তারা আবার চেষ্টা করতে পারেন। এসব কাজ করা হয় একটি সফটওয়্যার ব্যবহার করে।

প্রোগ্রামিং প্রতিযোগিতায় অংশ নিতে কী জানা লাগবে? প্রথমেই একটি প্রোগ্রামিং ভাষায় ওপর ভালো দখল প্রয়োজন। সি (C) বা সি প্লাস প্লাস (C++) এক্ষেত্রে সবচেয়ে ভালো। কারণ এ দুটি ভাষা মোটামুটি সব ধরনের প্রোগ্রামিং প্রতিযোগিতায় ব্যবহার করা হয়। অনেক ক্ষেত্রে জাভা ও ব্যবহার করা যায়, তবে কিছু কিছু সমস্যার সমাধান জাভা দিয়ে লেখা হলে সেগুলো রান করতে বেশি সময় লাগে। সি এর চেয়ে সি প্লাস প্লাস ব্যবহার করা বুদ্ধিমানের কাজ এই জন্য যে, সেখানে এসটিএল (STL : স্ট্যান্ডার্ড টেম্পলেট লাইব্রেরি) বলে খুব কার্যকরী লাইব্রেরি আছে, যেটির ব্যবহার কিছু কিছু ক্ষেত্রে অনেক সময় বাঁচিয়ে দেয়। তবে সি এবং সি প্লাস প্লাস-এর মধ্যে মিল অনেক

বেশি এবং সি দিয়ে প্রোগ্রামিং শেখা শুরু করলে সি প্লাস প্লাস শিখতে খুব একটা সময় লাগবে না। প্রোগ্রামিং প্রতিযোগিতায় ভালো করতে হলে দুটি জিনিস পাশাপাশি চালিয়ে যেতে হয়, প্রবলেম সলভিং ও লেখাপড়া।

প্রবলেম সলভ করার জন্য ভালো কিছু ওয়েবসাইট রয়েছে। বইয়ের পরিশিষ্ট অংশে একটি তালিকা দেওয়া আছে। এসব ওয়েবসাইটে প্রবলেম সলভিংয়ের চর্চা তো করা যায়ই, সেই সঙ্গে বিভিন্ন অনলাইন প্রতিযোগিতায় অংশ নেওয়া যায়। অনেকের ভুল ধারণা থাকে যে সব লেখাপড়া শেষ করে তার পরে প্রতিযোগিতায় অংশ নেওয়া শুরু করবে, সেটি ঠিক নয়। দুটি কাজই একসঙ্গে চালিয়ে যেতে হবে।

বিভিন্ন ওয়েবসাইটে প্রবলেম সলভ করার সময় অনেকে একই রকম অনেক প্রবলেম সলভ করে। সেটি না করে বিভিন্ন রকম প্রবলেম সলভ করার পেছনে সময় দেওয়াটাই শ্রেয়। অনেক সময় কঠিন কিছু প্রবলেমের পেছনে বেশ কয়েকদিন লেগে থাকতে হয়। সমস্যা সমাধানের জন্য এই লেগে থাকার ব্যাপারটা বিরক্তিকর ঠেকালেও হতাশ হওয়া চলবে না।

৫২টি প্রোগ্রামিং সমস্যা ও সমাধান

সমস্যা ১ – জোড়-বিজোড় ১

সমস্যার বিবরণ

কোনো একটি পূর্ণসংখ্যা দেওয়া থাকবে, সেটি জোড় না বিজোড় তা বের করতে হবে।
[\(<http://bit.ly/1-52-problem-1>\)](http://bit.ly/1-52-problem-1)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) দেওয়া থাকবে। পরবর্তীতে T এর মান যত, ততটি লাইন থাকবে। প্রতিটি লাইনে একটি করে পূর্ণসংখ্যা n ($0 \leq n \leq 2147483647$) দেওয়া থাকবে।

আউটপুট

প্রতিটি পূর্ণসংখ্যার জন্য, সংখ্যাটি জোড় হলে even আর বিজোড় হলে odd কথাটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	even
100	even
0	odd
1111	

সমাধান

জোড়-বিজোড় সংখ্যার সঙ্গে ছোটবেলা থেকেই তোমরা পরিচিত। সমস্যার বর্ণনায় প্রথমেই বলা আছে কোনো একটি সংখ্যা দেওয়া থাকবে সেটি জোড় না বিজোড় তা বের করতে হবে। খুবই সহজ একটি সমস্যা (মানে সমাধান সহজ আর কী)।

যেকোনো প্রোগ্রামিং সমস্যা সমাধান করার আগে সেটি মনোযোগ দিয়ে পড়ে নিতে হবে। প্রথমেই, ইনপুট/আউটপুটের দিকে খেয়াল কর। ইনপুটে বলা আছে প্রথম লাইনে একটি পূর্ণ সংখ্যা T থাকবে। T এর মান যত পরবর্তী ঠিক ততটি লাইনে একটি করে পূর্ণ সংখ্যা n দেওয়া থাকবে। এখানে দেখ, পূর্ণ সংখ্যাটির একটি রেঞ্জ দেওয়া আছে 0 থেকে 2147483647-এর মধ্যে। তোমরা যে int টাইপের ভ্যারিয়েবল ব্যবহার কর সেটির সর্বোচ্চ মান হচ্ছে 2147483647। তার মানে int টাইপের ভ্যারিয়েবল ব্যবহার করলেই চলবে।

উদাহরণ ইনপুটে দেখ, প্রথম লাইনে দেওয়া আছে 3 এবং পরবর্তী তিনটি লাইনে তিনটি আলাদা আলাদা সংখ্যা, 100, 0, 1111 দেওয়া আছে। এবার আউটপুটে দেখ, প্রথম সংখ্যা 100-এর জন্য আউটপুটে আছে even, দ্বিতীয় সংখ্যা 0-এর জন্য আউটপুটে দেওয়া আছে even, তৃতীয় সংখ্যা 1111-এর জন্য আউটপুট দেওয়া আছে odd।

এবার আমাদের কোড লেখার পালা। প্রথমেই `main()` ফাংশনে ভ্যারিয়েবল ডিক্লেয়ার করে নিতে হবে।

```
int T, i, n;
```

T ব্যবহার করব প্রথম ইনপুট নেওয়ার জন্য (যেটি নির্দেশ করবে যে এর পর আর কয়টি সংখ্যা আছে), i ব্যবহার করব `for` লুপের ইনডেক্স হিসেবে আর n হচ্ছে আমরা T-সংখ্যক লাইনে যে সংখ্যাগুলো ইনপুট হিসেবে নেব সে জন্য।

এ জন্য আমরা T-এর মান ইনপুট নেব।

```
scanf("%d", &T);
```

for লুপটি T সংখ্যকবার চলবে।

```
for(i = 1; i <= T; i++) {}
```

লুপ চলার সময় প্রতিবার আমরা n-এর মান ইনপুট নিব।

```
scanf("%d", &n);
```

যদি n দুই দ্বারা বিভাজ্য হয় তাহলে প্রিন্ট করব even। অন্যথায় প্রিন্ট করব odd।

```
if(n % 2 == 0)
{
    printf("even\n");
}
else
{
    printf("odd\n");
}
```

সমস্যার বর্ণনায় নিউলাইন প্রিন্ট করার কথা বলা না থাকলেও আমি দুটি printf()-এর শেষে একটি করে নিউলাইন প্রিন্ট করেছি। কারণ, লক্ষ করে দেখো উদাহরণ আউটপুটে even, odd এগুলো প্রতিটি নতুন লাইনে প্রিন্ট করা আছে।

মনে রাখতে হবে, সমস্যার বর্ণনায় যেভাবে বলা আছে ঠিক সেভাবেই আউটপুট প্রিন্ট করতে হবে। অনেক সময় দেখা যায় তোমরা নিজেদের ইচ্ছামতো "This is an even number" এভাবে প্রিন্ট করো। আবার অনেক সময় দেখা যায় Even, Odd এভাবে প্রিন্ট করো। এভাবে করা যাবে না। আউটপুটে সব ছোট হাতের অক্ষরে থাকলে ছোট হাতের অক্ষরেই প্রিন্ট করতে হবে, নইলে উভর ভুল হবে। কারণ হচ্ছে তুমি যে প্রোগ্রাম জমা দিবে, অনলাইন জাজ সফটওয়্যার, তোমার প্রোগ্রামটি চালাবে একটি নির্দিষ্ট ইনপুট ফাইল ব্যবহার করে (তাই বলে তোমাকে কিন্তু প্রোগ্রামে ফাইল নিয়ে কাজ করতে হবে না)। তারপর তোমার প্রোগ্রাম যেই আউটপুট দিবে, সেগুলোকেও একটি ফাইলে লেখা হবে। তারপর তোমার আউটপুটের সঙ্গে জাজের আউটপুট তুলনা করা হবে আরেকটি প্রোগ্রাম ব্যবহার করে। জাজের আউটপুট মানে হচ্ছে যিনি প্রোগ্রামিং সমস্যাটি তৈরি করেছেন তিনি একটি ইনপুট ফাইল ও সেই ইনপুট ফাইলের জন্য একটি আউটপুট ফাইল তৈরি করে রেখেছেন। এখন তোমার আউটপুট যদি জাজের আউটপুটের সঙ্গে ১০০% মিলে, তাহলেই কেবল তোমার উভর সঠিক বলে ধরে নেওয়া হবে। নইলে রং অ্যানসার! তুমি যদি তোমার প্রোগ্রামে কোনো অতিরিক্ত স্পেস প্রিন্ট করো, ছোটহাত-বড়হাতের অক্ষরে গড়বড় করে ফেলো, নিউলাইন প্রিন্ট করতে ভুলে যাও, তাহলে কিন্তু সমাধান সঠিক হবে না। প্রোগ্রামিং প্রতিযোগিতা - সেটা অনলাইনে হোক কিংবা সরাসরি, সেখানকার জাজ সফটওয়্যারটি বড়ই নির্দয়, বড়ই নিষ্ঠুর।

এবারে পুরো প্রোগ্রামটি লিখে ফেলি:

```
#include<stdio.h>

int main()
{
    int T, i, n;

    scanf("%d", &T);

    for(i = 1; i <= T; i++)
    {
        scanf("%d", &n);

        if (n%2 == 0)
        {
            printf("even\n");
        }
        else
```

```
    {
        printf("odd\n");
    }
}

return 0;
}
```

সমাধানটি এখন উপরে দেওয়া লিঙ্কে গিয়ে জমা দাও। প্রোগ্রামিং ল্যাস্যুলেজ সি নির্বাচন করবে আর কোড পেস্ট করে সাবমিট করার বদলে ফাইল আপলোড করে সাবমিট করবে। সমস্যাটির নিচে দেওয়া Answer বাটন ক্লিক করার পর তোমরা নিচের ছবির মতো দেখতে পাবে। ফাইল সাবমিট করার জন্য "Upload a file instead" বাটন, ল্যাস্যুলেজ হিসেবে সি default দেওয়া আছে এবং author এর জায়গায় নিজের নাম দিতে হবে।

জোড়-বিজোড়-১

କୋଣେ ଏହାଟି ପର୍ମାନ୍ତମ୍ଭା ଦେଖେଯା ଥାକଲେ ଶେଷି ଜୋଡ଼ ନା ବିଜୋଡ଼ ତା ସେଇ କରନ୍ତେ ହବେ ।

<p>জোড়-বিজোড়-১</p> <p>কোনো একটি পূর্ণসংখ্যা দেওয়া থাকলে সেটি জোড় না বিজোড় তা বের করতে হবে।</p> <p>Paste the answer here :</p>	<input type="button" value="Upload a file instead"/>
<div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Language: </div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> Author: </div> <div style="display: flex; justify-content: space-between;"> <div style="flex: 1;"> <input type="checkbox"/> Reserve this nick for me </div> <div style="flex: 1;"> <input checked="" type="checkbox"/> Give a secret phrase </div> </div>	

তোমরা চাইলে for লুপের পরিবর্তে while লুপও ব্যবহার করতে পারো এভাবে: `while(T--)`, সেক্ষেত্রে i ভ্যারিয়েবলটি ব্যবহার না করলে চলবে।

তোমরা যদি খুশি হয়ে যাও এই ভেবে যে এই বইয়ে সব সমস্যার সমাধানগুলোর কোড ১০০%
লিখে দেওয়া হবে, তাহলে তোমাদের জন্য দুঃসংবাদ। মাঝে মধ্যে পুরো সমাধান লেখা হবে, তবে
অনেক সময়ই আমরা পুরো সমাধানের অক্ষিয়াটি বলে দেব, তোমাদের কেবল চূড়ান্ত কোডটি
লিখতে হবে। সবচেয়ে ভালো হয় প্রতিটি প্রবলেম পড়ার পর সেটা সমাধানের জন্য কমপক্ষে পাঁচ
ঘণ্টা চেষ্টা করবে, যদি না পারো, তাহলে আলোচনা বা সমাধান দেখবে।

সমস্যা ২ – জোড়-বিজোড় ২

সমস্যার বিবরণ

কোনো একটি পূর্ণসংখ্যা দেওয়া থাকবে, সেটি জোড় না বিজোড় তা বের করতে হবে।
[\(<http://bit.ly/1-52-problem-2>\)](http://bit.ly/1-52-problem-2)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। পরবর্তীতে T সংখ্যক লাইনে একটি করে অখণ্ডাত্মক (Non-negative) পূর্ণসংখ্যা n দেওয়া থাকবে। একটি সংখ্যায় সর্বোচ্চ 100টি অঙ্ক (digit) থাকতে পারে।

আউটপুট

প্রতিটি পূর্ণসংখ্যার জন্য, সংখ্যাটি জোড় হলে even আর বিজোড় হলে odd প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	even
100	even
0	odd
1111	

সমাধান

তোমরা যদি আগের সমস্যাটির সমাধান করে থাকো, তাহলে সেটি আবার জমা দাও। উত্তর কী পেলে? রং অ্যানসার?

আগের সমস্যার সঙ্গে এই সমস্যাটির পার্থক্য কোথায়? মাত্র একটি জায়গায়। বলা আছে যে সংখ্যাটি জোড় না বিজোড় বের করতে হবে, সেটিতে সর্বোচ্চ 100-টি অঙ্ক থাকতে পারে। 100-টি অঙ্ক মানে কিন্তু অনেক বড় সংখ্যা। int ভ্যারিয়েবল হচ্ছে 32 বিট, তার মানে এই ধরনের ভ্যারিয়েবলে 2^{32} -টি সংখ্যা রাখা যায়। 2^{32} মানে 4294967296। তাহলে আমরা যদি signed int অর্থাৎ ঋণাত্মক ও ধনাত্মক উভয় ধরনের সংখ্যাই রাখি, তাহলে আমরা -2^{31} থেকে $2^{31}-1$ পর্যন্ত সংখ্যা (-2147483648 থেকে 2147483647) রাখতে পারব। -2147483648 থেকে -1 পর্যন্ত 2147483648 -টি আর 0 থেকে 2147483647 পর্যন্ত 2147483648 -টি সংখ্যা (0 থেকে শুরু, তাই 2147483648 -এর বদলে 2147483647)। মোট 4294967296-টি সংখ্যা।

আর যদি `unsigned int` ব্যবহার করো, তাহলে 0 থেকে 4294967295 পর্যন্ত সংখ্যা রাখা যাবে। তোমাদের মনে রাখার সুবিধার জন্য এখানে একটা চার্ট দিয়ে দিলাম।

ডেটা টাইপ	সর্বনিম্ন সীমা বা লিমিট	সর্বোচ্চ সীমা বা লিমিট	মোট সংখ্যা
<code>signed int</code>	-2^{31} $= -2147483648$	$2^{31}-1$ $= 2147483647$	4294967296 টি
<code>unsigned int</code>	0	$2^{32}-1$ $= 4294967295$	4294967296 টি

আরেক ধরনের ডেটা টাইপ আছে, যাকে বলে `long long int`। এর ফরয়েট স্পেসিফিকেশন হচ্ছে `%lld` (প্রথম দুইটা এল)। এটি 64 বিট। এখন তোমরা বের করো, এই ভ্যারিয়েবলে কত পর্যন্ত সংখ্যা রাখা যাবে। দেখবে 19 অঙ্কের বেশি নয়। কিন্তু আমাদের দরকার 100 অঙ্কের সংখ্যা রাখতে পারে, এমন ভেরিয়েবল!

আসলে এখানে স্ট্রিং ব্যবহার করলেই হয়। আর জোড় না বিজোড়, সেটি বোধ যাবে স্ট্রিংের শেষ ক্যারেক্টারটি বা বর্ণটি পরীক্ষা করলেই। তাহলে আমাদের প্রোগ্রামটি দাঁড়াবে এ রকম:

```
#include<stdio.h>

int main()
{
    int T, i;
    char n [101];

    scanf("%d", &T);

    for(i = 1; i <= T; i++)
    {
        scanf("%s", &n);

        // write your code here
    }

    return 0;
}
```

তাহলে প্রোগ্রাম ঠিকঠাকভাবে লিখে ওয়েবসাইট শিয়ে জমা দাও।

সমস্যা ৩ – অধোগামী সংখ্যা

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি 1 থেকে 1000 পর্যন্ত সবগুলো সংখ্যাকে বড় থেকে ছোট ক্রমানুসারে প্রিন্ট করবে।

(<http://bit.ly/1-52-problem-3>)

ইনপুট

প্রোগ্রামটিতে কোনো ইনপুট নেই।

আউটপুট

প্রতিটি লাইনে মোট পাঁচটি (5) করে সংখ্যা থাকবে এবং প্রতিটি সংখ্যা একটি '\t' (Tab) ক্যারেক্টার দিয়ে আলাদা করা থাকবে।

উদাহরণ

আউটপুট				
1000	999	998	997	996
995	994	993	992	991
990	989	988	987	986
...
20	19	18	17	16
15	14	13	12	11
10	9	8	7	6
5	4	3	2	1

সমাধান

এই সমস্যাটি তুলনামূলক একটি সহজ সমস্যা। এখানে 1000 থেকে 1 পর্যন্ত সংখ্যা প্রিন্ট করার কথা বলা হয়েছে। সমস্যার বর্ণনায় বলা আছে প্রোগ্রামটির কোনো ইনপুট নেই। এক্ষেত্রে আমাদের কাজ হলো শুধুমাত্র আউটপুট প্রিন্ট করা।

একটিমাত্র লুপ ব্যবহার করেই আমরা এই কোডটি করতে পারি। এক্ষেত্রে আমরা একটি `for` লুপের সাহায্য নির। লুপটির মান শুরু হবে 1000 থেকে এবং এক এক করে মান কমে লুপটি 1 এ এসে থেমে যাবে। লুপটি হবে এমন:

```
for(int i = 1000; i >= 1; i--) {}
```

আমাদের এখানে i এর মান প্রিন্ট করতে হবে। এ জন্য আমরা printf() ফাংশন ব্যবহার করব। i এর মান প্রিন্ট করার সময় আমাদের খেয়াল করতে হবে দুটি বিষয়:

1. আউটপুটে বলা আছে, প্রতি লাইনে 5 টি করে সংখ্যার কথা।
2. প্রতিটি সংখ্যা একটি ট্যাব ক্যারেন্টোর দিয়ে আলাদা করা থাকবে।

প্রথমেই আমরা চিন্তা করি 5টি করে সংখ্যার এক লাইনে থাকার ব্যাপারটা। আমরা জন্য লুপের শুরুতেই আরো একটি int ভ্যারিয়েবল ব্যবহার করতে পারি, যেটি এক এক করে বাড়তে থাকবে। যখন এই ভ্যারিয়েবলের মান 5 দ্বারা নিঃশেষে বিভাজ্য হবে তখন আমরা একটি নিউলাইন প্রিন্ট করব। তবে আলাদা ভ্যারিয়েবল ব্যবহার না করে, শুধুমাত্র i এর মান থেকেই এই সিন্ক্রান্ট নেওয়া যায়। সেটা কীভাবে করা যায় তা তোমরা একটু চিন্তা করলেই বুঝে ফেলবে।

দ্বিতীয়ত, বলা আছে একটি ট্যাব ক্যারেন্টোর কথা। এখানে আমরা যখন i-এর মান প্রিন্ট করব তখন printf() ফাংশনটি লিখব এভাবে:

```
printf("%d\t", i);
```

এখন তোমরা কোডটি গুছিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। তারপর সমস্যার লিঙ্কে গিয়ে সাবমিট করবে।

সমস্যা ৪ – ভাজক

সমস্যার বিবরণ

একটি সংখ্যার সব শুণনীয়ক (ভাজক) বের করতে হবে।

(<http://bit.ly/1-52-problem-4>)

ইনপুট

প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 10$)। এরপরের পরবর্তী T সংখ্যক লাইনে একটি করে পূর্ণ সংখ্যা N থাকবে, যেখানে $1 \leq N \leq 100000$ ।

আউটপুট

প্রতিটি কেসের জন্য একটি করে লাইন প্রিন্ট করতে হবে, শুরুতে কেস নম্বর দিতে হবে। এরপর N এর সব শুণনীয়ক ছোট থেকে বড় আকারে প্রিন্ট করতে হবে এবং প্রতিটি শুণনীয়ক শুধুমাত্র একবার প্রিন্ট করতে হবে। শুণনীয়কগুলোকে শুধুমাত্র একটি স্পেস দিয়ে আলাদা করতে হবে এবং লাইনের শেষে কোনো অতিরিক্ত স্পেস থাকবে না।

উদাহরণ

ইনপুট	আউটপুট
3	Case 1: 1 2 3 6
6	Case 2: 1 3 5 15
15	Case 3: 1 23
23	

সমাধান

একটি সংখ্যার সবগুলো শুণনীয়ক বা ভাজক বের করার সমস্যা এটি। আমরা জানি, কোনো একটি সংখ্যাকে যদি অপর এক বা একাধিক সংখ্যা দিয়ে নিঃশেষে ভাগ করা যায়, অর্থাৎ ভাগ করার পর ভাগশেষ শূন্য হয়, তাহলে ওই সংখ্যাগুলোকে সেই সংখ্যাটির শুণনীয়ক বলা হয়।

সমস্যাটির উদাহরণ ইনপুটের একটি সংখ্যা 6। 6 সংখ্যাটি 1, 2, 3 এবং 6 দ্বারা নিঃশেষে বিভাজ্য। তাই 1, 2, 3 এবং 6 হচ্ছে 6-এর শুণনীয়ক।

এবার আমরা আসি ইনপুটের বর্ণনায়। প্রথম ইনপুট হবে টেস্ট কেস T এবং পরবর্তী T সংখ্যক লাইনে ইনপুট হবে একটি করে পূর্ণসংখ্যা N । পূর্ণসংখ্যা N -এর মান সর্বোচ্চ হতে পারে

100000। আউটপুটের বর্ণনায় বলা আছে প্রতিটি কেসের জন্য একটি করে লাইনে আউটপুট প্রিন্ট করতে হবে। প্রতিটি শুণনীয়ক একবার করে একটি করে স্পেস দিয়ে প্রিন্ট করতে হবে।

এবারে আমাদের কোড লেখার পালা। প্রথমেই আমরা আমাদের প্রয়োজনীয় ভ্যারিয়েবল আমরা ডিক্লেয়ার করে নিব।

```
int T, i, N, j;
```

এখানে,

- T- টেস্ট কেসের জন্য।
- i- আমরা প্রতি লাইনে ইনপুট নেওয়ার জন্য যে লুপ ব্যবহার করব তার জন্য,
- N -আমরা যে সংখ্যাটির ভাজক বের করতে চাই সেটির জন্য এবং
- j- শুণনীয়ক বের করার জন্য আমাদের দ্বিতীয় আরেকটি লুপের দরকার হবে তার জন্য।

এবারে আমাদের ইনপুট নেওয়ার পালা। যেহেতু, T সংখ্যক লাইনে ইনপুট নিতে হবে তাই প্রথমেই আমরা T-এর মান ইনপুট নিব।

```
scanf("%d", &T);
```

এবারে আমাদের T সংখ্যক সংখ্যা ইনপুট নিতে হবে। আমরা এর জন্য একটি লুপ ব্যবহার করবো। লুপটি T সংখ্যকবার চলবে। লুপটি হবে এমন:

```
for(i = 1; i <= T; i++) {}
```

লক্ষ করো, লুপটি আমরা 1 থেকে শুরু করে T পর্যন্ত চালাব, 0 থেকে T-1 পর্যন্ত নয়। আমাদের Case 1, Case 2 এভাবে আউটপুট প্রিন্ট করতে হবে, তাই আমরা লুপটা 1 থেকেই শুরু করেছি। লুপটি যতবার চলবে ততবার আমরা N ইনপুট নিব।

```
scanf("%d", &N);
```

Case প্রিন্ট করার কাজটা আমরা N ইনপুট নেওয়ার পরের লাইনেই করে ফেলব।

```
printf("Case %d:", i);
```

শুণনীয়ক বের করার জন্য এখন আমরা আরেকটি লুপের সাহায্য নিব। লুপটি 1 থেকে শুরু করে N-এর মান পর্যন্ত চলবে (অবশ্যই N -এর মানসহ)। N সংখ্যাটি 1 থেকে N পর্যন্ত যতগুলো সংখ্যা দিয়ে নিঃশেষে বিভাজ্য সেই সংখ্যাগুলো খুঁজে বের করে প্রিন্ট করতে হবে। বলা আছে

গুণনীয়কগুলো একটি স্পেস ক্যারেক্টার দিয়ে আলাদা থাকবে, তাই প্রিন্ট ফাংশনে একটি বাড়তি স্পেস ক্যারেক্টার দিতে হবে।

```
for(j = 1; j <= num; j++)
{
    if(num%j == 0)
    {
        printf(" %d", j);
    }
}
```

যেহেতু প্রতিটি টেস্ট কেস আলাদা আলাদা লাইনে আউটপুটের উদাহরণে দেওয়া আছে, তাই দ্বিতীয় for লুপটি শেষ হওয়ার পর আমরা একটি নিউলাইন প্রিন্ট করব।

আমাদের সম্পূর্ণ কোডটি হবে এমন:

```
#include<stdio.h>
int main()
{
    int T, i, N, j;
    scanf("%d", &T);

    for(i = 1; i <= t; i++)
    {
        scanf("%d", &N);
        printf("case %d:", i);

        for(j = 1; j <= N; j++)
        {
            if(N%j == 0)
            {
                printf(" %d", j);
            }
        }

        printf("\n");
    }

    return 0;
}
```

এখন তোমরা সমাধান অনলাইন জাজে জমা দাও। যদি দেখ রং অ্যানসার, তাহলে চেষ্টা করো প্রোগ্রামের বাগ বের করতে।

হিন্টস্ দিয়ে দেই, কম্পিউটারের কাছে কিন্তু 'a' আর 'A', দুটি আলাদা জিনিস।

আরও একটি বিষয় বলে দেই, আমরা এখানে একটি লুপ ব্যবহার করে সবগুলো ভাজক বের করেছি। তোমরা সমস্যার বিবরণ পড়লে দেখতে পাবে, N এর মান সর্বোচ্চ 100000 পর্যন্ত হতে পারে। অর্থাৎ, তখন আমাদের এই লুপটি 100000 বার পর্যন্ত চলবে। এবারে তোমরা একটু চিন্তা করে দেখো তো, এর চেয়ে কম সংখ্যক বার লুপটি চালিয়েই কি এই সমাধানটা করা যেত? একটি সংখ্যার যতগুলো ভাজক থাকে সাধারণত তার অর্ধেক সংখ্যক থাকে তার বর্গমূলের চেয়ে ছোট, আর বাকি অর্ধেক থাকে তার বর্গমূলের চেয়ে বড়। আমরা পরবর্তী সমস্যাগুলোতে দেখব, কীভাবে বর্গমূল ও মৌলিক সংখ্যা বের করা যায় এবং কীভাবে এগুলোর সাহায্য নিয়ে আরও কম সময়ে ভাজক বের করা যায়।

সমস্যা ৫ – বাক্স ১

সমস্যার বিবরণ

তোমার হাতে যথেষ্ট পরিমাণে কাজকর্ম নেই দেখে তোমাকে একটি বাক্স আঁকার কাজ দেওয়া হলো। আসলে ব্যাপারটি তেমন কিছু কঠিন নয়, তোমাকে বর্গের একটি বাহর দৈর্ঘ্য বলা হবে আর তুমি চট করে '*' অক্ষরটি ব্যবহার করে ওই বর্গটি এঁকে ফেলবে। বর্গের কেবল বাহু আঁকলেই হবে না, তেতরের ঘরগুলোও '*' অক্ষরটি দিয়ে পূর্ণ করে দিতে হবে। যেহেতু তুমি প্রোগ্রামিং শেখা শুরু করেছ এবং লুপ পর্যন্ত শিখে ফেলেছ, তাই তুমি কাজটি করবে একটি প্রোগ্রাম লিখে। (<http://bit.ly/1-52-problem-5>)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 25$) থাকবে এবং তারপরে T-সংখ্যক লাইন থাকবে। প্রতিটি লাইনে একটি করে সংখ্যা (N) থাকবে যার মান 1 থেকে 80-এর মধ্যে হবে।

আউটপুট

প্রতিটি N-এর জন্য $N \times N$ বর্গ আঁকতে হবে। পুরো বর্গটি ' $*$ ' দিয়ে পূর্ণ করে দিতে হবে। প্রতিটি বর্গ একটি ফাঁকা লাইন দিয়ে পৃথক করে দিতে হবে। পৃথক করার কাজে ব্যতিত অন্য কোথাও অতিরিক্ত ফাঁকা লাইন বা স্পেস রাখা যাবে না।

উদাহরণ

ইনপুট	আউটপুট
3	*
1	
3	***
5	***

সমাধান

সমস্যাটি বেশ মজার। ' $*$ ' অক্ষর ব্যবহার করে বাক্স বানানো, তাও আবার প্রোগ্রামিংয়ের সাহায্যে। এই প্রোগ্রামটিতে ' $*$ ' অক্ষর ব্যবহার করে একটি চারকোণা বাক্স আউটপুটে দেখাতে হবে। ইনপুটের বর্ণনা আগের সমস্যাগুলোর মতোই। সেই সমাধানগুলো তোমরা দেখেছ, করেছ। আশা করি ইনপুটের বর্ণনা পড়ে কীভাবে ইনপুট নিতে হবে সেটা এখন তোমরা জানো।

আউটপুটের বর্ণনায় বলে দেওয়া আছে একটি $N \times N$ বর্গ আঁকতে হবে। পুরো বর্গটি ' $*$ ' দিয়ে পূর্ণ করে দিতে হবে। প্রতিটি বর্গ একটি ফাঁকা লাইন দিয়ে পৃথক করা থাকবে।

এখানে একটি বর্গ প্রিন্ট করার জন্য আমরা লুপের ভেতরে লুপ ব্যবহার করব। প্রোগ্রামিংয়ের ভাষায় আমরা একে বলি নেস্টেড লুপ। বলা আছে, $N \times N$ মানে লম্বালম্বি এবং আড়াআড়ি দুদিকেই N সংখ্যক ' $*$ ' থাকবে। তোমরা বর্গের মূল ধারণাটা যদি চিন্তা করো তাহলে এই ব্যাপারটি তোমাদের কাছে পরিষ্কার হবে।

যে সংখ্যাটি ইনপুট হিসেবে নেওয়া হবে সেই সংখ্যাটি (N) নির্দেশ করবে, বাক্সটিতে আড়াআড়ি এবং লম্বালম্বি করটি ' $*$ ' থাকবে।

যেমন: যখন N এর মান উদাহরণ ইনপুটে 3 তখন আউটপুটে বাক্সটি হবে এমন:

তিনটি লাইনে আলাদা আলাদা করে তিনটি '*' প্রিন্ট করা আছে।

নেস্টেড for লুপের গঠনটা এমন:

```
outer loop()
{
    inner loop()
    {
        // necessary code;
    }
}
```

outer এবং inner লুপের জন্য আলাদা আলাদা ভ্যারিয়েবল ডিক্রেয়ার করতে হবে।

```
int i, j;
```

প্রতিটি লুপ N এর মান যত ঠিক ততবার করে চলবে এবং '*' প্রিন্ট করবে। এছাড়া প্রতিটি কেসের সংখ্যা ইনপুট নেওয়ার জন্য আমাদের আরো একটি লুপের দরকার পড়বে। ঠিকমতো কোড লিখতে পারলে তোমরা বাক্স-১ সমস্যাটির সমাধান করতে পারবে। কোড লেখার সময় আউটপুটের বর্ণনা ভালো করে পড়ে নিবে।

সমস্যা ৬ – যোগফল নির্ণয়

সমস্যার বিবরণ

পাঁচ (5) অঙ্কের একটি সংখ্যার প্রথম এবং শেষ অঙ্কের যোগফল নির্ণয় করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-6>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 10000$), যা মোট কতটি পাঁচ অঙ্কের সংখ্যা ইনপুট নিতে হবে তা নির্দেশ করবে। পরবর্তীতে T সংখ্যক লাইনে একটি করে পূর্ণসংখ্যা N ($0 \leq N \leq 65535$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে N এর প্রথম এবং শেষ অঙ্কের যোগফল প্রিন্ট করতে হবে নিচের ছকে যেভাবে দেখানো আছে ঠিক সেই ফরম্যাটে। প্রতিটি আউটপুটের পরে একটি নিউলাইন প্রিন্ট করতে হবে। আউটপুটে "≡" আগের এবং পরের স্পেসটি লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
5	Sum = 6
12345	Sum = 14
56789	Sum = 1
14310	Sum = 3
10062	Sum = 16
96587	

সমাধান

তোমরা সবাই নিশ্চয়ই সংখ্যা ও অঙ্কের পার্থক্য জানো? 162 একটি সংখ্যা যার এককের ঘরের অঙ্কটি হচ্ছে 2, দশকের ঘরের অঙ্কটি হচ্ছে 6 আর শতকের ঘরের অঙ্কটি হচ্ছে 1। আর প্রথম ও শেষ অঙ্ক বলতে কী বোঝানো হচ্ছে? গণিতে সংখ্যার প্রথম অঙ্ক বলতে বুঝায় তার সর্ববামের অঙ্কটি, যাকে ইংরেজীতে বলে Most Significant Digit (MSD)। আর শেষ অঙ্ক বলতে বুঝায় সর্বডানের অঙ্কটি এককের ঘরের অঙ্কটি, যাকে ইংরেজীতে বলে Least Significant Digit (LSD)।

সমস্যার বর্ণনায় বলা আছে পাঁচ অঙ্কের একটি সংখ্যার প্রথম এবং শেষ অঙ্কের যোগফল বের করতে হবে। উদারহণস্বরূপ আমরা নমুনা আউটপুটের প্রথম সংখ্যা 12345 বিবেচনায় আনি। খেয়াল করে দেখো, আমরা যদি এই সংখ্যাটিকে 10 দিয়ে ভাগ করি তাহলে আমাদের ভাগশেষ থাকবে 5, যা সংখ্যাটির সর্বশেষ অঙ্ক (LSD)। তাহলে আমরা আমাদের সংখ্যাটির শেষ অঙ্কটি পেয়ে গেলাম। এবার আমাদের সংখ্যাটির প্রথম অঙ্কটি বের করার পালা। দেখি সেটি কীভাবে করা যায়।

12345-কে 10 দিয়ে ভাগ করলে ভাগফল হচ্ছে 1234 আর ভাগশেষ 5।

1234-কে 10 দিয়ে ভাগ করলে ভাগফল 123 আর ভাগশেষ 4।

123-কে 10 দিয়ে ভাগ করলে ভাগফল 12 আর ভাগশেষ 3।

12-কে 10 দিয়ে ভাগ করলে ভাগফল 1 আর ভাগশেষ 2।

1-কে 10 দিয়ে ভাগ করলে ভাগফল 0 আর ভাগশেষ 1।

যখনই আমাদের ভাগফল 0 হবে, তখন ভাগশেষ যেই সংখ্যাটা থাকবে, সেটি এক অঙ্কের সংখ্যা (যেহেতু 10 দিয়ে ভাগ করা হচ্ছে) আর সেটিই হচ্ছে আমাদের কাঞ্চিত প্রথম অঙ্ক (MSD)।

তাহলে আমরা প্রোগ্রাম লিখব কীভাবে? একটি int ভেরিয়েবলে পাঁচ অঙ্কের সংখ্যাটি ইনপুট নেব (ধরা যাক, N), তারপর 10 দিয়ে মড (মডুলাস অপারেটর, %) করে ভাগশেষ বের করব, যেটি হচ্ছে সংখ্যাটির শেষ অঙ্ক বা LSD। তারপর যেকোনো একটি লুপের সাহায্যে (for কিংবা while) সংখ্যাটি 10 দিয়ে মড (%) করে ভাগশেষ বের করব এবং সংখ্যাটিকে 10 দিয়ে ভাগ করে ভাগফল বের করব। সবশেষে ভাগফলটি আবার ওই ভ্যারিয়েবলে অ্যাসাইন করব। যখন, ভাগফল শূন্য হয়ে যাবে, তখন লুপ থেকে বের হয়ে যাব আর শেষ যেই ভাগশেষ পেয়েছিলাম, সেটিই হচ্ছে প্রথম অঙ্ক বা MSD। তবে যেহেতু এখানে নির্দিষ্ট করে ঠিক 5 টি অঙ্কের কথা বলে দেওয়া হয়েছে, তাই লুপ ব্যবহার না করেও এই সমস্যাটি সমাধান করা যাবে। সেটা কীভাবে তা তোমাদের একটু চিন্তা করে বের করতে হবে।

দুটি অঙ্ক বের করার পরের কাজটি খুবই সহজ। এবার সংখ্যা দুটি যোগ করে আউটপুট প্রিন্ট করে ফেলো।

সমস্যা ৭ – সংখ্যা গণনা

সমস্যার বিবরণ

একটি লাইনে অনেকগুলো সংখ্যা দেওয়া থাকবে। সংখ্যাগুলোর মধ্যে এক বা একাধিক স্পেস ক্যারেক্টার থাকবে। লাইনে মোট কয়টি সংখ্যা আছে সেটি বের করতে হবে।
(<http://bit.ly/1-52-problem-7>)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। তারপর T -এর মান যত, ততটি লাইন থাকবে। প্রতিটি লাইনে এক বা একাধিক সংখ্যা থাকবে যদের পরমমান 10000000-এর বেশি হবে না। একটি লাইনের সংখ্যাগুলোর মধ্যে এক বা একাধিক স্পেস ক্যারেক্টার থাকবে।

আউটপুট

প্রতিটি লাইনে কয়টি সংখ্যা আছে সেটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
4	7
1 -2 10000 -50 20 7 445	1
9	3
-98 876 65	3
223 9876452 212	

সমাধান

এই সমস্যাটি সমাধান করার জন্য আমরা `strtol()` নামে একটি ফাংশন ব্যবহার করব। এই ফাংশনকে বলা হয় `string to long` কনভার্টার, অর্থাৎ, এই ফাংশনকে কোনো সংখ্যা স্ট্রিং ডেটা টাইপে ইনপুট দিলে সে, `long` ডেটা টাইপে সেই সংখ্যাটি রিটার্ন করে। আর এই ফাংশনটি রয়েছে `stdlib.h` হেডার ফাইলে। চলো আমরা প্রথমে দেখে নিই এই ফাংশনটি কীভাবে ব্যবহার করা যায়।

```
#include<stdio.h>
#include<stdlib.h>

int main()
```

```

{
    char line[] = "1 -2 10000 -50 20 7 445";
    char *p, *e;
    long input;
    int count = 0;
    p = line;

    for (p = line; ; p = e)
    {
        input = strtol(p, &e, 10);
        if (p == e)
        {
            break;
        }
        count++;
    }
    printf("%d\n", count);

    return 0;
}

```

তোমরা প্রথমেই এই কোডটি নিজেদের কম্পিউটারে চালিয়ে দেখ এটা কেমন আউটপুট দেয়। দেখবে এটি ঠিক 7 সংখ্যাটি প্রিন্ট করে। লক্ষ করে দেখ আমাদের line নামক স্ট্রিংটিতেও কিন্তু ঠিক 7টি পূর্ণসংখ্যা রয়েছে। লক্ষ করো, আমরা stdlib.h হেডার ফাইলটি শুরুতেই ইনকুড করে নিয়েছি। পরে আমরা line নামে একটি ভ্যারিয়েবল ডিক্লেয়ার করেছি, যাতে একটি স্ট্রিং রয়েছে। এই স্ট্রিংটিতে মোট 7টি পূর্ণসংখ্যা রয়েছে। strtol() ফাংশনটি ব্যবহার করার জন্য আমাদের দুটি ক্যারেক্টার পয়েন্টার ব্যবহার করতে হবে। তাই আমরা p ও e নামে দুটি পয়েন্টার নিয়েছি। পরে line নামক স্ট্রিং থেকে একটি করে সংখ্যা পড়ার জন্য আমরা একটি for লুপ ব্যবহার করেছি। এই লুপটিতে লক্ষ করলে দেখবে লুপটি শেষ হওয়ার কোনো শর্ত এখানে দেওয়া হয়নি। বরং লুপের ভেতরে একটি শর্ত ব্যবহার করে break করা হয়েছে। লুপ যখন শুরু হবে p পয়েন্টারটি line এর শুরুতে পয়েন্ট করবে। যখন strtol() ফাংশনটি একবার ব্যবহার করা হবে, e পয়েন্টারটি ওই সংখ্যার ঠিক যেখানে শেষ হলো line এর ঠিক সেই ইনডেক্সে পয়েন্ট করবে এবং পরেরবার p এর মানও সেখান থেকে শুরু হবে। এভাবে প্রতিবার যখন লুপ চলবে একটি করে সংখ্যা line থেকে পড়া হবে এবং count এর মান এক এক করে বাঢ়তে থাকবে। এক পর্যায়ে যখন আমরা স্ট্রিংয়ের শেষে চলে আসব তখন e এর মান হয়ে যাবে null আর তার পরেরবার p এর মান ও হয়ে যাবে null। ঠিক তখনই আমাদের লুপটা break করবে। তোমরা যারা একেবারেই নতুন তাদের এই কোডটি সামান্য দুর্বোধ্য মনে হতে পারে। যাদের স্ট্রিং ও

পয়েন্টার বিষয়ে ধারণা নেই তাদেরকে আগে এই দুটি বিষয়ে ধারণা পরিষ্কার করে নিতে হবে।
আর এই `strtol()` ফাংশনের বিস্তারিত বর্ণনা পাবে এখানে:

<http://www.cplusplus.com/reference/cstdlib/strtol/>

তাহলে তো দেখে ফেললে একটি স্ট্রিং থেকে কীভাবে তাতে কয়টি সংখ্যা আছে বের করে ফেলা যায়। এখন আমাদের কাজ হচ্ছে এই স্ট্রিংটি ইনপুট নেওয়া। যেহেতু এখানে স্ট্রিংয়ের দৈর্ঘ্য কত হতে পারে তা বলা নেই তাই আমরা ধরে নেব এর দৈর্ঘ্য 100000।

```
char line[100000];
scanf("%[^\\n]", line);
```

খেয়াল করো আমরা কিন্তু %s ব্যবহার করে ইনপুট নেইনি। এভাবে কেন ইনপুট নেওয়া হলো সেটা পরে অন্য একটি সমস্যার সমাধানে ব্যাখ্যা করা হয়েছে। তাহলে এখন তোমরা পুরো কোডটা গুছিয়ে লিখে ফেলো আর এই লিঙ্কে গিয়ে সাবমিট করে দাও। যদি ফলাফলে রানটাইম এরর আসে, তাহলে বুঝতে হবে স্ট্রিংয়ের দৈর্ঘ্য আরও বেশি ও হতে পারে। তখন স্ট্রিংয়ের দৈর্ঘ্য বাড়িয়ে আবার সাবমিট করতে হবে।

`strtol()` ফাংশন ব্যবহার না করেও কিন্তু প্রোগ্রামটি লেখা যায়। তোমরা সেটি চেষ্টা করে দেখ।

সমস্যা ৮ – ছোট থেকে বড়

সমস্যার বিবরণ

তিনটি পৃথক সংখ্যা দেওয়া থাকবে। এদের ছোট থেকে বড় আকারে প্রিন্ট করতে হবে।
[\(<http://bit.ly/1-52-problem-8>\)](http://bit.ly/1-52-problem-8)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$)। এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে তিনটি করে পূর্ণসংখ্যা n_1, n_2, n_3 থাকবে যারা প্রত্যেকে স্বতন্ত্র (অর্থাৎ, কোনো দুটি সংখ্যা পরস্পর সমান নয়) এবং 1000 এর সমান বা ছোট।

আউটপুট

প্রতিটি কেসের জন্য একটি করে কেস নম্বর প্রিন্ট করতে হবে। এরপরে প্রদত্ত তিনটি সংখ্যাকে ছোট থেকে বড় আকারে সাজিয়ে প্রিন্ট করতে হবে। পাশাপাশি দুটি সংখ্যার মধ্যে শুধুমাত্র একটি স্পেস প্রিন্ট করতে হবে। নমুনা আউটপুটে আরও বিস্তারিত দেখতে পারো।

উদাহরণ

ইনপুট	আউটপুট
3	Case 1: 1 2 3
3 2 1	Case 2: 1 2 3
1 2 3	Case 3: 5 6 10
10 5 6	

সমাধান

তোমরা যারা ইতিমধ্যেই বিভিন্ন প্রকার সর্টিংয়ের সঙ্গে পরিচিত, যেমন বাবল সর্ট, সিলেকশন সর্ট ইত্যাদি, তারা চাইলেই যেকোনো সর্টিং অ্যালগরিদম ব্যবহার করে সমস্যাটির সমাধান করে ফেলতে পার। সর্টিং মানে হচ্ছে ক্রমানুসারে সাজানো (ছোট থেকে বড় কিংবা বড় থেকে ছোট)। আর এই সমস্যার সমাধানে সর্টিং করার জন্য অবশ্যই তোমাকে অ্যারে ব্যবহার করতে হবে। কিন্তু আমি চাই তোমরা কোনো অ্যারে ব্যবহার না করে সমস্যাটির সমাধান করো। কীভাবে? সবচেয়ে ভালো হয় নিজে কিছু সময় চিন্তা করে সমাধান বের করতে পারলে। চেষ্টা করে না পারলে (কিংবা পারলেও) নিচের অংশ পড়া শুরু করো।

এই সমস্যায় তোমাদের যদি তিনটি সংখ্যার পরিবর্তে দুটি সংখ্যা দেওয়া হতো এবং সেগুলো ছেট-বড় ক্রমানুসারে সাজাতে বলা হতো, তাহলে তোমরা কী করতে? আমি পুরো কোডটি লিখে দিচ্ছি, তোমরা সহজেই বুঝতে পারবে।

```
int main()
{
    -
    int T, n1, n2, temp, kase;
    scanf("%d", &T);
    for(kase = 1; kase <= T; kase++)
    {
        scanf("%d %d", &n1, &n2);
        if (n1 > n2)
        {
            temp = n1;
            n1 = n2;
            n2 = temp;
        }

        printf("Case %d: %d %d\n", kase, n1, n2);
    }

    return 0;
}
```

লজিক খুবই সহজ, প্রথম সংখ্যাটি যদি দ্বিতীয় সংখ্যার চেয়ে বড় হয়, তাহলে সংখ্যা দুটি অদল-বদল করে ফেললেই হয়! তাহলে তিনটি সংখ্যা থাকলে কী করতে হবে, সেটি নিচয়ই বুঝতে পারছ। মোট তিনবার অদল-বদল করতে হবে। সেটি তোমরা নিজেরা করে ফেলো।

আরেকটি ব্যাপার লক্ষ কর। আমি ভ্যারিয়েবল ডিক্রেয়ার করার সময় kase নাম দিয়েছি, case দিইনি। কারণ কী? কারণ case হচ্ছে সি ভাষায় সংরক্ষিত একটি শব্দ (reserved keyword)। তাই এই নামে কোনো ভ্যারিয়েবলের নামকরণ করা যাবে না, ফাংশনের নামেও এসব শব্দ ব্যবহার করা যাবে না। সি ভাষায় অন্যান্য সংরক্ষিত শব্দগুলো হচ্ছে:

auto	else	long	switch
break	enum	register	typedef
case	extern	return	union
char	float	short	unsigned
const	for	signed	void
continue	goto	sizeof	volatile
default	if	static	while
do	int	struct	_Packed
double			

সমস্যা ৯ – পূর্ণবর্গ সংখ্যা

সমস্যার বিবরণ

একটি সংখ্যা পূর্ণবর্গ কি না সেটি বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-9>)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$). এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি পূর্ণসংখ্যা N ($0 \leq N \leq 2^{31}$)।

আউটপুট

প্রোগ্রামটির আউটপুটে N পূর্ণবর্গ সংখ্যা হলে YES প্রিন্ট করতে হবে। অন্যথায় প্রিন্ট করতে হবে NO।

উদাহরণ

ইনপুট	আউটপুট
3	YES
16	NO
18	YES
196	

সমাধান

পূর্ণবর্গ সংখ্যা হচ্ছে সেসব সংখ্যা, যেসব সংখ্যার বর্গমূল একটি পূর্ণসংখ্যা। তাহলে এই সমস্যার সমাধান করতে গিয়ে আমাদের প্রথমে যে কাজটি করতে হবে, তা হলো বর্গমূল বের করা। তারপর সেই বর্গমূলটি পূর্ণ সংখ্যা কি না, তা যাচাই করা। বর্গমূল নির্ণয় করার জন্য math.h হেডার ফাইলে sqrt() ফাংশন রয়েছে। আর পূর্ণসংখ্যা কি না, তা যাচাই করার বেশ কয়েকটি উপায় রয়েছে।

```
double sq_root = sqrt(num);

if (ceil(sq_root) == floor(sq_root))
{
    printf("YES");
}
else
```

```

{
    printf("NO");
}

```

এই সমাধানটিতে আমরা `ceil()` ও `floor()` নামে দুটি ফাংশন ব্যবহার করেছি। `ceil()` আর `floor()` ফাংশনের কাজ কী? `ceil()` ফাংশন একটি বাস্তব সংখ্যা ইনপুট নেয় এবং সেই সংখ্যার চেয়ে বড় সর্বনিম্ন পূর্ণসংখ্যা রিটার্ন করে। যেমন `ceil(2.1)` লিখলে, সেটি রিটার্ন করবে `3.0`, `ceil(2.8)` লিখলেও সেটি `3.0` রিটার্ন করবে। `ceil` হচ্ছে `ceiling` শব্দের সংক্ষিপ্ত রূপ যার অর্থ হচ্ছে ছাদ। আমরা সিলিং ফ্যান দিয়ে যেমন ঘরের ছাদের ফ্যান বা পাখা বোঝাই এটি-ই সেই সিলিং। আর `floor` হচ্ছে ঘরের মেঝে। `floor()` ফাংশন একটি বাস্তব সংখ্যা ইনপুট নেয় এবং সেই সংখ্যার চেয়ে ছোট সর্বোচ্চ পূর্ণসংখ্যা রিটার্ন করে। `floor(2.1)` রিটার্ন করবে `2.0`। একইভাবে `floor(2.8)` রিটার্ন করবে `2.0`। আমরা এখানে যে যুক্তি ব্যবহার করলাম তা হচ্ছে, বর্গমূল `sq_root` যদি পূর্ণসংখ্যা হয়, তবে তার `ceil()` এবং `floor()` এর মান সমান হবে। আশা করি, বিষয়টি তোমরা বুঝতে পেরেছ।

তবে এই সমাধানে ছোট একটি সমস্যা রয়েছে। `ceil()` ও `floor()` দুটি ফাংশনেরই রিটার্ন টাইপ হচ্ছে `double` বা `float`। দুটি `double` বা `float` টাইপের ভ্যারিয়েবল == দিয়ে তুলনা করলে অনেক সময় ভুল হয়। এটি প্রোগ্রামারের দোষ নয়, কম্পিউটারের প্রসেসরের দোষ। তো তোমার কী করার আছে? তুমি কোড এভাবে লিখতে পারো:

```

double sq_root = sqrt(num);
double difference = ceil(sq_root) - floor(sqrt);

if (difference < 0.000001)
{
    printf("YES\n");
}
else
{
    printf("NO\n");
}

```

তবে আমি এই সমস্যা সমাধানের জন্য তৃতীয় একটি পদ্ধতি বেছে নেব। সেটিও কোড দেখলেই তোমরা বুঝতে পারবে। শুধু বলে দিই যেকোনো `double` ভ্যারিয়েবলকে `int` ভ্যারিয়েবলে অ্যাসাইন করলে আপনাআপনি টাইপকাস্ট হয়ে যায়। টাইপকাস্ট কী জিনিস সেটি না জানলে কোনো বই পড়ে বা বস্তুকে জিজ্ঞাসা করে যেনে নাও। অথবা প্রশ্ন করতে পারো <http://programabad.com> -এ।

```
int num, sq_root;  
  
scanf("%d", &num);  
sq_root = sqrt(num);  
  
if (sq_root * sq_root == num)  
{  
    printf("YES\n");  
}  
else  
{  
    printf("NO\n");  
}
```

সমস্যার বিবরণ

তুমি গিয়েছ ক্রিকেট খেলা দেখতে। হঠাৎ খেয়াল করলে ইলেকট্রনিক স্কোরবোর্ড আর সব ঠিক দেখালেও রানরেট ভুল দেখাচ্ছে। ব্যাপারটা ওদের বলতে গিয়ে জানলে যে ওরাও ভুলটা খেয়াল করেছে। কিন্তু অস্ট্রেলিয়া থেকে আমদানি করা স্কোরবোর্ডের প্রোগ্রাম ঠিক করতে সেই দেশের ইঞ্জিনিয়ার আনতে হবে, তার জন্য মন্ত্রণালয় থেকে অনুমতি, অনুদান, সই, সিলচাপ্পির, এটাসেটা অনেক কিছু লাগবে বলে কাজটা হয়ে উঠছে না। কিন্তু তুমি বুবাতে পেরেছ ব্যাপারটা স্রেফ একটা প্রোগ্রামিং বাগ। দু লাইনের একটা কোড লিখেই ঠিক করে ফেলতে পারবে। সেটা বলতেই, কিছুটা ভেবে টেকনিশিয়ান লোকটা রাজি হয়ে গেল। শর্ত একটাই, তার বসকে কিছু বলা যাবে না। তুমি ও তোমার প্রোগ্রামিং ক্ষিল সত্যিকার একটা কাজে লাগানোর সুযোগ পেয়ে তুমুল উৎসাহে কোড করতে শুরু করে দিলে।

খেলাটা হচ্ছে 50 ওভারের ওয়ানডে (ODI) ম্যাচ। প্রতিবার যখন ডিস্প্লেতে ওভারপ্রতি বর্তমান রানের হার (current run rate) এবং জেতার জন্য কার্জিক্ষিত রানের হার (required run rate) দেখানো হয় তখন প্রতিপক্ষের করা মোট রান, ব্যাটসম্যানদের বর্তমান রান এবং খেলার আর কত বল বাকি আছে তা জানা থাকে।

উল্লেখ্য, ক্রিকেটে 6 বলে 1 ওভার হয় এবং জিততে হলে প্রতিপক্ষের মোট রানের চেয়ে অন্তত 1 রান বেশি করতে হয়।

(<http://bit.ly/1-52-problem-10>)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। ওই সংখ্যার মান যত, এর পরে ততগুলো লাইনে তিনটি করে সংখ্যা থাকবে। প্রথম সংখ্যাটি প্রতিপক্ষের মোট রান, r_1 ($1 \leq r_1 \leq 1000$), দ্বিতীয় সংখ্যাটি ব্যাটসম্যানদের বর্তমান রান, r_2 ($1 \leq r_2 \leq r_1$) এবং তৃতীয় সংখ্যাটি খেলার আর কত বল বাকি আছে, B ($1 \leq B \leq 300$) তা নির্দেশ করে।

আউটপুট

প্রতি লাইনের জন্য সেই লাইনের দেওয়া তথ্য থেকে হিসাব করে প্রথমে ওভারপ্রতি বর্তমান রানের হার এবং এরপর একটি স্পেস দিয়ে কার্জিক্ষিত রানের হার প্রিন্ট করতে হবে। প্রতিটি হার অবশ্যই দশমিকের পরে শুধুমাত্র দুই অঙ্ক পর্যন্ত দেখাতে হবে।

উদাহরণ

ইনপুট	আউটপুট
4	6.00 7.00
300 294 6	3.00 6.06
200 100 100	5.77 12.60
333 250 40	5.00 0.63
118 100 180	

সমাধান

তোমরা নিচয়েই ক্রিকেট খেলা বোঝ, অন্তত আমার চাইতে তো বেশি অবশ্যই। তো এই সমস্যাটি সমাধান করার সময় ভুলে গেলে চলবে না যে প্রতিপক্ষের রান যদি 300 হয়, তাহলে দ্বিতীয় ইনিংসে যারা ব্যাট করছে, তাদের জয়ের জন্য 301 রান করতে হবে। তাহলে কথা বাড়িয়ে লাভ নেই, সরাসরি কোডে চলে যাই (হ্যাঁ, পুরো কোডটি লিখে দিচ্ছি):

```
#include<stdio.h>

int main()
{
    int T, i, r1, r2, B, ball_played;
    double current_rr, asking_rr; // Run Rates

    scanf("%d", &T);

    while(T--)
    {
        scanf("%d %d %d", &r1, &r2, &B);
        ball_played = 300 - B;
        current_rr = (r2 / ball_played) * 6;
        asking_rr = ((r1 - r2 + 1) / B) * 6;
        printf("%0.2lf %0.2lf\n", current_rr, asking_rr);
    }

    return 0;
}
```

তোমরা কোডটি একবার পড়লেই বুঝতে পারবে। অনেক সহজ। কিন্তু প্রোগ্রাম যখন রান করবে, দেখবে যে উদাহরণে যে ইনপুট আছে, সেগুলোর জন্য তোমার আউটপুট উদাহরণের আউটপুটের সঙ্গে মিলছে না!

কারণ হচ্ছে কম্পিউটার প্রোগ্রামে আমরা যখন পূর্ণসংখ্যাকে পূর্ণসংখ্যা দিয়ে ভাগ করি, ভাগফলও আসে পূর্ণসংখ্যা। তাই আমরা ভাগফলটি কোনো double টাইপের ভ্যারিয়েবলে রাখলেও সেখানে ওই পূর্ণসংখ্যাই আসে। একটি ছোট প্রোগ্রাম লিখে দেখা যাক:

```
int main()
{
    int num1 = 5, num2 = 2;
    double ans;
    ans = num1 / num2;
    printf("ans = %0.2lf\n", ans);
    return 0;
}
```

আউটপুট কী পাচ্ছ? **ans = 2.00**। তাই আমাদের যেটি করতে হবে, ভাগ করার আগেই ইন্টিজারকে double টাইপে নিয়ে যেতে হবে। সহজ বুদ্ধি হচ্ছে ইন্টিজারকে 1.0 দিয়ে গুণ করে ফেলা:

```
ans = num1 * 1.0 / num2;
```

তাহলে আউটপুট আসবে **ans = 2.50**, যা আমাদের প্রত্যাশিত আউটপুট। তাহলে তোমরা প্রোগ্রামটি ঠিকঠাক করে লিখে ফেলো।

সমস্যা ১১ – গৌণিক বা ফ্যাক্টরিয়াল

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি সংখ্যার গৌণিক বা ফ্যাক্টরিয়াল বের করতে পারে। (<http://bit.ly/1-52-problem-11>)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$). এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি পূর্ণসংখ্যা N ($0 \leq N \leq 15$)।

আউটপুট

প্রোগ্রামটির আউটপুটে গৌণিক বা ফ্যাক্টরিয়াল প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	720
6	3628800
10	1307674368000
15	

সমাধান

সমস্যাটির বর্ণনায় বলা আছে প্রথমে একটি সংখ্যা T থাকবে এবং তারপর T সংখ্যক লাইনে একটি পূর্ণসংখ্যা N ইনপুট নিতে হবে। তাহলে প্রথমেই আমাদের T এর মান ইনপুট নিতে হবে, এখানে আমরা T-এর ডেটা টাইপ হিসেবে int ব্যবহার করব। T সংখ্যক লাইনে ইনপুট নেওয়ার জন্য আমরা একটি for লুপ ব্যবহার করব। যেহেতু T সংখ্যক লাইনে N এর মান ইনপুট নিতে হবে, এখানে N এর ডেটা রেঞ্জ দেওয়া আছে সর্বোচ্চ 15। তাই N এর জন্য আমরা int ডেটা টাইপ ব্যবহার করব।

```
int T, N;
scanf("%d", &T);
for(int i = 0; i <= T; i++) {}
```

এবারে আমরা for লুপের ভেতরে N এর মান ইনপুট নিব।

```
scanf("%d", &N);
```

এবারে আমাদের ফ্যাক্টরিয়াল বা গৌণিকের কাজের জন্য কোড লেখার পালা। ফ্যাক্টরিয়ালের কোড আমরা রিকার্শন (recursion) ব্যবহার করে করতে পারি, কিন্তু সেক্ষেত্রে আমাদের রানটাইম লেগে যাবে অনেক বেশি। তাই আমরা এখানে ফ্যাক্টরিয়ালের কাজটি লুপের সাহায্যে করব।

ফ্যাক্টরিয়ালের ধারণাটি আসলে কী? আমরা জানি, কোনো সংখ্যার ফ্যাক্টরিয়াল সেই সংখ্যাটিসহ তার আগের সংখ্যাগুলোর গুণফল। যেমন: 5 এর ফ্যাক্টরিয়াল $120 (1 \times 2 \times 3 \times 4 \times 5 = 120)$ । এই কাজটি করার জন্য আমাদের 1 থেকে N পর্যন্ত সংখ্যার গুণফল বের করতে হবে। N-এর ফ্যাক্টরিয়ালের মান গণনা করার জন্য আমরা আরেকটি ভ্যারিয়েবল factorial ব্যবহার করবো। লক্ষ করে দেখ, 15 এর ফ্যাক্টরিয়াল বা গৌণিক বেশ বড় একটি সংখ্যা। তাই এখানে factorial এর জন্য আমরা long long int ডেটা টাইপ ব্যবহার করব। প্রথমে আমরা এই ভ্যারিয়েবলের মান রাখব 1 অর্থাৎ **factorial = 1;** এবং N পর্যন্ত একটি for লুপ দিয়ে N এর ফ্যাক্টরিয়াল এর মান বের করব। ফ্যাক্টরিয়ালের জন্য আমাদের কোডটি হবে এমন:

```
long long int factorial = 1, i;
for(i = 2; i <= N; i++)
{
    factorial = factorial * i;
}
```

এখানে, **factorial = factorial*i;** এই statement এ প্রতিবার factorial এর মানের সঙ্গে i এর মান গুণ হচ্ছে N পর্যন্ত। যা আমাদের কাঞ্চিত আউটপুট। আরও একটা ব্যাপার লক্ষ করো, আমরা i এর মান 1 দিয়ে শুরু না করে 2 দিয়ে শুরু করেছি, কেননা $1 \times 1 = 1$ তাই, 1 দিয়ে আলাদাভাবে গুণ করার দরকার নেই। আমাদের আউটপুট statement-টি হবে এমন:

```
printf("%lld\n", fact);
```

প্রতিবার আউটপুট প্রিন্ট শেষে একটি নিউলাইন অবশ্যই প্রিন্ট করতে হবে। এবার তোমাদের কাজ হচ্ছে গুচ্ছিয়ে সুন্দরভাবে কোডটি বাটপট লিখে ফেলা।

সমস্যা ১২ – ফ্যাক্টরিয়াল ১০০

সমস্যার বিবরণ

একটি সংখ্যার ফ্যাক্টরিয়ালের শেষে কতটি শূন্য (0) আছে, তা বের করতে হবে।
[\(<http://bit.ly/1-52-problem-12>\)](http://bit.ly/1-52-problem-12)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$). এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি পূর্ণসংখ্যা N ($0 \leq N \leq 100$)।

আউটপুট

N ফ্যাক্টরিয়ালের শেষে কতটি শূন্য (0) আছে, তা এক লাইনে প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	1
6	3
15	24
100	

সমাধান

N-এর ফ্যাক্টরিয়াল যেই সংখ্যাটি, তার শেষে মোট কয়টি 0 আছে, সেটি আমাদের বের করতে হবে। শুরুতেই আমরা দেখি, একটি সংখ্যার শেষে কয়টি 0 আছে, সেটি কীভাবে বের করা যায়।

কোনো সংখ্যাকে 10 দিয়ে ভাগ করলে ভাগশেষ হতে পারে 0 থেকে 9 পর্যন্ত যেকোনো সংখ্যা। 10 দিয়ে ভাগ করলে ভাগশেষ 0 তখনই হবে, যখন সংখ্যাটির সবচেয়ে ডানদিকের অঙ্ক (এককের ঘরের অঙ্ক বা LSD) 0 হবে। একটি উদাহরণ দেওয়া যাক। 3100-এর শেষে কয়টি 0 আছে? প্রথমে সংখ্যাটিকে 10 দিয়ে মড করি (মানে ভাগশেষ বের করি)।

3100 % 10 এর মান হবে 0। তার মানে সবচেয়ে ডানদিকের 0 পেয়ে গেলাম। তারপর 3100-কে 10 দিয়ে ভাগ করে দিই ($3100 / 10$), ভাগফল হবে 310। এখন 310-কে আবার 10 দিয়ে মড করি, ভাগশেষ 0। তাহলে এখন পর্যন্ত ডানদিকে দুটি 0 পেলাম। তারপর 310-কে 10 দিয়ে ভাগ করি, ভাগফল 31। 31-কে আবার 10 দিয়ে মড করি, ফলাফল 1। এখানেই আমাদের খেমে যেতে হবে। আমরা যতক্ষণ ভাগশেষ 0 পাচ্ছিলাম ততক্ষণ একই কাজের পুনরাবৃত্তি করছিলাম।

$3100 \% 10 = 0$	$3100 / 10 = 310$
$310 \% 10 = 0$	$310 / 10 = 31$
$31 \% 10 = 1$	stop

প্রোগ্রামিংয়ে কাজটি লুপ দিয়ে করতে হবে, একথা নিচয়ই তোমরা সবাই বুঝে ফেলেছ।

```
int mod, zero_count = 0;

while(number > 0)
{
    mod = number % 10;
    if (mod == 0) zero_count++;
    else break;
    number = number / 10;
}
```

এখন তোমরা ভাবছ, সমস্যার সমাধান তো হয়েই গেলে। N-এর ফ্যাক্টরিয়াল বের করে শেষে কয়টি 0 আছে, তা ওপরের কোড ব্যবহার করেই বের করে ফেলা যাবে। কিন্তু ঘাপলা হচ্ছে N-এর মান 100! পর্যন্ত হতে পারে, আর 100! সংখ্যাটি এতে বড় সংখ্যা যে int, এমনকি long long int ব্যবহার করেও সংখ্যাটি রাখা যাবে না। তাহলে কি স্ট্রিং ব্যবহার করব?

স্ট্রিং ব্যবহার করে চেষ্টা করতে পারো, কিন্তু এত কষ্ট না করলেও চলবে। আসলে $100!$ -এর শেষে কয়টি 0 আছে, সেটি বের করার জন্য $100!$ -এর পুরো মান আমাদের বের করার দরকার নেই। একটি সংখ্যার শেষে একটি 0 কখন আসে? যখন সেটির একটি উৎপাদক 10 হয়। যেমন : $500 = 5 \times 10 = 5 \times 10 \times 10$ । তার মানে দুটি 10 আছে, তাই সংখ্যাটির শেষেও দুটি 0।

আবার 10 কীভাবে তৈরি হয়? $10 = 5 \times 2$ । 5 ও 2, উভয়ই মৌলিক সংখ্যা, তাই এদেরকে আর উৎপাদকে ভাঙ্গা যাবে না।

এখন,

$$\begin{aligned} 10! &= 10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 \\ &= (5 \times 2) \times (3 \times 3) \times (2 \times 2 \times 2) \times 7 \times (3 \times 2) \times 5 \times (2 \times 2) \times 3 \times 2 \times 1 \\ &= 7 \times (5 \times 5) \times (3 \times 3 \times 3 \times 3) \times (2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2) \times 1 \end{aligned}$$

এখানে 5 আছে দুটি, আর 2 তো এর চেয়ে অনেক বেশি আছে (7টি)। আর 5-এর সঙ্গে 2 গুণ করলেই 10 হয়, মানে একটি শূন্য তৈরি হয়। তাই যেহেতু দুটি 5 রয়েছে (আর সেগুলোর জন্য যথেষ্ট 2 আছে), তাই সবার শেষে শূন্যও থাকবে 2টি। তোমরা $10!$ এর মান বের করলে দেখবে

এর মান হচ্ছে 3628800। ডানদিকে দৃটি 0। আমরা কিন্তু এই মান বের না করেই সেটি বলে দিতে পেরেছি।

আর কোড করে দেব না। তোমরা এখন নিজেরা চেষ্টা করে কোড লিখে ফেলো।

সমস্যা ১৩ – টমি মিয়ার প্রোবাবিলিটি

সমস্যার বিবরণ

টমি মিয়া প্রোবাবিলিটিতে (সন্তান্তিতা) বিশ্বাস করে। ওকে কেউ যখন জিজ্ঞেস করে, "তুমি ডেমোক্রেসিতে বিশ্বাস করো? তুমি সিনেমার গল্পে বিশ্বাস করো? তুমি ... ", টমি মিয়া চোখ বন্ধ করে দুপাশে গভীরভাবে মাথা নেড়ে বলে, 'উহ, আমি শুধু প্রোবাবিলিটিতে বিশ্বাস করি। বেশিরভাগ মানুষ প্রোবাবিলিটি বোঝে না। সেজন্য ওরা আর কথা বাঢ়ায় না।

তো কেউই খুব অবাক হলো না যখন টমি মিয়া তার বড় ছেলেকে ইংলিশ মিডিয়াম স্কুলে ভর্তি করানোর জন্য প্রস্তুতি নেওয়া শুরু করল। এখন সমস্যা হচ্ছে, ভর্তি পরীক্ষায় ইংরেজি অনুবাদ করতে হবে। টমি মিয়ার ছেলে শব্দগুলোর ইংরেজি অনুবাদ জানে, কিন্তু সে ব্যাকরণ জানে না। এখন সমস্যা হচ্ছে ব্যাকরণ না মানলে ঠিক বাক্য গঠন হয় না। যেমন ধরো, 'তুমি ভাত খাও' এটা তুমি যদি ইংরেজিতে অনুবাদ করে লিখো, 'rice eat you' তাহলে কেউ ভাববে না তুমি কবি। সবাই ভাববে তুমি ব্যাকরণ জানো না, কিংবা ভাববে তুমি চাচ্ছো ভাত তোমাকে খেয়ে ফেলুক!

টমি মিয়া প্রোবাবিলিটিতে বিশ্বাস করে। তো প্রতিটি বাক্যের জন্য টমি মিয়া জানতে চায় তার বড় ছেলের সঠিক ইওয়ার প্রোবাবিলিটি কত।

(<http://bit.ly/1-52-problem-13>)

ইনপুট

প্রথম লাইনে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) থাকবে। পরবর্তীতে T সংখ্যক লাইন থাকবে এবং প্রতি লাইনে একটি ইংরেজি বাক্য থাকবে – সেখানে 10টির বেশি শব্দ থাকবে না। প্রতি শব্দে 20টির বেশি বর্ণ (Letter) থাকবে না।

আউটপুট

প্রতি লাইনে অনুবাদ সঠিক হওয়ার প্রোবাবিলিটি প্রিন্ট করতে হবে। তুমি ধরে নিতে পারো, প্রতিটি বাকেয়ের জন্য শুধু একটি মাত্র সঠিক অনুক্রম আছে। অনুক্রম মানে হচ্ছে order। আউটপুট দেখাতে হবে $1/n$ আকারে যেখানে n একটি পূর্ণসংখ্যা।

উদাহরণ

ইনপুট	আউটপুট
2	1/6
eat you rice	1/12
no way no good	

সমাধান

সমস্যাটি পড়ে ঘাবড়ানোর কিছু নেই। তোমাদের যদি বিন্যাস-সমাবেশ জানা থাকে; তাহলে সমাধান খুবই সহজ। না জানা থাকলে কলেজ পড়া যা কারও কাছে জেনে নাও, বা নিজেও উচ্চ মাধ্যমিক গণিত বই সংগ্রহ করে পড়ে নিতে পারো।

n -সংখ্যক জিনিসকে সাজানো যায় $n!$ উপায়ে। আবার জিনিসগুলোর মধ্যে যদি n_1 সংখ্যক জিনিস একই রকম হয়, আর n_2 সংখ্যক জিনিস একই রকম হয়, তখন একটু ঝামেলা বেঁধে যায়। ধরা যাক বিভিন্ন রঙয়ের 10টি বল আছে, এখন 3টি বল যদি একই রঙয়ের হয়, ধরা যাক নীল, তাহলে আমরা কত উপায়ে বলগুলো সাজাতে পারি? আবার যদি 3টি নীল বল, 2টি সাদা বল, 4টি লাল বল আর 1টি সবুজ বল হয়, তাহলে বলগুলো কত উপায়ে সাজানো যায়?

তোমরা বই থেকে সূত্র দেখে প্রোগ্রাম লিখে ফেলো না যেন। ভালোমতো পড়াশোনা ও একটু চিন্তাবন্দন করো।

আরেকটি বিষয় হচ্ছে হিসাব-নিকাশে যাওয়ার আগে, কোন শব্দ কতবার আছে, সেটি গণনা করতে হবে। এটি মোটেও কঠিন কোনো কাজ নয়, যদি তোমরা "শব্দ গণনা – ২" সমস্যাটি সমাধান করে থাকো।

সমস্যা ১৪ – অক্ষরের ঘনষ্ঠা

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি বাক্যে কোনো একটি বিশেষ বর্ণ কতবার আছে সেটি প্রিন্ট করবে।

<http://bit.ly/1-52-problem-14>

ইনপুট

প্রথম লাইনে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) থাকবে। পরবর্তীতে $2 \times T$ সংখ্যক লাইন থাকবে। পরবর্তী প্রতি একজোড়া লাইনের প্রথমটিতে একটি স্ট্রিং (string) থাকবে যার দৈর্ঘ্য সর্বোচ্চ 10000 এবং দ্বিতীয় লাইনে একটি ক্যারেক্টর (character) থাকবে।

আউটপুট

প্রোগ্রামটির আউটপুটে ক্যারেক্টারটি স্ট্রিংয়ে কতবার আছে সেটি প্রিন্ট করতে হবে নিম্নরূপে।

উদাহরণ

ইনপুট	আউটপুট
2 hello world l hello world a	Occurrence of 'l' in 'hello world' = 3 'a' is not present

সমাধান

এই সমস্যার বর্ণনায় বলা আছে যে প্রতিটি কেসে ইনপুট দেওয়া হবে দুটি লাইন। প্রথম লাইনে একটি স্ট্রিং ও দ্বিতীয় লাইনে একটি ক্যারেক্টর। তাই প্রথম লাইনটি ইনপুট নেওয়ার জন্য আমরা একটি স্ট্রিং অর্থাৎ ক্যারেক্টর অ্যারে ব্যবহার করব। আর দ্বিতীয় লাইন ইনপুট নেওয়ার জন্য একটি ক্যারেক্টর ব্যবহার করা যায়, আবার একটি স্ট্রিং ও ব্যবহার করা যায়।

```
char first_line[10001];
char second_line[2];
```

দ্বিতীয় লাইনে আমি স্ট্রিং ব্যবহার করছি এবং এর দৈর্ঘ্য 2 বলে দিচ্ছি। কারণ একটি ক্যারেক্টার ইনপুট হলে সেটির জন্য একবার ব্যবহার করা হবে এবং স্ট্রিংয়ের শেষে নাল ক্যারেক্টারের জন্য এক ঘর সংরক্ষিত রাখতে হবে।

কী ধরনের ডেটা স্ট্রাকচার ব্যবহার করে ইনপুট নিতে হবে সেটি তো আমরা ঠিক করলাম, এবাবে ইনপুট নেওয়ার পার্শ্ব। `scanf("%s", first_line);` - এভাবে ইনপুট নিলে হবে না, কারণ প্রথম লাইনে স্পেস ক্যারেক্টারও থাকতে পারে। তাই প্রথম লাইন ইনপুট নিতে হবে এভাবে

```
scanf(" %[^\n]", first_line);
```

আর দ্বিতীয় লাইন ইনপুট নেওয়া যায় এভাবে: `scanf("%s", second_line);`

আমরা যেই অক্ষরটি খুঁজব, সেটি আছে `second_line[0]`-এ। একটি লুপ ব্যবহার করে প্রথম লাইনের সব অক্ষরের সঙ্গে মিলিয়ে দেখতে হবে যে আমরা যেই অক্ষরটি খুঁজছি, এটি সেই অক্ষর কি না, যদি তা হয়, তাহলে আমরা যেই ভেরিয়েবল ব্যবহার করে গণনা করব, তার মান 1 বাড়িয়ে দিতে হবে।

```
count = 0;
for(i = 0; i < strlen(first_line); i++)
{
    if (second_line[0] == first_line[i])
    {
        count++;
    }
}
```

এখন হচ্ছে আউটপুট প্রিন্ট করার পালা। `printf()` ফাংশন ব্যবহার করে আউটপুট দেওয়ার কাজটি করা যাবে:

```
printf("Occurrence of '%c' in '%s' = %d\n", second_line[0],
       first_line, count);
```

লক্ষ করো যে আমি ফাংশনের ভেতরে একটি নিউলাইন প্রিন্ট করেছি। আর তার আগে পরীক্ষা করতে হবে যে অক্ষরটি যদি একেবারে না থাকে, তাহলে প্রিন্ট করতে হবে এ রকম করে:

```
printf('%c' is not present\n", second_line[0]);
```

এখন তোমরা কোডটি শুনিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো। তারপর এই সমস্যার লিঙ্কে গিয়ে সাবমিট করে দেখো।

সমস্যা ১৫ – অক্ষর গণনা

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে বাক্যে সবগুলো বর্ণ (alphabet) কতবার করে এসেছে সেটি প্রিন্ট করবে।

(<http://bit.ly/1-52-problem-15>)

ইনপুট

প্রথম লাইনে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) থাকবে। পরবর্তীতে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি স্ট্রিং S (S এর দৈর্ঘ্য 1000 এর বেশি নয়)। স্ট্রিংের সবগুলো বর্ণই ছোটো হাতের অক্ষরে লেখা থাকবে।

আউটপুট

প্রোগ্রামটির আউটপুটে স্ট্রিং S এ কোন বর্ণটি কতবার আছে সেটি বর্ণক্রমানুসারে প্রিন্ট করতে হবে। আউটপুট "==" চিহ্নের আগে ও পরের স্পেসটি লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
3	e = 1
hello	h = 1
baby	l = 2
programming	o = 1
	a = 1
	b = 2

	y = 1
	a = 1
	g = 2
	i = 1
	m = 2
	n = 1
	o = 1
	p = 1
	r = 2

সমাধান

আগের সমস্যাটির সঙ্গে এই সমস্যায় পার্থক্য কোথায়? আগের সমস্যাতে শুধু একটি বিশেষ বর্ণ একটি স্ট্রিংয়ের মধ্যে কতবার রয়েছে তা গণনা করতে বলা হয়েছিল। আমরা count নামে একটি ভেরিয়েবল ব্যবহার করে সেই কাজটি করেছি। আর এই সমস্যায় সবগুলো বর্ণই কতবার রয়েছে তা গণনা করতে বলা হয়েছে। তাহলে এক্ষেত্রে আমাদের 26টি ভেরিয়েবল ব্যবহার করতে হবে। তা না করে আমরা এখানে 26টি পূর্ণসংখ্যার একটি অ্যারে ব্যবহার করতে পারি।

```
int count[26];
```

এখন আমাদের কাজ হচ্ছে এই বর্ণগুলো কতবার আছে তা গণনা করা। একটা বিষয় খেয়াল করো আমরা 'a' কতবার আছে তা রাখব count[0]-তে, 'b' কতবার আছে তা রাখব count[1]-এ, এভাবে করে 'z' কতবার আছে তা রাখব count[25]-এ। এর জন্য আমরা ASCII কোডের ধারণা ব্যবহার করব।

```
for(i = 0; i < strlen(s); i++)
{
    if( s[i] >= 'a' && s[i] <= 'z' )
    {
        count[s[i]-'a']++;
    }
}
```

মাত্র চার লাইনের কোডেই আমরা কাজটি করে ফেলেছি। তোমরা একটু চিন্তা করে দেখো দেখি বিষয়টি বুবাতে পারো কি না। আমরা প্রথমে শর্ত দিয়ে পরীক্ষা করে দেখেছি বর্ণটি এর মান 'a' থেকে 'z' এর মধ্যে কি না। তারপরে আমরা সেই বর্ণের মান থেকে 'a' এর মান বিয়োগ করে যে সংখ্যাটা পাওয়া যায়, count অ্যারের সেই ইনডেক্সের মান এক করে বাড়িয়ে দিয়েছি।

আসকি চার্টটি লক্ষ করো:

'a' = 97	'h' = 104	'o' = 111	'v' = 118
'b' = 98	'i' = 105	'p' = 112	'w' = 119
'c' = 99	'j' = 106	'q' = 113	'x' = 120
'd' = 100	'k' = 107	'r' = 114	'y' = 121
'e' = 101	'l' = 108	's' = 115	'z' = 122
'f' = 102	'm' = 109	't' = 116	
'g' = 103	'n' = 110	'u' = 117	

কম্পিউটার মেমোরিতে ঠিক এভাবেই সাংখ্যিক মানে বর্ণগুলোকে রাখা হয়। এই সাংখ্যিক মানগুলোর ওপর বীজগাণিতিক যোগবিয়োগ ব্যবহার করা যায়। যেমন:

```
'a' - 'a' = 0
'b' - 'a' = 1
'c' - 'a' = 2
'd' - 'a' = 3
```

তবে এই কোডটি চালানোর আগে অবশ্যই count ভেরিয়েবলের সবগুলো ইনডেক্সের মান শূন্য করে নিতে হবে। আর প্রিন্ট করার সময় খেয়াল করতে হবে যাদের মান শূন্য সেগুলো যেন আমরা প্রিন্ট না করি:

```
for(i = 0; i < 26; i++)
{
    if(count[i] != 0)
    {
        printf("%c = %d\n", 'a'+i, count[i]);
    }
}
```

লক্ষ করো, আমরা আউটপুট প্রিন্ট করার সময়ও 'a'+i কে %c দিয়ে প্রিন্ট করেছি। এটা কীভাবে কাজ করছে তা তোমরা এই চার্টটা দেখলে বুঝতে পারবে।

i	'a'+i	%d	%c
0	97+ 0 = 97	97	'a'
1	97+ 1 = 98	98	'b'
2	97+ 2 = 99	99	'c'
3	97+ 3 = 100	100	'd'
...
25	97+25 = 122	122	'z'

আশা করি, তোমরা বিষয়টি বুঝতে পেরেছ। তাহলে এখন এই সমস্যাটি সমাধান করে সাবমিট করে ফেলো।

সমস্যা ১৬ – শব্দ বিপর্যয়

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে, যেটি একটি বাক্যের সবগুলো শব্দকে উল্টো করে দেখাবে।
[\(<http://bit.ly/1-52-problem-16>\)](http://bit.ly/1-52-problem-16)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$)। এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি স্ট্রিং S (S এর দৈর্ঘ্য 1000 এর বেশি নয়)।

আউটপুট

প্রোগ্রামটির আউটপুটে S এর প্রত্যেকটি শব্দকে উল্টো করে প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
4 This is a test Hello World don't underestimate the power of a girl	sihT si a tset olleH dlrow t'nod etamitserednu eht rewop fo a lrig

সমাধান

এটি একটি স্ট্রিং-সংক্রান্ত সমস্যা। সমস্যাটির সমাধান করতে হলে তোমাদের স্ট্রিং সম্পর্কে পরিষ্কার ধারণা থাকতে হবে। তো আমি আশা করছি স্ট্রিংয়ের মৌলিক বিষয়গুলো তোমরা জানো।

সমস্যার সমাধান করতে গিয়ে প্রথমেই আমরা ইনপুট নেওয়ার ব্যবস্থা করে ফেলি। তোমরা কেউ কেউ `gets(S)` ফাংশন ব্যবহার করে এক লাইনের স্ট্রিং S ইনপুট নাও, কিন্তু এটি ব্যবহার করতে গেলে কিছু সমস্যার পরবে। কী সমস্যা, সেটি তোমরা `gets()` ব্যবহার করার চেষ্টা করলেই বুঝতে পারবে। এর চেয়ে ভালো হয় `scanf(" %[^\n]", S)` ব্যবহার করলে। লক্ষ করো, % চিহ্নের আগে কিন্তু একটি স্পেস আছে। সেটি দেওয়ার কারণ হচ্ছে, পূর্বের লাইনের শেষে যেই নিউলাইন ক্যারেক্টোর (মানে '\n') আছে, সেটি যেন এর ভেতরে ঢুকে যায়। এখানে স্পেস-এর মানে হচ্ছে হোয়াইটস্পেস, তাই নিউলাইন ক্যারেক্টোরও এর ভেতরে ঢুকে যাবে। তাহলে আমরা নিচের কোড নিখে ফেলতে পারি:

```

#include<stdio.h>

int main()
{
    int i, T;
    char S[1002], word[1002];

    scanf("%d", &T);
    for (i = 0; i < T; i++)
    {
        scanf(" %[^\n]", S);
        printf("%s\n", S);
    }

    return 0;
}

```

এখানে একবার আমরা প্রিন্ট করেও দেখছি যে ইনপুট যা নিছি, তা ঠিকঠাক নিছি কি না। যদি ঠিকঠাক থাকে, তাহলে তোমরা প্রিন্ট করার লাইনটি মুছে দেবে।

এরপরের কাজ হচ্ছে একটি লাইনে যে এক বা একাধিক শব্দ আছে, সেগুলো পৃথক করা। সেটা আমরা একটি লুপের মাধ্যমে করে ফেলতে পারি। সমস্যার বর্ণনায় বলা আছে যে শব্দগুলো এক বা একাধিক স্পেস ক্যারেক্টোর দিয়ে পৃথক করা আছে। তাহলে আমাদের লজিক কী হতে পারে? S স্ট্রিংয়ের প্রতিটি বর্ণ একটি লুপের সাহায্যে পরীক্ষা করতে থাকব (শুরু থেকে শেষ পর্যন্ত)। যদি সেটি স্পেস ভিন্ন অন্য কোনো অক্ষর হয়, তাহলে ওই বর্ণটি word অ্যারেতে বা স্ট্রিংয়ে রাখব। আর যখন S স্ট্রিংয়ের অক্ষরগুলো পরীক্ষা করতে করতে কোনো স্পেস ক্যারেক্টোর পেয়ে যাব, তার মানে আমরা একটি শব্দ পেয়ে গিয়েছি এবং আপাতত সেই শব্দটি প্রিন্ট করব।

```

for (j = 0, k = 0; j < strlen(S); j++)
{
    if (S[j] != ' ')
    {
        word[k] = S[j];
        k++;
    }
    else if (k > 0)
    {
        word[k] = '\0';
        printf("%s\n", word);
        k = 0;
    }
}

```

```
}
```

এখানে দেখো, শব্দ প্রিন্ট করার আগের লাইনে আমি লিখেছি `word[k] = '\0';` এর কারণ কী? লক্ষ করো, আমরা k ভেরিয়েবলটি ব্যবহার করছি অ্যারের ইনডেক্স হিসেবে। প্রতিবার `word[k]-তে` যখন `s[j]` অ্যাসাইন করছি, পরের লাইনে k-এর মান এক বাড়িয়ে দিচ্ছি। অর্থাৎ পরেরবার `word[k]-তে` যখন কিছু অ্যাসাইন করব, সেটি অ্যারের পরবর্তী ঘরে যাবে। নইলে প্রতিবার একই ঘরে নতুন নতুন মান অ্যাসাইন হবে। আর আমরা যখন `else if` রকমে চুকছি, `word[k]-এর` মধ্যে '`\0`' অ্যাসাইন করে কম্পাইলারকে বুঝিয়ে দেওয়া হচ্ছে যে স্ট্রিংটি এখানেই শেষ। এটি বলে না দিলে কম্পাইলার সঠিকভাবে বুঝতে পারবে না যে স্ট্রিংটি কোথায় শেষ হয়েছে। আবার দেখো, `word` প্রিন্ট করার পরের লাইনে আমি লিখেছি `k = 0;` তার মানে এরপর যখন `word[k]-তে` কোনো অক্ষর রাখা হবে, সেটি আবার অ্যারের প্রথম ঘর থেকে শুরু হবে। এখন যদি তোমরা প্রোগ্রামটি রান করো, দেখবে যে ইনপুটে যেই লাইন দিচ্ছ, তার শেষ শব্দটি প্রিন্ট হচ্ছে না। কেন?

কারণ শেষ শব্দের বেলায় `else if` রকমের ভেতরে ঢোকার আগেই `for` লুপের কন্ডিশন (`j < s_len`) মিথ্যা বা `false` হয়ে যাচ্ছে, তাই সে লুপ থেকে বেরিয়ে আসছে। তাহলে লুপের শেষে আমাদের পরীক্ষা করতে হবে যে `word`-এ কিছু আছে কি না। এটি বোঝার সহজ উপায় হচ্ছে `k`-এর মান পরীক্ষা করা। `k` যদি 0-এর চেয়ে বড় হয়, তার মানে `word`-এ আমরা কিছু রাখা শুরু করেছিলাম।

```
if (k > 0)
{
    word[k] = '\0';
    printf("%s\n", word);
}
```

তাহলে এখন পর্যন্ত আমাদের কোড দাঁড়াচ্ছে এমন -

```
#include<stdio.h>
#include<string.h>

int main()
{
    int i, j, k, t, s_len;
    char S[1002], word[1002];

    scanf("%d", &t);
    for (i = 0; i < t; i++)
```

```

{
    scanf(" %[^\n]", S);
    s_len = strlen(S);
    for (j = 0, k = 0; j < s_len; j++)
    {
        if (S[j] != ' ')
        {
            word[k] = s[j];
            k++;
        }
        else if (k > 0)
        {
            word[k] = '\0';
            printf("%s\n", word);
            k = 0;
        }
    }
    if (k > 0)
    {
        word[k] = '\0';
        printf("%s\n", word);
    }
}

return 0;
}

```

প্রোগ্রাম রান করে দেখো সব ঠিকঠাক আছে কি না। এখন আমরা প্রতিটি শব্দ আলাদাভাবে প্রিন্ট করতে পারছি। কিন্তু আমাদেরকে শব্দগুলো উল্টো প্রিন্ট করতে হবে। তাই উল্টো প্রিন্ট করার জন্য একটি ফাংশন লিখে ফেলি।

```

void print_reverse(char str[])
{
    int i;
    for (i = strlen(str) - 1; i >= 0; i--)
    {
        printf("%c", str[i]);
    }
}

```

এখন কোডে `printf("%s\n", word);` এর জায়গায় `print_reverse(word);` লিখতে হবে। তারপরে তুমি যদি তোমার কোডটি রান করো, আউটপুট হিজিবিজি মনে হবে। এর কারণ হচ্ছে শব্দগুলোর মধ্যে স্পেস দাও নি। তাই পরপর দুটি শব্দের মধ্যে স্পেস (একটি স্পেস

ক্যারেষ্টার) প্রিন্ট করার কাজটি তোমার ওপরই ছেড়ে দিলাম। তবে সবার শেষে যেই শব্দ, তার পরে কিন্তু স্পেস ক্যারেষ্টার দেওয়া চলবে না, দিলে রং অ্যানসার হবে।

আরেকটি জিনিস তোমাকে একটু ভাবতে বলব। প্রোগ্রামে যে দুটি স্ট্রিং ব্যবহার করেছি, দুটি সাইজই 1000-এর চেয়ে বেশি দিয়েছি, বিশেষ করে word-এর সাইজও। এটি কেন করা হলো?

সমস্যা ১৭ – স্বরবর্ণ গণনা

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে, যেটি একটি বাক্যে কতগুলো স্বরবর্ণ বা vowel আছে সেটি গণনা করবে।

(<http://bit.ly/1-52-problem-17>)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 100$). এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি স্ট্রিং S (S এর দৈর্ঘ্য 1000 এর বেশি নয়)। স্ট্রিংয়ের সবগুলো বর্ণই ছোটো হাতের অক্ষরে লেখা থাকবে।

আউটপুট

প্রোগ্রামটির আউটপুটে স্ট্রিং S-এর মধ্যে স্বরবর্ণের সংখ্যা (Number of vowels) প্রিন্ট করতে হবে। '=' চিহ্নের আগের এবং পরের স্পেস লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
3 i am a programmer happy coding hello world	Number of vowels = 6 Number of vowels = 3 Number of vowels = 3

সমাধান

সমস্যার বর্ণনায় বলা আছে যে একটি বাক্যে কতগুলো স্বরবর্ণ (vowel) আছে, সেটি বের করতে হবে। সমস্যার বর্ণনায় যদিও বলা নেই, কিন্তু আমরা ধরে নিতে পারি যে একটি বাক্য এক লাইনে থাকবে, যদিও এটি বাক্যের সঠিক সংজ্ঞা নয়। প্রোগ্রামিং প্রতিযোগিতায় কোনো সমস্যার বর্ণনায় এ রকম কনফিউশন থাকলে জাজের কাছে প্রশ্ন পাঠানো যায়। আর স্বরবর্ণ কোনগুলো সেটি মনে না থাকলে আমি বলে দিই - a, e, i, o u-এই পাঁচটি হচ্ছে ইংরেজি স্বরবর্ণ।

টেস্ট কেসের সংখ্যা ও স্ট্রিংটি ইনপুট নেওয়ার সময় `scanf()` ফাংশন ব্যবহার করা যেতে পারে। "শব্দ বিপর্যয়" সমস্যার আলোচনায় ইনপুট নেওয়ার পদ্ধতি দেওয়া আছে।

এখন বাকি কাজটুকু এসো নিজে করি। সমস্যার সমাধান জমা দেওয়ার পর রং অ্যানসার পেলে আউটপুট ফরম্যাট ঠিক আছে কি না, সেটা পরীক্ষা করতে হবে, আর অ্যারে সাইজও ঠিক আছে কি না, সেটা দেখতে হবে।

সমস্যা ১৮ – স্বরবর্ণ – ব্যাঞ্জনবর্ণ

.সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি বাক্যে অবস্থিত স্বরবর্ণ (vowels) ও ব্যাঞ্জনবর্ণ (consonants) গুলোকে আলাদা আলাদা প্রিন্ট করবে।
[\(http://bit.ly/1-52-problem-18\)](http://bit.ly/1-52-problem-18)

ইনপুট

ইনপুটের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($1 \leq T \leq 100$). এরপরে T সংখ্যক লাইন থাকবে। প্রতিটি লাইনে থাকবে একটি স্ট্রিং S। স্ট্রিংয়ের সবগুলো বর্ণই ছোটো হাতের অক্ষরে লেখা থাকবে।

আউটপুট

প্রোগ্রামটির আউটপুটে স্ট্রিংটিতে অবস্থিত স্বরবর্ণ ও ব্যাঞ্জনবর্ণ গুলোকে আলাদা আলাদাভাবে পরপর প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2 this is very easy it is a rainy sunday	iieea thssvrysy iiaaisa tsrnysndy

সমাধান

সমস্যার বর্ণনায় বলা আছে একটি বাক্যে কতগুলো স্বরবর্ণ ব্যাঞ্জনবর্ণ আছে সেগুলো বের করতে হবে। অর্থাৎ vowel এবং consonants গুলো আলাদা আলাদাভাবে প্রিন্ট করতে হবে। তোমরা এর আগে "স্বরবর্ণ গণনা" যে সমস্যাটির সমাধান দেখেছ, সেখানে স্বরবর্ণ বা vowel সম্পর্কে বলা আছে।

এবার আমরা ইনপুটের বর্ণনায় আসি। ইনপুটের বর্ণনায় বলা আছে প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T, পরবর্তী T সংখ্যক লাইনে ইনপুট নিতে হবে একটি স্ট্রিং। যেহেতু এখানে স্ট্রিং-এর সাইজের ব্যাপারে কোনো উল্লেখ করা হয়নি, আমরা সাইজ 10000 ধরে নিয়ে কাজ করব।

আউটপুটের বর্ণনায় বলা আছে স্বরবর্ণ এবং ব্যাঞ্জনবর্ণ আলাদা আলাদা লাইনে পরপর প্রিন্ট করতে হবে। আমরা এখন আমাদের কোডটি করার জন্য প্রয়োজনীয় ভেরিয়েবল ডিক্রেশার করে দেই।

```
char S[10001];
int i, T;
```

এখানে,

- S[10001]- আমরা যে বাক্যটি ইনপুট স্ট্রিং হিসেবে নিব সেটির জন্য
- i - আমরা যে for লুপটি ব্যবহার করব সেটির জন্য এবং
- T - T সংখ্যক লাইনে আমরা যে বাক্যটি নিতে চাচ্ছ তার জন্য।

প্রথমেই আমাদের ইনপুট নেওয়ার কাজটি করে ফেলতে হবে। অনেক সময় তোমরা gets() ব্যবহার করে ইনপুট নিয়ে থাকো স্ট্রিংয়ের ক্ষেত্রে। কিন্তু, gets() দিয়ে স্ট্রিং ইনপুট নেওয়ার সময় একটু সমস্যা হয়। যেটা তোমরা স্ট্রিং ইনপুট নেওয়ার সময়ই বুঝতে পারবে।

যেহেতু, T সংখ্যক লাইনে ইনপুট নিতে হবে তাই আমাদের প্রথম ইনপুট হবে T.

```
scanf("%d", &T);
```

লুপটি T-এর মান যত ততবার চলবে।

```
for(i = 0; i < T; i++) {}
```

লুপটি যতবার চলবে, প্রতিবার আমরা S স্ট্রিংটি ইনপুট নিব। `scanf(" %[^\n]", s)` ব্যবহার করলে। লক্ষ করো, % চিহ্নের আগে কিন্তু একটি স্পেস আছে। সেটি দেওয়ার কারণ হচ্ছে, আগের লাইনের শেষে যে নিউলাইন ক্যারেক্টার (মানে '\n') আছে, সেটি যেন এর ভেতরে ঢুকে যায়। এখানে স্পেসের মানে হচ্ছে হোয়াইটস্পেস, তাই নিউলাইন ক্যারেক্টারও এর ভেতরে ঢুকে যাবে।

ইনপুট নেওয়ার পর আমরা আরেকটি for লুপ ব্যবহার করব যেটি কিনা, স্ট্রিং S-এ নাল ক্যারেক্টার ('\0') না পাওয়া পর্যন্ত এগিয়ে নিয়ে যাবে। আমাদের লুপটি হবে এমন:

```
for(i = 0; s[i] != '\0'; i++) {}
```

লুপের ভেতরের কাজটি খুবই সহজ। আমরা একটা কন্ডিশন দিয়ে বিচার করে vowel-গুলো আলাদা করে ফেলব এবং আউটপুট হিসেবে সেগুলো প্রিন্ট করব।

```
for(i = 0; s[i] != '\0'; i++)
{
    if(s[i] == 'a' || s[i] == 'e' || s[i] == 'i' ||
       s[i] == 'o' || s[i] == 'u')
    {
        printf("%c", s[i]);
    }
}
```

এবার খেয়াল করে দেখো, উদাহরণ আউটপুটের দিকে। সেখানে consonants প্রিন্ট করার আগে একটি নিউলাইন প্রিন্ট করা আছে। কিন্তু সমস্যার বর্ণনায় সেটি বলা নেই। কিন্তু এটা দিতে হবে। তাই আমরা for লুপটির পরে একটি নিউলাইন প্রিন্ট করব।

```
printf("\n");
```

এখানে আমরা আবার consonants বা ব্যাঞ্জনবর্ণের জন্য আবার একটি for লুপের সাহায্য নিয়েছি। যেটি আগের প্রায় আগের for লুপটির মতোই শুধুমাত্র "==" এর পরিবর্তে সেখানে

আমরা "!=" ব্যবহার করেছি। এখন তোমাদের মনে প্রশ্ন আসতে পারে, if এর পরে else ব্যবহার করলেই তো আমরা consonants বা ব্যঙ্গবর্ণগুলো পেয়ে যেতাম। এই প্রশ্নের উত্তর তোমরা if এর পরে else ব্যবহার করলেই পেয়ে যাবে। আমাদের কোডটি হবে এমন:

```
for(i = 0; S[i] != '\0'; i++)
{
    if(S[i] != 'a' && S[i] != 'e' && S[i] != 'i' &&
       S[i] != 'o' && S[i] != 'u' && S[i] != ' ')
    {
        printf("%c", S[i]);
    }
}
```

খেয়াল করে দেখ আগের শর্তগুলোর শেষে এবারে কিন্তু আমি আরও একটি শর্ত জুড়ে দিয়েছি। এবার তোমাদের কাজ হলো কোডটি গুছিয়ে সুন্দর করে লিখে ফেলা।

সমস্যা ১৯ – শব্দ গণনা ১

সমস্যার বিবরণ

একটি বাকে অবস্থিত শব্দগুলোর মধ্যকার স্পেস হিসাব করে মোট শব্দ করগুলো আছে সেটি বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-19>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে ইনপুট হবে একটি স্ট্রিং S। যেখানে, S-এর সর্বোচ্চ দৈর্ঘ্য হবে 10000। প্রতিটি স্ট্রিংয়ে দুটি শব্দ একটিমাত্র স্পেস (' ') ক্যারেক্টার দিয়ে আলাদা করা থাকবে। স্ট্রিংটিতে ইংরেজী বর্ণ ও স্পেস ছাড়া অন্য কোনো ক্যারেক্টার থাকবে না।

আউটপুট

প্রোগ্রামটির আউটপুটে S-এ মোট কতগুলো শব্দ আছে সেটি গণনা করে দেখাতে হবে। আউটপুটে "
=" চিহ্নের আগে এবং পরে একটি করে স্পেস দিতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3 Hello World Dhaka is the capital of Bangladesh It is the winter of discontent	Count = 2 Count = 6 Count = 6

সমাধান

এক লাইনের একটি স্ট্রিংয়ের মাঝখানের স্পেসগুলোর থেকে আমরা ওই স্ট্রিংয়ে কতগুলো শব্দ আছে সেটা বের করতে পারি। এখানে আমাদের S স্ট্রিংটিকে আমরা যদি প্রতিটি substring এ আলাদা আলাদা করে একটি ভেরিয়েবলে হিসাব করি তাহলেই আমরা মোট কতটি শব্দ আছে সেটি পেয়ে যাব।

আমাদের বিবেচনায় আনতে হবে স্পেস কতটি আছে সেটি গণনা করা। ধরা যাক, একটি স্ট্রিং আমরা ইনপুট নিয়েছি "Hello World"। এখানে দুইটি শব্দের মধ্যে স্পেস আছে একটি। তাহলে আমাদের শব্দের সংখ্যা হবে আমাদের স্পেসের সংখ্যার থেকে এক বেশি।

আমরা একটা লুপ চালাব যেটি স্ট্রিং S নাল ক্যারেক্টার না পাওয়া পর্যন্ত চলবে এবং স্পেসের সংখ্যা গণনা করে প্রিন্ট করবে। কোডের এই অংশটুকু হবে এমন:

```
int count = 0;
for(i = 0; S[i] != '\0'; i++)
{
    if(S[i] == ' ')
    {
        count++;
    }
}
printf("%d", count+1);
```

তবে খেয়াল করো, যদি দুটি শব্দের মধ্যে একাধিক স্পেস ক্যারেক্টার থাকে তখন কিন্তু আবার এই প্রোগ্রাম কাজ করবে না। কোডের বাকি অংশটুকু লেখা তোমাদের কাজ। পুরো কোডটি লিখে নমুনা ইনপুট/আউটপুটের সঙ্গে তোমার কোড মিলিয়ে দেখো সব ঠিকঠাক দেখায় কি না। তারপর অনলাইন জাজে জমা দাও।

সমস্যা ২০ – শব্দ গণনা ২

সমস্যার বিবরণ

একটি বাকে মোট কতগুলো শব্দ আছে সেটি বের করার প্রোগ্রাম লিখতে হবে। দুটি শব্দের মধ্যে
যেকোনো যতি চিহ্ন থাকতে পারে। অর্থাৎ দুটি শব্দের মধ্যে স্পেস ক্যারেক্টার ছাড়াও কমা,
সেমিকোলন ইত্যাদি থাকতে পারে।

(<http://bit.ly/1-52-problem-20>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা
নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে প্রোগ্রামটির ইনপুট হবে একটি স্ট্রিং S। স্ট্রিং এর সর্বোচ্চ
সীমা হবে 10000।

আউটপুট

প্রোগ্রামটির আউটপুটে মোট কতগুলো শব্দ স্ট্রিং S-এ আছে সেটি count করে দেখাতে হবে।
আউটপুটে "=" চিহ্নের আগে এবং পরে একটি করে স্পেস দিতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2 Hurrah! was heard on all sides Hello, I'm Brooker and you're watching TV	Count = 6 Count = 7

সমাধান

শব্দ গণনা-১ সমস্যাটির মতোই এই সমস্যাটি। এখানে স্পেস ছাড়াও আমাদের আরও অন্যান্য
যতি চিহ্ন নিয়ে কাজ করতে হবে। অর্থাৎ যদি কোনো বাকে কমা (,) বা সেমিকোলন (;) ইত্যাদি
থাকে তাহলে সেসব বিবেচনার এনে আমাদের প্রতিটি শব্দ আলাদা করতে হবে। এখানে আমরা
প্রতিটি শব্দ আলাদা করার জন্য strtok() ফাংশনের সাহায্য নিতে পারি।

strtok() ফাংশন দিয়ে একটি স্ট্রিং-কে ছোট ছোট ভাগে ভাগ করা যায়, যেটাকে string
tokenization বলা হয়ে থাকে। strtok() ফাংশনটির দুটি প্যারামিটার। প্রথম প্যারামিটার
হবে যে স্ট্রিং-টি আমরা ইনপুট হিসেবে নেব এবং দ্বিতীয় প্যারামিটার হলো যেসব যতি চিহ্ন দিয়ে
আমরা শব্দগুলোকে আলাদা করব। প্রথম শব্দটি আমরা এভাবে পেয়ে যাব এবং সেটি একটি

ক্যারেন্টার পয়েন্টার ভেরিয়েবলে রাখব। স্ট্রিংের বাকি শব্দগুলো পাওয়ার জন্য আমরা একটি লুপ চালাব NULL পর্যন্ত।

strtok() দিয়ে কোডটি হবে এমন:

```
word = strtok(s,",!;?. ");
count = 0;
while(word != NULL)
{
    if(strlen(word) > 0)
    {
        count++;
    }
    word = strtok(NULL, ",!;?. ");
}
printf("%d\n",count);
```

এখানে word হচ্ছে **char *word**। কোডটির প্রথম লাইনে আমরা স্ট্রিংটির প্রথম শব্দটি আলাদা করেছি। এরপর আরও একটি ক্যারেন্টার পয়েন্টার token-এর সাহায্যে আমরা স্ট্রিংটি NULL পয়েন্টার না পাওয়া পর্যন্ত পরীক্ষা করেছি। word-এর লেংথ শূন্য না হওয়া পর্যন্ত count করেছি মোট কতগুলো শব্দ আমরা পাচ্ছি।

বাকি কাজ এখন তোমাদের।

সমস্যা ২১ – উল্টো দেখা

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখ, যেটি একটি শব্দকে উল্টো করে দেখাবে।
(<http://bit.ly/1-52-problem-21>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে প্রোগ্রামটির ইনপুট হবে একটি স্ট্রিং S। স্ট্রিং এর সর্বোচ্চ দৈর্ঘ্য হবে 1000।

আউটপুট

প্রোগ্রামটির আউটপুটে, S স্ট্রিংটিকে উল্টো করে প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	gnirts
string	gnimmargorp
programming	anurA
Aruna	

সমাধান

সমস্যাটির বর্ণনায় বলা আছে ইনপুট হবে একটি পূর্ণ সংখ্যা T এবং একটি স্ট্রিং S; যেখানে স্ট্রিং S এর সর্বোচ্চ দৈর্ঘ্য হবে 1000. এখন প্রথম ইনপুট T এর জন্য আমরা ডেটা টাইপ হিসেবে নিব int এবং স্ট্রিং ইনপুটের জন্য নিব একটি ক্যারেক্টোর অ্যারে।

```
int T;
char S[1001];
```

কী ধরনের ডেটা স্ট্রাকচার ব্যবহার করে ইনপুট নিতে হবে সেটি তো আমরা ঠিক করলাম। এখন ইনপুট নেওয়ার পালা। যেহেতু বলা আছে T সংখ্যক লাইনে ইনপুট নিতে হবে, সেহেতু আমরা প্রথমে ইনপুট নিব T,

```
scanf("%d", &T);
```

T সংখ্যক লাইনে ইনপুট নেওয়ার জন্য আমাদের একটি লুপ ব্যবহার করতে হবে যেটি T সংখ্যাক্রমের চলবে। সেক্ষেত্রে আমরা **while** অথবা **for** লুপ ব্যবহার করতে পারি।

while(T--){} অথবা for(int i = 0; i < T; i++){}

এবার স্ট্রিং ইনপুট নেওয়ার ব্যাপারটি আগের অনেকগুলো সমস্যায় বলা আছে। মনে না থাকলে সেটা দেখে নিতে পারো। এই ব্যাপারে তাই এখানে বিস্তারিত বললাম না।

এখানে সমস্যাটির সমাধানের জন্য লুপ ব্যবহার করেছি। এজন্য, প্রথমেই আমাদের দরকার ইনপুট স্ট্রিং-টির দৈর্ঘ্য। দৈর্ঘ্য বের করার জন্য এখানে আমরা **strlen()** ফাংশন ব্যবহার করেছি। স্ট্রিং এর দৈর্ঘ্য থেকে শুরু করে এক এক করে কমিয়ে আমরা লুপটিকে শূন্যতে এনে শেষ করেছি এবং স্ট্রিংটি প্রিন্ট করেছি।

```
#include<stdio.h>
#include<string.h>

int main()
{
    .
    char S[100];
    int i, T, len;
    scanf("%d", &T);

    while(T--)
    {
        scanf(" %[^\n]", S);
        len = strlen(S);
        for(i = len-1; i >= 0; i--)
        {
            printf("%c", S[i]);
        }
    }

    return 0;
}
```

এই কোডটি লিখে রান করলে তোমরা দেখবে কোডটা ঠিকমতো কাজ করে কি না। যদি না করে ভুলগুলো নিজে নিজে ঠিক করে জাজ করার জন্য জমা দেবে।

সমস্যা ২২ – মৌলিক সংখ্যা

সমস্যার বিবরণ

প্রাইম বা মৌলিক সংখ্যা হচ্ছে 1 থেকে বড় সেইসব সংখ্যা যেগুলোকে 1 এবং ওই সংখ্যাটি ছাড়া অন্য কোনো সংখ্যা দিয়ে ভাগ করা যায় না। যেমন: 11, 13, 17 ইত্যাদি। এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি নির্দিষ্ট সীমা পর্যন্ত কতগুলো প্রাইম বা মৌলিক সংখ্যা আছে সেটি প্রকাশ করবে।

(<http://bit.ly/1-52-problem-22>)

ইনপুট

প্রোগ্রামটির ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে প্রোগ্রামটির ইনপুট হবে দুইটি পূর্ণসংখ্যা a ও b ($1 \leq a \leq b \leq 100000$)।

আউটপুট

প্রোগ্রামটির আউটপুটে a থেকে b পর্যন্ত কতগুলো মৌলিক সংখ্যা আছে তা প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	3
1 5	6
25 50	4
12 27	

সমাধান

এই সমস্যার সমাধান করতে গিয়ে সবচেয়ে সহজ পদ্ধতি হচ্ছে, ইনপুটে যে দুটি সংখ্যা থাকবে (a ও b , যেখানে $a \leq b$), একটি লুপ চালিয়ে তাদের শুরু থেকে শেষ পর্যন্ত প্রতিটি সংখ্যা মৌলিক কি না পরীক্ষা করা এবং মৌলিক হলে সেটি গণনা করা। অবশেষে মোট কয়টি মৌলিক সংখ্যা পাওয়া গেল, সেটি প্রিন্ট করা। তবে এটি মোটেও ভালো কোনো পদ্ধতি নয়। প্রোগ্রামিং প্রতিযোগিতার সময় তোমরা এভাবে সমাধান করলে টাইম লিমিট এক্সেডেড এর পাবে।

যেহেতু সংখ্যাগুলো হবে 1 থেকে 100000-এর মধ্যে, তাই আমরা শুরুতেই সিভ মেথডের সাহায্যে এই সীমার মধ্যে কোনগুলো মৌলিক সংখ্যা আর কোনগুলো মৌলিক সংখ্যা নয়, সেগুলো

বের করে ফেলব। তারপরে আমরা ইনপুট নেওয়া শুরু করব আর একটি লুপ চালিয়ে বলে দিতে পারব যে ওই সীমার মধ্যে কয়টি মৌলিক সংখ্যা আছে। তোমরা যদি মৌলিক সংখ্যা বের করার সিভ মেথডের সঙ্গে পরিচিত না হয়ে থাক, তাহলে "কম্পিউটার প্রোগ্রামিং - ১ম খণ্ড" বইটির মৌলিক সংখ্যা অধ্যায়টি ভালো করে পড়ে নিও।

তাহলে তোমাদের কোড দাঁড়াবে অনেকটা এ রকম:

```
#include<stdio.h>
#define size 100001
char ara[size];

void sieve()
{
    int i, j, root;
    for(i = 2; i < size; i++)
    {
        ara[i] = 1;
    }
    root = sqrt(size);
    for(i = 2; i <= root; i++)
    {
        if(ara[i] == 1)
        {
            for(j = 2; i * j <= size; j++)
            {
                ara[i * j] = 0;
            }
        }
    }
}

int main()
{
    int T, a, b, count;
    sieve();

    scanf("%d", &T);
    while(T--)
    {
        scanf("%d %d", &a, &b);
        count = 0;
        for(i = a; i <= b; i++)
        {
            if(ara[i])
```

```

    {
        count++;
    }
    printf("%d\n", count);
}
return 0;
}

```

এখানে দেখো আমি শুরুতেই সিভ মেথড কল করেছি, এবং এই ফাংশনের কাজ শেষ হলে arr-
এর ভেতর যেসব সংখ্যা মৌলিক, সেখানে থাকবে 1 আর যেসব সংখ্যা মৌলিক নয়, সেখানে
থাকবে 0। এই কাজটি আমার মাত্র একবারই করা লাগবে। তারপরে যতবারই ইনপুট নেওয়া
হোক না কেন, আমি সেই অ্যারের ভেতর একটি লুপ চালিয়ে বলে দিতে পারব ওই সীমার ভেতরে
কয়টি মৌলিক সংখ্যা আছে। এ রকম মাঝে মধ্যে তোমরা কিছু প্রোগ্রামিং সমস্যা পাবে, যেখানে
আগে-ভাগে সবকিছু হিসাব করে একটি অ্যারেতে রেখে দিলে খুব সুবিধা হয়। আশা করি, এই
পদ্ধতিটি তোমরা আয়ত্তে এনে ফেলতে পারবে।

সমস্যা ২৩ – বর্ণমালা থেকে সংখ্যা

সমস্যার বিবরণ

ইংরেজি বর্ণমালা A,B,C,D... এর সঙ্গে আমাদের পরিচয় হয় শৈশবেই। আমরা জানি ইংরেজি
বর্ণমালা ২৬টি। তোমার কাজ হচ্ছে A=1, B=2, ... Z=26 বিবেচনা করে এমন একটি প্রোগ্রাম
লেখা যেটি একটি বর্ণগুচ্ছকে সাংখ্যিকগুচ্ছে রূপান্তর করবে। যেমন: ABZ=1226.
(<http://bit.ly/1-52-problem-23>)

ইনপুট

প্রোগ্রামটির ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ
করে। পরবর্তী T সংখ্যক লাইনে প্রোগ্রামটির ইনপুট হবে একটি স্ট্রিং S (S এর দৈর্ঘ্য 100 এর
বেশি নয়)।। স্ট্রিংয়ের সবগুলো বর্ণই বড় হাতের অক্ষরে লেখা থাকবে।

আউটপুট

প্রোগ্রামটির আউটপুটে স্ট্রিং S-এর প্রতিটি অক্ষরের সাংখ্যিক মান প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	1226
ABZ	26252
ZYB	31549147
CODING	

সমাধান

দেখতে কঠিন মনে হলেও আসলে এটি খুবই সহজ একটি প্রোগ্রামিং সমস্যা। ASCII কোডের সাহায্যে আমরা এই সমস্যাটির সমাধান করব। এখন কথা হচ্ছে ASCII কোড কী? ক্যারেক্টারের ASCII কোড কিন্তু আলাদা আলাদা। যেমন: 'A'-এর ASCII কোড 65 এবং 'a'-এর ASCII কোড 97। সমস্যাটি ভালভাবে লক্ষ করলে তোমরা দেখবে এখানে শুধুমাত্র uppercase letters অর্থাৎ সাধারণ বাংলায় আমরা যেগুলোকে বড় হাতের অক্ষর বলে থাকি সেগুলো নিয়ে কাজ করা হয়েছে। সেক্ষেত্রে কাজ আমাদের আরও সহজ হয়ে গেল। আমাদের শুধুমাত্র uppercase অক্ষরগুলোর ASCII কোড নিয়ে কাজ করতে হবে। ASCII কোডের মানগুলো ক্রমবর্ধমান, অর্থাৎ 'A'-এর কোড 65, 'B'-এর কোড 66 এভাবে বাড়তে থাকবে। এখন এখান থেকে ইংরেজি 26টি বর্ণের জন্য ASCII কোড আমরা বের করে ফেলতে পারব। তাহলে এই কোডটি করার জন্য আমাদের A-Z পর্যন্ত 26টি বর্ণের ASCII কোড নিয়ে কাজ করতে হবে।

প্রোগ্রামটির আউটপুটের বর্ণনায় বলা আছে প্রতিটি অক্ষরের সাংখ্যিক মান প্রিন্ট করতে হবে। মানে 'A' = 1, 'B' = 2, 'C' = 3,..., 'Z' = 26। প্রথম নমুনা ইনপুট/আউটপুট খেয়াল করে দেখো: ABZ ইনপুট হলে আউটপুট 1226। যা 'A', 'B' এবং 'Z'-এর সাংখ্যিক মান।

যদি 'A'-এর ASCII কোড 65 থেকে 64 বিয়োগ করে দেই, তাহলে আমরা 'A'-এর সাংখ্যিক মান 1 পেয়ে যাব। আবার যদি 'B'-এর ASCII কোড 66 থেকে আমরা 64 বিয়োগ দেই, আমরা 'B'-এর সাংখ্যিক মান 2 পাব। অর্থাৎ, প্রতিটি ASCII কোড থেকে 64 বিয়োগ দিলে আমরা অক্ষরগুলোর সাংখ্যিক মান পেয়ে যাব।

আমাদের যেহেতু স্ট্রিং নিয়ে কাজ করতে হবে, তাই প্রথমেই আমাদের স্ট্রিং ইনপুট নিতে হবে। আমরা স্ট্রিংটিতে '0' না পাওয়া পর্যন্ত একটি লুপ চালাব। আমাদের 'A' থেকে 'Z' পর্যন্ত

অক্ষরগুলোকে বিবেচনায় আনার জন্য একটি কন্ডিশন ব্যবহার করতে হবে। আমাদের কোডটি হবে এমন:

```
for(i = 0; s[i] != '\0'; i++)
{
    if(s[i] >= 'A' && s[i] <= 'Z')
    {
        printf("%d", s[i] - 64);
    }
}
```

ভালভাবে লক্ষ করো `printf()` ফাংশনটিতে আমি ইন্টিজারের ফরম্যাট স্পেসিফিয়ার `%d` ব্যবহার করেছি। কিন্তু কেন? কারণ, আমাদের অক্ষরগুলোর সাংখ্যিক মান (Numerical Value) দরকার এজন্য আমাদের ক্যারেক্টারটির ASCII কোড জানা জরুরি। প্রিন্ট ফাংশনে আমরা সেই কাজটি করেছি। ক্যারেক্টারের ASCII কোড থেকে 64 বিয়োগ করে দিয়েছি।

এখন তোমরা প্রোগ্রামটি লিখে পরীক্ষা-নিরীক্ষা করো এবং অনলাইন জাজে জমা দাও। আশা করি তোমার সমাধান সঠিক হবে।

একটি প্রোগ্রামিং সমস্যা আমরা অনেকভাবেই সমাধান করতে পারি। ওপরে তো তোমরা আসকি কোড নিয়ে কাজ করলে, এখন দেখা যাক অন্য কোনো বুদ্ধি বের করা যায় কি না। নিচের কোডটি লক্ষ করো:

```
#include<stdio.h>

int main()
{
    char c1, c2, c3;
    c1 = 'A';
    c2 = 'B';
    c3 = 'C';

    printf("%d, %d, %d\n", c1 - 'A', c2 - 'A', c3 - 'A');

    return 0;
}
```

প্রোগ্রামটি রান করে কয়েক মিনিট চিন্তা করলেই তোমরা বুঝে ফেলবে কত সহজে আমরা সমস্যাটির সমাধান করে ফেলতে পারি, আর পুরো কোডটি করতেও তেমন বেগ পেতে হবে না। আর কোডটি সঠিক হল কি না, সেটি অনলাইন জাজে জমা দিলেই বুঝতে পারবে।

সমস্যা ২৪ – একান্তর উপাদান

সমস্যার বিবরণ

একটি অ্যারের উপাদানগুলো থেকে একান্তর উপাদান (Alternate elements) বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-24>)

ইনপুট

প্রোগ্রামটির ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে প্রোগ্রামটির ইনপুট হবে একটি পূর্ণসংখ্যা n ($1 \leq n \leq 100$), যা অ্যারের উপাদান সংখ্যা নির্দেশ করে। পরবর্তীতে অ্যারের n সংখ্যক উপাদান ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে অ্যারের একান্তর উপাদানগুলো পাশাপাশি প্রিন্ট করতে হবে। প্রতিটি উপাদান একটি স্পেস ক্যারেক্টোর দিয়ে আলাদা থাকবে।

উদাহরণ

ইনপুট	আউটপুট
2	1 3 5
5 1 2 3 4 5 10 1 4 55 66 22 0 76 11 23 78	1 55 22 76 23

সমাধান

সমস্যার বর্ণনায় বলা আছে একটি অ্যারের উপাদানগুলো (elements) থেকে একান্তর উপাদানগুলো প্রিন্ট করতে হবে। যদি একটি অ্যারে দেওয়া থাকে যেখানে পাঁচটি উপাদান আছে যথাক্রমে: 1 2 3 4 5। পাঁচটি উপাদানের এই অ্যারেটির একান্তর উপাদানগুলো হবে 1 3 5। বিজোড় সংখ্যাগুলো কিন্তু নয়, এখানে অ্যারের প্রথম ইনডেক্স থেকে শুরু করে প্রতিবার একটি করে উপাদান বাদ দেওয়া হয়েছে। অর্থাৎ, 1 আছে কিন্তু 2 নেই, 3 আছে কিন্তু 4 নেই আবার 5 আছে।

এখানে ইনপুটের বর্ণনা আগের অন্যসব সমস্যার মতোই। T সংখ্যক লাইনে ইনপুট নিতে হবে। প্রথমে ইনপুট নিতে হবে অ্যারের উপাদান সংখ্যা n , মানে কতগুলো সংখ্যা অ্যারেতে থাকবে। এরপর ইনপুট নিতে হবে n সংখ্যক অ্যারে উপাদান।

```

scanf("%d", &n);
for(i = 0; i < n; i++)
{
    //অ্যারে ইনপুট নিতে হবে এখানে
}

```

অ্যারের উপাদান আশা করি এখন তোমরা প্রিন্ট করতে পারো। অ্যারে উপাদান প্রিন্ট করার সময় আমরা for লুপটিকে এক এক করে বৃদ্ধি (increment) করি। যেটি অ্যারের ইনডেক্সের মান। একটি উপাদান বাদ দেওয়ার সময় আমরা যদি লুপটিকে 2 করে increment করি তাহলেই আমাদের কাজ হয়ে যাবে।

একটি উদাহরণ দিলে ব্যাপারটি পরিষ্কার হবে।

```
int num[5] = {1, 2, 3, 4, 5};
```

এখানে,

```

num[0] = 1;
num[1] = 2;
num[2] = 3;
num[3] = 4;
num[4] = 5;

```

প্রথম অ্যারে উপাদানটি অর্থাৎ ইনডেক্স 0 এর উপাদান 1, ইনডেক্সের মান যদি আমরা 2 বাড়াই তাহলে মান ইনডেক্সের বর্তমান মান হবে $i = 0 + 2 = 2$, আমাদের $num[2] = 3$ । আবার আমরা যদি ইনডেক্সের মান আবারও দুই বৃদ্ধি করি তাহলে আমাদের ইনডেক্সের মান হবে, 4। তখন আমরা প্রিন্ট করব $num[4] = 5$ । এবার আমরা যদি অ্যারেরই মানগুলো প্রিন্ট করি তাহলেই আমরা আউটপুটে একান্তর অ্যারে উপাদানগুলো পেয়ে যাব।

```

for(i = 0; i < n; i+=2)
{
    //এখানে আউটপুট প্রিন্ট করতে হবে
}

```

বাকিটা এখন তোমাদের কাজ।

সমস্যা ২৫ – লঘিষ্ঠ সাধারণ গুণিতক

সমস্যার বিবরণ

দুটি সংখ্যার ল.সা.গ. নির্ণয় করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-25>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেস নির্দেশ করে।
প্রতি T সংখ্যক টেস্ট কেসের জন্য প্রোগ্রামটির ইনপুট হবে দুটি পূর্ণসংখ্যা a এবং b ($1 \leq a, b \leq 100000$)।

আউটপুট

প্রোগ্রামটির আউটপুটে a এবং b এর লঘিষ্ঠ সাধারণ গুণিতক বা LCM প্রিন্ট করতে হবে।
আউটপুটে "=" চিহ্নের আগের এবং পরের স্পেস কারেক্টের লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
2	LCM = 30
30 15	LCM = 48
12 16	
3	LCM = 70
14 35	LCM = 12
.4 6	LCM = 105
105 15	

সমাধান

ল.সা.গ বা লঘিষ্ঠ সাধারণ গুণিতকের সঙ্গে তোমরা পরিচিত। স্কুল কলেজে তোমরা এই সম্পর্কিত
অনেক সমস্যা সমাধান করে এসেছ। ল.সা.গ বের করার একটি সহজ সূত্র আছে।

দুটি সংখ্যার ল.সা.গ \times দুটি সংখ্যার গ.সা.গ = দুটি সংখ্যার গুণফল।

তোমরা দুটি সংখ্যার গুণফল অবশ্যই বের করতে পারো। এখন দুটি সংখ্যার গ.সা.গ যদি বের
করা যায় তাহলে এই সূত্রের সাহায্যে আমরা ল.সা.গ বের করতে পারব। ইউক্লিডিয়ান
অ্যালগরিদমের সাহায্যে গ.সা.গ বের করার একটি চমৎকার পদ্ধতি আছে। অ্যালগরিদমটি এমন:

```

function gcd(a, b)
    while b ≠ 0
        t := b;
        b := a mod b;
        a := t;
    return a;

```

একটি উদাহরণ দিলে ফাংশনটির কার্যক্রম আমাদের কাছে পরিষ্কার হবে। ধরা যাক, $a = 12$ এবং $b = 16$, এদের গ.সা.গু বের করতে হবে। অ্যালগরিদম অনুযায়ী:

1. $a=12, b = 16$
2. **while**($b \neq 0$) এভাবে b এর মান যতক্ষণ 0 না হবে লুপটি ততক্ষণ চলবে। অর্থাৎ লুপের ভেতরের তিনটি ধাপ আমরা b এর মান 0 না হওয়া পর্যন্ত বারবার করতে থাকব।
3. $t = 16; b = 12\%16$ এখানে যেহেতু 12 কে 16 দিয়ে ভাগ করা যায় না, তাই ভাগশেষ হবে 12, সুতরাং $b = 12, a = t = 16$ (যেহেতু, প্রথমেই আমরা t এর মান 16 দিয়েছি)
4. তৃতীয় ধাপে আমরা a এবং b এর নতুন মান পেয়েছি, $a = 16$ এবং $b = 12$: এখন আমাদের $t = b = 12; b = a \% b = 16 \% 12 = 4; a = t = 12$
5. এখন $a = 12, b = 4$ এখন $t = b = 4; b = a \% b = 12 \% 4 = 0; a = 4$; এখানে আমরা b এর মান শূন্য পেয়ে গোলাম তাই এই লুপটি এখানেই শেষ হয়ে যাবে। এটি a এর মান **return** করবে 4।

নিচে চার্ট আকারে এই ইটারেশনগুলো দেখানো হলোঃ

$b \neq 0$	t	b	a
		16	12
true	$t \leftarrow b = 16$	$b \leftarrow a \% b$ $\leftarrow 12 \% 16$ $\leftarrow 12$	$a \leftarrow t = 16$
true	$t \leftarrow b = 12$	$b \leftarrow a \% b$ $\leftarrow 16 \% 12$ $\leftarrow 4$	$a \leftarrow t = 12$
true	$t \leftarrow b = 4$	$b \leftarrow a \% b$ $\leftarrow 12 \% 4$ $\leftarrow 0$	$a \leftarrow t = 4$
false	stop	stop	stop

gcd() ফাংশনের কোডটি হবে এমন:

```

int gcd(int a, int b)
{
    int temp;
    while(b != 0)
    {
        temp = b;
        b = a%b;
        a = temp;
    }
    return a;
}

```

আমি এখানে gcd() পর্যন্ত কোড লিখে দিলাম। যেটি দিয়ে তোমরা দুটি সংখ্যার গ.সা.গ বের করতে পারবে। এরপর ল.সা.গ বের করার যে সূত্রটি উপরে আমি উল্লেখ করেছি সেটির সাহায্যে ল.সা.গ বের করার প্রোগ্রামটি বাটপট লিখে অনলাইন জাজে জমা দাও।

সমস্যা ২৬ – এলিয়েন গুপী

সমস্যার বিবরণ

ফাইঁ সসারে করে কিরিমিরি গ্রহ থেকে এলিয়েন গুপী পৃথিবীতে এসেছে। সে সঙ্গে করে নির্দিষ্ট পরিমাণ খাবার নিয়ে এসেছে। গুপী যে গ্রহ থেকে এসেছে সেখানকার প্রাণীরা প্রতিদিনের জন্য সরবরাহ করা খাবারের অর্ধেক একদিনে খেয়ে থাকে। খাবার যখন এক কেজির কম বা সমান হবে গুপী তখন গ্রহে ফিরে যাবে।

তোমার কাজ হচ্ছে এমন একটি প্রোগ্রাম লেখা, যেটি নির্দিষ্ট পরিমাণ খাবার (কেজি-তে) শেষ করতে গুপীর কতদিন লাগবে সেটি বের করা।

(<http://bit.ly/1-52-problem-26>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 1000$) যেটি টেস্ট কেসের সংখ্যা বোঝায়। প্রতিটি টেস্ট কেসে একটি বাস্তব সংখ্যা X ইনপুট নিতে হবে যেটি খাবারের পরিমাণ নির্দেশ করে ($0.0 \leq X \leq 100000.0$)।

আউটপুট

প্রোগ্রামটির আউটপুটে X কেজি খাবার শেষ করতে গুপীর কয়দিন সময় লাগবে সেটি প্রিন্ট করতে হবে “days” হিসেবে।

উদাহরণ

ইনপুট	আউটপুট
3	6 days
40.0	8 days
200.0	9 days
300.0	

সমাধান

খুব সহজ একটি সমস্যা। সমস্যার বর্ণনায় বলা আছে, কিরিমিরি গ্রহের প্রাণীরা প্রতিদিনের সরবহরাহকৃত খাবারের অর্ধেক খায় একদিনে। সরববাহকৃত খাবার শেষ করতে কতদিন লাগবে গুপীর আমাদের সেটি বের করতে হবে। অর্থাৎ গুপী তার গ্রহ থেকে কতদিনের খাবার নিয়ে এসেছে সেটা বের করতে হবে।

প্রোগ্রামটির ইনপুটের বর্ণনায় বলা আছে T সংখ্যক লাইনে ইনপুট নেওয়ার কথা। আমাদের ইনপুট হবে দুটি সংখ্যা- একটি টেস্ট কেসের সংখ্যা T এবং অন্যটি একটি দশমিক সংখ্যা যা খাবারের পরিমাণ নির্দেশ করে।

```
scanf("%d", &T);
for(i = 0; i < T; i++)
{
    scanf("%lf", &X);
}
```

এখানে X হচ্ছে গুপীর সঙ্গে আনা খাবারের পরিমাণ। গুপী যেহেতু রোজ X-এর অর্ধেক খাবার খায়, তাহলে কতদিনে X পরিমাণ খাবার শেষ করতে পারবে সেটি হিসাব করার জন্য আমরা আরেকটি ভেরিয়েবল count ব্যবহার করব। আউটপুট হবে count-এর মোট মান। সমস্যার

বর্ণনায় বলা আছে ১ কেজির কম যখন খাবারের পরিমাণ হয়ে যাবে গুপ্তি তখন তার গ্রহে ফিরে যাবে। তাহলে এখন আমরা একটি লুপ চালাব যেটি কিনা X-এর মান 1.0 থেকে ছোট না হওয়া পর্যন্ত চলবে। X-এর ডেটা টাইপ যেহেতু double তাই আমরা 1.0 লিখেছি।

```
while(X > 1.0)
{
    X = X / 2;
    count++;
}
printf("%d\n", count);
```

আমাদের সম্পূর্ণ কোডটি হবে এমন:

```
#include<stdio.h>

int main()
{
    int T, count;
    double X;
    scanf("%d", &T);

    while(T--)
    {
        scanf("%lf", &X);
        count=0;
        while(X > 1.00)
        {
            X = X / 2;
            count++;
        }
        printf("%d\n", count);
    }

    return 0;
}
```

কোডটি তোমরা লিখে রান করে নমুনা ইনপুট/আউটপুট দিয়ে ভালোভাবে দেখে নাও। যদি কোনো সমস্যা বা অগ্রিম দেখতে পাও নিজেরা চিন্তা করে ঠিক করো।

সমস্যা ২৭ – আর্মস্ট্রং সংখ্যা

সমস্যার বিবরণ

যদি একটি সংখ্যার প্রতিটি অঙ্ককে সংখ্যাটির মোট অঙ্কের সমান পাওয়ার দিয়ে বৃদ্ধি করে, পাওয়ারগুলোকে আবার যোগ করে সেই সংখ্যাটি পুনরায় পাওয়া যায়, তবে সংখ্যাটিকে আর্মস্ট্রং সংখ্যা বলে। সেক্ষেত্রে তিন অঙ্কের একটি সংখ্যার পাওয়ার বা শক্তি হবে 3। যেমন: 153 একটি আর্মস্ট্রং সংখ্যা, কারণ $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ ।

এমন একটি প্রোগ্রাম লিখতে হবে যেটি তিন অঙ্কের একটি সংখ্যা আর্মস্ট্রং সংখ্যা কি না, তা নির্ণয় করবে।

(<http://bit.ly/1-52-problem-27>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে ইনপুট হবে তিন অঙ্কের পূর্ণসংখ্যা n ($100 \leq n \leq 999$)।

আউটপুট

প্রোগ্রামটির আউটপুটে যদি n আর্মস্ট্রং সংখ্যা হয় তাহলে প্রিন্ট করবে "n is an armstrong number!" অন্যথায় প্রিন্ট করবে "n is not an armstrong number!"।

উদাহরণ

ইনপুট	আউটপুট
3	100 is not an armstrong number!
100	153 is an armstrong number!
153	371 is an armstrong number!
371	

সমাধান

সমস্যাটির বর্ণনায় বলা আছে, একটি সংখ্যার প্রতিটি অঙ্ককে তাদের মোট অঙ্কের সমান পাওয়ার বা শক্তি দিয়ে বৃদ্ধি করার পর, অঙ্কগুলোকে আলাদা আলাদাভাবে যোগ করে যদি আবার সেই সংখ্যাটি পাওয়া যায় তবে সেই সংখ্যাটিকে আর্মস্ট্রং সংখ্যা বলে। সমস্যায় একটি উদাহরণ দিয়ে ব্যাপারটা বোঝানো হয়েছে।

সমস্যাটির ইনপুটের বর্ণনার সঙ্গে তোমরা পরিচিত। T সংখ্যক লাইনে ইনপুট নিতে হবে একটি পূর্ণসংখ্যা n। সমস্যায় n-এর রেঞ্জ উল্লেখ করা আছে। যেহেতু বলা হয়েছে তিন অঙ্কের সংখ্যা তাই আমাদের খুব বেশি চিন্তার কিছু নেই। এখানে সবসময়ই আমাদের সংখ্যাটির অঙ্কগুলোর পাওয়ার হবে 3।

এখন তিন অঙ্কের একটি সংখ্যাকে আমরা যদি 10 দিয়ে ভাগ করি তাহলে যদি ভাগশেষ থাকে তবে সেটি এককের ঘরের অঙ্কটি এবং ভাগফল হবে দশক এবং শতকের ঘরের অঙ্কদুটি।

ধরা যাক,

আমাদের সংখ্যাটি, $n = 153$

ভাগশেষ, $rem = 153 \% 10 = 3$

$n = 153 / 10 = 15$

এখন আমাদের ভাগশেষ 3-কে যদি আমরা তিনবার গুণ করি তাহলে আমরা 3^3 পেয়ে যাব। এখানে ভাগ করার পর আমাদের ভাগফল হচ্ছে 15 যেটি আমাদের n-এর নতুন মান হবে। যদি আমরা আবার n-কে 10 দ্বারা ভাগ করি, তখন আমরা ভাগফল পাব 1 এবং ভাগশেষ পাব 5। এই ভাগশেষ 5-কে যদি আমরা তিনবার গুণ করি তাহলে পাব 5^3 । এখন n-এর বর্তমান ভ্যালু 1। 1-কে আবার যদি আমরা 10 দিয়ে ভাগ করি তাহলে ভাগফল হিসেবে পাব শূন্য, কারণ 1-কে 10 দিয়ে ভাগ দেওয়া যায় না। ভাগশেষ পাব 1। 1-কে যদি আমরা তিনবার গুণ করি তাহলে পাব 1^3 ।

এখন আমরা যদি 3^3 , 5^3 , 1^3 যোগ করি তাহলে আমরা আমাদের ইনপুট সংখ্যা n পেয়ে যাব। এখানে তোমরা হয়তো লক্ষ করে দেখেছো, আমরা দুটি কাজ বারবার করেছি। ভাগশেষ এবং ভাগফল বের করা যতক্ষণ না n-এর মান শূন্য হয়। অর্থাৎ আমাদের একটি লুপ ব্যবহার করতে হবে। প্রতিটি অঙ্কের ঘনফলকে যোগ করতে হবে। যদি যোগ করার পর প্রাপ্ত সংখ্যাটি n-এর সমান হয় তাহলে আমরা বলতে পারব n একটি আর্মস্ট্রং সংখ্যা।

আমাদের কোডটি হবে এমন:

```
#include<stdio.h>
int main()
{
    int n, T, res = 0, rem, temp = 0;
    scanf("%d", &T);

    while(T--)
    {
        scanf("%d", &n);
```

```

temp = n;
res = 0;

while(temp!=0)
{
    rem = temp%10;
    res += (rem*rem*rem);
    temp /= 10;
}
return res;
}

```

এখানে আমি আমন্ত্রিং সংখ্যা প্রিন্ট করার জন্য কোনো কিছু লিখে দিইনি। সেটা তোমাদের কাজ।
একটু চিন্তা করে প্রিন্ট করার অংশটুকু তোমরা লিখে ফেলো।

সমস্যা ২৮ – এলোমেলো অ্যারে

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি অ্যারে কোনো ক্রমে সাজানো বা সর্টড আছে
কিনা তা নির্ণয় করবে।

(<http://bit.ly/1-52-problem-28>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা
নির্দেশ করে। পরবর্তীতে T সংখ্যক টেস্ট কেস থাকবে। প্রতিটি টেস্ট কেসের প্রথম লাইনে থাকবে
একটি পূর্ণসংখ্যা n ($n \leq 20$), যেটি অ্যারের উপাদান সংখ্যা নির্দেশ করে। এর পরের n সংখ্যক
লাইনে n সংখ্যক অ্যারের উপাদান ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে অ্যারেটি সাজানো (sorted) কি না সেটি প্রিন্ট করবে। যদি সাজানো হয় তাহলে প্রিন্ট করবে "YES" অন্যথায় "NO".

উদাহরণ

ইনপুট	আউটপুট
2	YES
5	NO
1	
2	
3	
4	
5	
10	
1	
2	
3	
4	
5	
6	
7	
9	
10	
8	

সমাধান

একটি অ্যারে সর্টেড কি না সেটি আমরা কী করে বুবাব? আর সেটি ছোট থেকে বড় নাকি বড় থেকে ছোট ক্রমে সাজানো, সেটিই বা বুবব কী করে?

এই সমস্যা সমাধানে নিচের ধাপগুলো অনুসরণ করা যেতে পারে:

1. সব সংখ্যাগুলো একটি অ্যারেতে ইনপুট নাও।
2. সংখ্যাগুলো ছোট থেকে বড় ক্রমে সাজানো কি না, সেটি বের করার চেষ্টা করো। যদি সাজানো থাকে, তাহলে YES প্রিন্ট করে দাও। আর যদি সাজানো না থাকে তাহলে পরের ধাপে যাও।
3. সংখ্যাগুলো বড় থেকে ছোট ক্রমে সাজানো কি না, সেটি বের করার চেষ্টা করো। যদি সাজানো থাকে, তাহলে YES প্রিন্ট করে দাও। আর যদি সাজানো না থাকে তাহলে NO প্রিন্ট করে দাও।

আর সাজানো কি না সেটি পরীক্ষা করার সময় তোমাদের একটি ফ্ল্যাগ ভ্যারিয়েবল ব্যবহার করতে হবে। ছোট থেকে বড় সাজানো কি না, তা বোঝার কোড অনেকটা এ রকম:

```
int sorted = 1;

for(i = 1; j < n; i++)
{
    if (array[i] < array[i-1])
    {
        sorted = 0;
        break;
    }
}
```

সমস্যার সমাধান করে অনলাইন জাজে জমা দাও। আশা করি, উত্তর সঠিক হবে।

সমস্যা ২৯ – চিহ্ন পরিচয়

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি ক্যারেক্টার uppercase, lowercase, digit নাকি special character সেটি প্রকাশ করবে।
[\(<http://bit.ly/1-52-problem-29>\)](http://bit.ly/1-52-problem-29)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে একটি করে ক্যারেক্টার ch ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে ch ক্যারেক্টারটি uppercase বা বড় হাতের বর্ণ হলে "Uppercase Character" প্রিন্ট করবে, ক্যারেক্টারটি lowercase বা ছোট হাতের বর্ণ হলে "Lowercase Character" প্রিন্ট করবে, অঙ্ক বা ডিজিট হলে "Numerical Digit" প্রিন্ট করবে। অন্যথায় প্রিন্ট করবে "Special Character".

উদাহরণ

ইনপুট	আউটপুট
4	Lowercase Character
a	Uppercase Character
A	Numerical Digit
5	Special Character
;	

সমাধান

সমস্যাটি কম্পিউটার কিবোর্ডের বিভিন্ন চিহ্ন চেনার সমস্যা। যেমন: কম্পিউটারের কাছে 'A' এবং 'a' এ দুটি জিনিস আলাদা। 'A' হচ্ছে uppercase letter বা বড় হাতের বর্ণ এবং 'a' হচ্ছে lowercase letter যা ছোট হাতের বর্ণ। এছাড়াও আরও অন্যান্য চিহ্ন আছে যেমন: '(', ')', '*', '&' এগুলোকে আমরা বলে থাকি special character।

সমস্যাটির ইনপুটের বর্ণনা আগের অন্যান্য সমস্যার মতোই। T সংখ্যক লাইনে ক্যারেক্টার ch ইনপুট নিতে হবে। এখানে ক্যারেক্টার ইনপুট নিব আমরা এভাবে:

```
scanf("%s", &ch);
```

এভাবে ইনপুট নিলে আমরা একটিমাত্র ক্যারেক্টার ইনপুট নেওয়ার সময় কোনো সমস্যার সমূর্ধীন হব না। সাধারণত দেখা যায়, ক্যারেক্টার ইনপুট নেওয়ার সময় তোমরা একটি সমস্যা দেখতে পাও, যে কিবোর্ডের enter বাটনটি চাপ দিলে সেটি আরেকটা ক্যারেক্টারের ইনপুট নেয় এবং আউটপুটে ঝামেলা করা শুরু করে। এভাবে ইনপুট নিলে সেই ঝামেলাটা হবে না।

তোমরা বুবাতেই পারছ এই কোডটি কন্ডিশনাল লজিকের সাহায্যে করতে হবে। এখন নিচয়ই তোমরা কিবোর্ডের প্রতিটি ক্যারেক্টার নিয়ে কাজ করবে না। আমি এখানে কোডটি আসকি ভ্যালুর সাহায্য নিয়ে করেছি। আমাদের কোডটি হবে এমন:

```

#include<stdio.h>

int main ()
{
    char ch;
    int T;
    scanf("%d", &T);
    while(T--)
    {
        scanf("%s", &ch);

        if(ch >= 'a' && ch <= 'z')
        {
            printf("Lowercase character\n");
        }
        else if(ch >= 'A' && ch <= 'Z')
        {
            printf("Uppercase character\n");
        }
        else if(ch >= '0' && ch <= '9')
        {
            printf("Numerical Digit\n");
        }
        else
        {
            printf("Special character\n");
        }
    }
    return 0;
}

```

কোডটি লিখে অনলাইন জাজে জমা দাও। জমা দেওয়ার পর যদি Wrong Answer দেখায় তাহলে অবশ্যই কোডটিতে কোনো সমস্যা আছে। সেগুলো ঠিক করে আবার জমা দিয়ে দেখতে পারো।

সমস্যা ৩০ – যোগ্য সংখ্যা ১

সমস্যার বিবরণ

যোগ্য সংখ্যা বা Perfect Number হচ্ছে সে সব সংখ্যা, যেসব সংখ্যার নিজের চেয়ে ছোট ভাজকগুলোর যোগফল সংখ্যাটির সমান। যেমন: 6 একটি যোগ্য সংখ্যা, কারণ এটি $1+2+3=6$ । তোমার কাজ হচ্ছে এমন একটি প্রোগ্রাম লেখা যেটি একটি সংখ্যা পারফেক্ট কি না, সেটি নির্ণয় করবে।

(<http://bit.ly/1-52-problem-30>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 1000$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তীতে T সংখ্যক পূর্ণসংখ্যা N ($N \leq 2^{64}-1$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে N পারফেক্ট সংখ্যা হলে "Yes, N is a perfect number!" অন্যথায় "NO, N is not a perfect number!" প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	YES, 6 is a perfect number!
6 28 30	YES, 28 is a perfect number!
	NO, 30 is not a perfect number!

সমাধান

এই বইটির প্রথম দিকেই তোমরা ভাজক সম্পর্কে ধারণা পেয়েছ। সমস্যাটির বর্ণনায় তোমরা যোগ্য সংখ্যা কী, সেটি সম্পর্কে ধারণা পেয়েছ। এই সমস্যাটির সমাধানে কোনো একটি পূর্ণসংখ্যা যোগ্য সংখ্যা কি না, সেটি বের করতে হবে।

ইনপুটের ধারণা আগের সমস্যাগুলোর মতোই। টেস্ট কেস ইনপুট নিতে হবে, সেই ইনপুটের মান যত ততগুলো পূর্ণসংখ্যা ইনপুট নিতে হবে।

সমস্যাটির বর্ণনা পড়ে তোমরা কী বুঝতে পারলে? এখানে আমাদের ভাজক বের করার পর সেগুলোর যোগফল বের করতে হবে। প্রাণ্ট যোগফল যদি আমাদের ইনপুট সংখ্যাটির সমান হয় তাহলে সেটি একটি Perfect Number বা যোগ্য সংখ্যা।

প্রথমেই, আমরা সংখ্যাটি ইনপুট হিসেবে নিয়ে ফেলি।

```
scanf("%llu", &N);
```

এবার একটি for লুপের সাহায্যে সংখ্যা N-এর ভাজকগুলো আমরা বের করব। আমাদের লুপের গঠনটা এমন হবে:

```
unsigned long int sum = 0;
for(i = 1; i < N; i++)
{
    if(N%i == 0)
    {
        sum += i;
    }
}
```

এখানে আমরা ভাজকগুলোর যোগফল বের করেছি। এই যোগফল sum যদি আমাদের নেওয়া ইনপুট N-এর সমান হয় তাহলে আমরা N-কে যোগ্য সংখ্যা হিসেবে প্রিন্ট করব। একটি কন্ডিশনাল লজিক ব্যবহার করতে হবে। আর লুপটি কিন্তু আরও ছোট করে ফেলা যায়, $i < N$ কন্ডিশনের পরিবর্তে $i \leq \sqrt{N}$ ব্যবহার করে, যেখানে \sqrt{N} হচ্ছে N-এর বর্গমূল। আরও একটি বিষয় বলে দেই, লক্ষ করলে দেখবে এই সমস্যায় N এর সর্বোচ্চ মান হতে পারে, $2^{64}-1$, যেটা কিন্তু বেশ বড়সড় একটা সংখ্যা ($18,446,744,073,709,551,615$)। তাই অনেক উন্নত কম্পিউটারেও এই লুপটি চলতে অনেক সময় লাগবে। এমনকি যদি আমরা এর বর্গমূল (~ 4294967295) পর্যন্তও লুপ চালাই তাহলেও প্রোগ্রামটি চলতে প্রচুর সময় নেবে। এটাকে কীভাবে আরও দ্রুত চালানো যায় তোমরা একটু চিন্তা করে দেখো তো। পরের সমস্যায় আমরা এর একটা বুদ্ধিমান সমাধান দেখব।

এই কোডটিতে একটি ভুলও রয়েছে। এখানে N এর সর্বোচ্চ মান হতে পারে, $2^{64}-1$ । অর্থাৎ, `unsigned long int` টাইপের ডেটার সর্বোচ্চ সীমা যত, N-এর মান তত পর্যন্ত হতে পারে। এখন কিছু কিছু সংখ্যার ক্ষেত্রে, সংখ্যার ভাজকগুলোর যোগফল সংখ্যাটির চেয়ে বড়ও হতে পারে। যেমন ধরা যাক 12। 12-এর চেয়ে ছোট ভাজকগুলো হচ্ছে 1, 2, 3, 4 ও 6, যাদের যোগফল, $1 + 2 + 3 + 4 + 6 = 16$ যা কি না 12-এর থেকে বড়। এমন সব সংখ্যার ক্ষেত্রে যদি সংখ্যাটি যথেষ্ট বড় হয়, তবে তার ভাজকের যোগফল `unsigned long int` টাইপের ডেটার সীমায় রাখা সম্ভব হবে না। তাতে করে ডেটা ওভারফ্লো হবে এবং ক্ষেত্র বিশেষে ভুল উন্নত পাওয়ার সম্ভাবনা থাকে। সমস্যার সহজীকরনের স্বার্থে আমরা আপাতত ধরে নিচ্ছি এমন ওভারফ্লো-এর ক্ষেত্রে ভুল করে যোগফল প্রকৃত সংখ্যাটির সমান হয়ে যাবে না।

যদি কোনো প্রোগ্রামিং সমস্যার আউটপুটে কোনো কিছু প্রিন্ট করার কথা বলা দেওয়া থাকে, তাহলে বেশ ভালো করে প্রিন্ট statement-টি দেখে নেওয়া উচিত। অনেক সময় খুবই ছোট ভুলের জন্য তোমাদের কোডটি ভুল (রং অ্যানসার) হতে পারে।

সমস্যা ৩১ – যোগ্য সংখ্যা ২

সমস্যার বিবরণ

যোগ্য সংখ্যা বা Perfect Number হচ্ছে সে সব সংখ্যা, যেসব সংখ্যার ভাজকগুলোর যোগফল ওই সংখ্যার সমান। যেমন: 6 একটি যোগ্য সংখ্যা, কারণ এটি 1, 2, 3 দিয়ে বিভাজ্য এবং এই তিনটি সংখ্যার যোগফল 6 ($1+2+3=6$)। তোমার কাজ হচ্ছে এমন একটি প্রোগ্রাম লেখা যেটি একটি সংখ্যা পারফেক্ট কি না, সেটি প্রকাশ করবে।

(<http://bit.ly/1-52-problem-31>)

ইন্ডি

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তীতে T সংখ্যক পূর্ণসংখ্যা N ($N \leq 40000000$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে 1 থেকে N পর্যন্ত সবগুলো যোগ্য সংখ্যা বা পারফেক্ট নাম্বার প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2	6
100 500	28
	-
	6
	28
	496

সমাধান

তোমরা যদি যোগ্য সংখ্যা ১ এর সমাধান ঠিকভাবে করে থাকো, তাহলে এই সমস্যার সমাধানও সহজে করে ফেলতে পারবে। কেবল এখনে একটি লুপ বেশি হবে। কোডটি লিখে ফেলো। আচ্ছা, থাক। আমিই পুরো কোড লিখে দিই। তবে কোডের কোনো অংশ ব্যাখ্যা করব না। নতুনদের কোডের কিছু অংশ বুঝতে সমস্যা হতে পারে। নিজেরা চিন্তা করো।

```
#include<stdio.h>
#include<math.h>

int main()
{
    int T, i, N, num, sqrt_num, sum;
    scanf("%d", &T);

    while(T--)
    {
        scanf("%d", &N);

        for(num = 1; num <= N; num++)
        {
            sum = 1;
            sqrt_num = sqrt(num);

            for(i = 2; i <= sqrt_num; i++)
            {
                if (num % i == 0)
                {
                    sum = sum + i + num / i;
                }
            }
            if (sum == num)
            {
                printf("%d\n", num);
            }
        }
    }
    return 0;
}
```

প্রোগ্রাম চলতে কি অনেক বেশি সময় নিচ্ছে? তুমি যদি 1 থেকে 40000000-এই সীমার মধ্যে যোগ্য সংখ্যা খুঁজে বের করার কাজটি ঠিকঠাকভাবে করতে পারো, তুমি দেখবে যে তুমি মাত্র

পাঁচটি সংখ্যা খুঁজে পেয়েছ, যাদেরকে যোগ্য সংখ্যা বলা যাবে। সংখ্যাগুলো হচ্ছে 6, 28, 496, 8128, 33550336।

এখন তোমাদের একটি সহজ বুদ্ধি শিখিয়ে দিই। হয়তো এই বুদ্ধি কখনো কাজে লেগে যেতেও পারে। আমরা একটা প্রোগ্রাম রান করে সংখ্যাগুলো বের করে ফেলার পরে আরেকটি প্রোগ্রাম লিখব, যেখানে আমরা আমাদের জানা সংখ্যাগুলো ব্যবহার করব।

```
int ara[] = {6, 28, 496, 8128, 33550336};
```

তারপরে কেবল একটি লুপ চালিয়ে দেখব যে নির্দিষ্ট সীমার মধ্যে কোন কোন সংখ্যা আছে :

```
for(i = 0; i < 5; i++)
{
    if (ara[i] <= n)
    {
        printf("%d\n");
    }
    else
    {
        break;
    }
}
```

তোমরা এখন পুরো প্রোগ্রামটি লিখে ফেলো আর অনলাইন জাজে জমা দাও।

সমস্যা ৩২ – X-এর গুণিতক

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে, যেটি একটি সংখ্যা N পর্যন্ত একটি সংখ্যা X-এর গুণিতকসমূহ প্রিন্ট করবে।

(<http://bit.ly/1-52-problem-32>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে দুটি করে পূর্ণসংখ্যা X এবং N ($1 \leq X \leq N \leq 1000000$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে X থেকে N পর্যন্ত X এর গুণিতকসমূহ প্রিন্ট করতে হবে। যদি X এর মান N এর থেকে বড় হয় তাহলে "Invalid" প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	2
2 10	4
99 1000	6
10 5	8
	10
	99
	198
	297
	396
	495
	594
	693
	792
	891
	990
	Invalid!

সমাধান

সমস্যাটির বর্ণনায় বলা আছে ইনপুট হবে একটি পূর্ণসংখ্যা T, যেটি আমাদের টেস্ট কেসের সংখ্যা। তাহলে আমাদের প্রথম কাজ হবে ইনপুট নেওয়া। আমরা T এর জন্য ডেটা টাইপ নিব int।

```
int T;
```

এরপর T সংখ্যক লাইনে ইনপুট নিতে হবে আরও দুটি পূর্ণসংখ্যা X এবং N। খেয়াল করে দেখো এখানে N ভ্যারিয়েবলটির জন্য একটি রেঞ্জ দেওয়া আছে। সেই রেঞ্জ এর সংখ্যা বিবেচনায় রেখে আমরা X এবং N এর জন্য int ডেটা টাইপ নিব।

```
int X, N;
```

প্রথমে আমরা একটা লুপ চালাবো যেটি T এর মান যত ততবার চলবে:

```
for(i = 0; i < T; i++) {}
```

লুপের মধ্যে প্রতিবার ইনপুট নিতে হবে X এবং N-এর মান:

```
scanf("%d %d", &X, &N);
```

এবার মূল কোডটি লেখার জন্য প্রথমে আমরা বিবেচনায় আনব Invalid এর কন্ডিশনটি। যদি Invalid কন্ডিশনটি সত্যি না হয় তখন আমরা 1 থেকে N পর্যন্ত একটি লুপ চালাব এবং লুপের ভেতরে 1 থেকে N পর্যন্ত X এর গুণিতক বের করার কাজটি করব। X এর গুণিতক হবে 1 থেকে N পর্যন্ত সেই সব সংখ্যা যা X দ্বারা নিঃশেষে বিভাজ্য। কোডটি হতে পারে এমন:

```
for(j = 1; j <= N; j++)
{
    if(j%X == 0)
    {
        printf("%d\n", j);
    }
}
```

এবারে প্রোগ্রামটি লিখে চালিয়ে দেখো ঠিকমতো ফলাফল দিচ্ছে কি না। যদি ঠিকঠাক লিখে ফেলতে পারো তবে এখন চলো আমরা একটু চিন্তা করে দেখি এই সমাধানটি অপটিমাল হলো কি না। আমরা এখানে 1 থেকে শুরু করে N পর্যন্ত একটা লুপ চালিয়েছি। এখানে N এর মান সর্বোচ্চ হতে পারে 1000000। অর্থাৎ, আমাদের এই লুপটি চলবে সর্বমোট 1000000 বার।

প্রতি বারে আমাদের হিসাব করে দেখতে হচ্ছে সংখ্যাটি X এর গুণিতক কি না। মনে করো যদি আমরা j এর মান 1 এর বদলে X দিয়ে শুরু করি এবং প্রতি ধাপে এক এক করে না বাড়িয়ে একেবারে X এর সমান করে বাড়াই, তাহলে আমাদের অনেকগুলো ইটারেশন কর্মে যাবে। আর সেই সঙ্গে আমাদের আর তখন i f ব্যবহার করে পরীক্ষা করারও দরকার পড়বে না। কেননা, আমরা X এর মানের সমপরিমাণ করে j এর মান বৃদ্ধি করছি, তাই j এর মান সবসময় X এর গুণিতকই থাকবে।

```
for(j = X; j <= N; j += X)
{
    printf("%d\n", j);
}
```

এভাবে করে আমরা অনেক ক্ষেত্রেই কোডের রানটাইম কমিয়ে ফেলতে পারি। দুটি সমাধানই একই কাজ করছে, কিন্তু পরের সমাধানটি আগেরটার চেয়ে দ্রুত চলছে। এই যে, বৃদ্ধি খাটিয়ে প্রোগ্রামের রানটাইম কমিয়ে ফেললাম, এই কাজটাকে আমরা বলি অপটিমাইজেশন। পরের কোডটি আগের কোডটির চেয়ে অপটিমাল।

এখন তোমরা কোডটি গুছিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো। তারপর এই সমস্যার লিঙ্কে গিয়ে সাবমিট করে দেখো।

সমস্যা ৩৩ – বিভাজনসাধ্য ১

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি নির্দিষ্ট সীমার অন্তর্ভুক্ত যতগুলো সংখ্যা অপর একটি পূর্ণসংখ্যা দ্বারা নিঃশেষে বিভাজ্য সেটি বের করবে।
(<http://bit.ly/1-52-problem-33>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে তিনটি পূর্ণসংখ্যা A, B, C ($1 \leq A, B, C \leq 10^{16}$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে A থেকে B পর্যন্ত যতগুলো সংখ্যা C দ্বারা নিঃশেষে বিভাজ্য সেই সংখ্যাগুলো প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	3
2 20 3	6
50 60 5	9
55 100 6	12
.	15
	18
	50
	55
	60
	60
	66
	72
	78
	84
	90
	96

সমাধান

খুবই সহজ একটি সমস্যা। তিনটি পূর্ণসংখ্যা A, B, C দেওয়া থাকবে। C দিয়ে A থেকে B পর্যন্ত যতগুলো সংখ্যা নিঃশেষে বিভাজ্য সেগুলো বের করতে হবে।

তোমরা নিচয়ই বুঝতে পারছ যে, এখানে A থেকে B পর্যন্ত যাওয়ার জন্য আমাদের একটি লুপের দরকার হবে এবং আমরা লুপটি চালানোর সময় প্রতিবার A থেকে B পর্যন্ত সংখ্যাগুলোকে C দিয়ে ভাগ করতে থাকব।

ইনপুটের বর্ণনায় আগের মতোই বলা আছে T সংখ্যক লাইনে ইনপুট নেওয়ার কথা। ইনপুট হবে তিনটি পূর্ণসংখ্যা A, B, C। মনে রাখবে, ভ্যারিয়েবলের নাম যথাযথ হওয়া বাঞ্ছনীয়। ধরো, এখানে আমি A, B, C না দিয়ে ইচ্ছা মতো নিজের বা আমার বন্ধুদের নাম ব্যবহার করলাম (অনেকে কিন্তু তোমরা এমনটি করো)। আপাতদৃষ্টিতে কোনো সমস্যা না হলেও ইচ্ছামত নাম দিলে আরেকটি পরে যখন বেশ বড় কোড করবে তখন বুঝতে নিজেরই কষ্ট হবে, কোনটা দিয়ে কী করা হয়েছিল।

A থেকে B পর্যন্ত সব সংখ্যাগুলো পাওয়ার জন্য আমরা যে লুপ চালাব সেটি এমন:

```
for(i = A; i <= B; i++) {}
```

আমার কাজ শেষ। এখন A এবং B-এর মধ্যকার সংখ্যাগুলোকে C দিয়ে যে সংখ্যাগুলোকে নিঃশেষে ভাগ করা যায় সেগুলো প্রিন্ট করার কাজ তোমাদের। সমস্যাটি সমাধান করার পরে আবার চিন্তা করে দেখো A, B, C এর সর্বোচ্চ মান কত হতে পারে এবং তোমার লুপটি কতবার চলবে। "x-এর গুণিতক" সমস্যায় আমরা যে অপটিমাইজেশন টেকনিক ব্যবহার করেছি, সেটা কি কোনোভাবে এই সমস্যাতেও ব্যবহার করা যায়?

সমস্যা ৩৪ – বিভাজনসাধ্য ২

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি 1 থেকে শুরু করে একটি নির্দিষ্ট সীমার অন্তর্ভুক্ত যতগুলো সংখ্যা দুটি পূর্ণসংখ্যা দিয়ে নিঃশেষে বিভাজ্য সেটি বের করবে।

(<http://bit.ly/1-52-problem-34>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে তিনটি পূর্ণসংখ্যা A, B, C ($1 \leq A, B \leq 10^9$ এবং $C \leq 10^{16}$)

আউটপুট

প্রোগ্রামটির আউটপুটে 1 থেকে C পর্যন্ত যতগুলো সংখ্যা A এবং B দিয়ে নিঃশেষে বিভাজ্য সেই সংখ্যাগুলো প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	6
2 3 50	12
3 5 50	18
5 6 100	24
.	30
	36
	42
	48
	15
	30
	45
	30
	60
	90

সমাধান

সমস্যার বর্ণনায় বলা আছে তিনটি পূর্ণসংখ্যা A, B, C দেওয়া থাকবে। 1 থেকে C পর্যন্ত যতগুলো সংখ্যা A এবং B দিয়ে নিঃশেষে বিভাজ্য সেগুলো আউটপুটে দেখাতে হবে।

ইনপুটের বর্ণনায় বলা আছে T সংখ্যক লাইনে ইনপুট নেওয়ার কথা। প্রথমে একটি ভ্যারিয়েবল নিতে হবে T, যেটির সাহায্যে আমরা একটি লুপ চালাব এবং T সংখ্যক লাইনে ইনপুট নিব। টেস্ট কেসের সংখ্যা যত অর্থাৎ T-এর মান যত হবে ঠিক ততটি লাইনে আমরা A, B, C ইনপুট নিব।

```
scanf("%d", &T);
```

এখানে আমরা for লুপের পরিবর্তে while লুপ দিয়েও কাজ করতে পারি। আমি এখানে while লুপ ব্যবহার করেছি। যেখানে T-এর মান এক এক করে কমেছে। লুপের ভেতরে আমরা A, B, C-এর মান নিয়েছি। C এর মান বিবেচনায় আমরা এখানে long int ব্যবহার করব:

```
while(T--)
{
    scanf("%ld %ld %ld", &A, &B, &C);
}
```

এবার কোডের মূল কাজটি করার পালা। আমাদের এখন আরও একটি লুপ চালাতে হবে যেটি 1 থেকে C পর্যন্ত চলবে এবং এই সীমায় অবস্থিত যতগুলো সংখ্যা A এবং B দিয়ে নিঃশেষে বিভাজ্য সেটি প্রিন্ট করে দেখাবে। কোডটি হতে পারে এমন:

```
for(i = 1; i <= C; i++)
{
    if(i%A == 0 || i%B == 0)
    {
        printf("%ld\n", i);
    }
}
```

এখন চলো দেখি এই সমাধানটিকে আমরা অপটিমাইজেশন করতে পারি কি না। অবশ্যই তোমরা নিজেরা আগে চেষ্টা করে দেখবে কোনোভাবে অপটিমাইজেশন করতে পারো কি না। তারপরে আমার সমাধানটি পড়ে দেখবে। আমরা এর আগে এ রকম আরও দুটি সমস্যার সমাধান করেছি: "x-এর গুণিতক" ও "বিভাজনসাধ্য-১"। দুটি সমস্যাতেই ছিল একটি নির্দিষ্ট সীমার মধ্যে একটি সংখ্যার গুণিতকের তালিকা খুঁজে বের করা। তাই, আমরা খুব সহজেই ব্যবহৃত লুপটি এক এক করে না বাড়িয়ে একেবারে ওই সংখ্যার সমান হারে বাড়িয়ে লুপের ইটারেশন সংখ্যা অনেক

কমিয়ে ফেলতে পেরেছিলাম। কিন্তু এবারে সমস্যা হচ্ছে একেবারে দুটি সংখ্যারই গুণিতক এমন সব সংখ্যা খুঁজে বের করা। তাহলে আমরা এখন লুপ্টি করে ইনক্রিমেন্ট করব? চিন্তা করে দেখো আমরা আসলে বের করতে চাচ্ছি, এমন সব সংখ্যা যারা দুটি সংখ্যারই গুণিতক বা যাকে গণিতের ভাষায় বলে সাধারণ গুণিতক। তাহলে আমরা যদি সংখ্যা দুটির লিখিষ্ট সাধারণ গুণিতক বা L.S.G. বের করে ফেলি চট করে আর লুপ্টিকে L.S.G. এর সমান হারে ইনক্রিমেন্ট করতে থাকি তাহলেও কিন্তু আমরা একই সমাধান পাব। আর এই ক্ষেত্রে সমাধানটি আগের সমাধানের চেয়ে অনেক দ্রুত চলবে। এবার তোমাদের কাজ সম্পূর্ণ কোডটি ঠিকঠাক মতো লেখা এবং যদি কোডে কোনো ভুল থাকে সেটি খুঁজে বের করে ঠিক করা।

সমস্যা ৩৫ – বৃত্তের বাইরে

সমস্যার বিবরণ

একটি স্থানাঙ্ক বৃত্তের বাইরে না ভেতরে অবস্থিত, সেটি বের করার প্রোগ্রাম লিখতে হবে।
[\(http://bit.ly/1-52-problem-35\)](http://bit.ly/1-52-problem-35)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক ইনপুট থাকবে। প্রতিটি ইনপুটে প্রথম লাইনে থাকবে একজোড়া ধনাত্মক বা ঋণাত্মক সংখ্যা যেটি বৃত্তের কেন্দ্রের স্থানাঙ্ক (X_c, Y_c) , এরপর ইনপুট হবে বৃত্তের ব্যাসার্ধ r । পরে ইনপুট নিতে হবে একটি স্থানাঙ্ক (X, Y) ($|X|, |Y|, |X_c|, |Y_c| < 100000$)।

আউটপুট

প্রোগ্রামটির আউটপুটে (X, Y) বৃত্তের অভ্যন্তরে থাকলে প্রিন্ট করতে হবে "The point is inside the circle" অন্যথায় প্রিন্ট করতে হবে "The point is not inside the circle"।

উদাহরণ

ইনপুট	আউটপুট
2	The point is not inside the circle
1 1	The point is inside the circle
4	
10 -14	
1 1	
8	
5 6	

সমাধান

কোনো বিন্দু একটি বৃত্তের বাইরে না ভেতরে আছে, সেটি জানার জন্যে প্রথমে বৃত্তের কেন্দ্র থেকে ওই বিন্দুর দূরত্ব বের করতে হবে। সেটি যদি বৃত্তটির ব্যাসার্ধের চেয়ে বেশি হয়, তাহলে বিন্দুটি বৃত্তের বাইরে, আর না হলে বৃত্তের ভেতরে।

দুটি বিন্দুর মধ্যে দূরত্ব বের করবে কী করে? বিন্দু দুটির স্থানাঙ্ক জানা থাকলে একটি সূত্রের সাহায্যেই সেটি বের করা যায়। একটি বিন্দুর স্থানাঙ্ক (x_1, y_1) ও অপর বিন্দুর স্থানাঙ্ক (x_2, y_2) জানা থাকলে তাদের মধ্যবর্তী দূরত্ব হচ্ছে :

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

সূত্রের ব্যাখ্যা জানার জন্য তোমরা হাইস্কুল বা কলেজের জ্যামিতি বইয়ের শরণাপন্ন হতে পারো।

এখন তোমাদের কেবল জানতে হবে যে `math.h` হেডার ফাইলে `sqrt()` নামক একটি ফাংশন আছে, যেটি দিয়ে বর্গমূল বের করার কাজটি করা যায়। বাকি কাজ খুব সহজ।

সমস্যা ৩৬ – শব্দ সাজানো

সমস্যার বিবরণ

N-সংখ্যক শব্দকে ইংরেজি আক্ষরিক ত্রুমানুসারে (Alphabetically) সাজাতে হবে।
(<http://bit.ly/1-52-problem-36>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তীতে T সংখ্যক টেস্ট কেস থাকবে। প্রতিটি টেস্ট কেসের প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা N ($N \leq 20$), পরবর্তী N সংখ্যক লাইনে ইনপুট হবে একটি স্ট্রিং S (S এর দৈর্ঘ্য 10000 এর বেশী নয়)।

আউটপুট

প্রোগ্রামটির আউটপুটে N-সংখ্যক স্ট্রিং S কে আক্ষরিক ত্রুমানুসারে প্রিন্ট করতে হবে। আউটপুটের শুরুতে এবং শেষে একটি '\n' প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
1	apple
5	bat
x-ray	cat
apple	house
cat	x-ray
bat	
house	

সমাধান

শব্দ সাজানো সমস্যাটি স্ট্রিং কম্পেয়ার বা তুলনার একটি সমস্যা। এখানে আমাদের যে কাজটি করতে হবে সেটি হচ্ছে N সংখ্যক ইনপুট স্ট্রিংগুলোকে আক্ষরিক ত্রুমানুসারে সাজাতে হবে। অর্থাৎ, সর্ট করতে হবে। সমস্যাটির সমাধান করার জন্য আমরা এখানে N-সংখ্যক স্ট্রিংকে পরস্পরের সঙ্গে তুলনা করতে পারি। আমি এখানে বাবল সর্ট অ্যালগরিদমের সাহায্য নিয়েছি। সর্টিং অ্যালগরিদমটি হবে এমন:

```

for(i = 0; i < N; i++)
{
    for(j = i; j < N; j++)
    {
        if(strcmp(name[i], name[j]) > 0)
        {
            -
            strcpy(temp, name[i]);
            strcpy(name[i], name[j]);
            strcpy(name[j], temp);
        }
    }
}

```

এখানে N হচ্ছে স্ট্রিংয়ের সংখ্যা। আমরা `string.h` হেডার ফাইলের দুটি ফাংশন `strcmp()` এবং `strcpy()` এর সাহায্য নিয়েছি। `strcmp()` দিয়ে দুটি স্ট্রিংগুলোকে কম্পেয়ার করা হচ্ছে এবং `strcpy()` দিয়ে আরেকটি স্ট্রিং ভ্যারিয়েবল `temp`-এ আমরা স্ট্রিং কপি করেছি। সাধারণত, আমরা দুটি সংখ্যা তুলনা করতে গেলে যেটা করে থাকি $a>b$, স্ট্রিংয়ের ক্ষেত্রে তুলনা করার সময় `strcmp()` ফাংশন ব্যবহার করে থাকি।

উপরের কোডটিকু করার পরে, স্ট্রিংগুলোকে প্রিন্ট করলে সেগুলো আক্ষরিক ক্রমানুসারে প্রিন্ট হবে। আউটপুটে দুটি নিউলাইন থাকবে। একটি ইনপুট শুরু হওয়ার আগে এবং একটি ইনপুট শেষ হওয়ার পরে।

এই কোডটিকেও কিন্তু আমরা অপটিমাইজেশন করতে পারি। আমরা এখানে ছোট থেকে বড় আকারে সাজানোর জন্য বাবলসর্ট অ্যালগরিদম ব্যবহার করেছি। যদি আমরা আরও দ্রুত কোনো অ্যালগরিদম ব্যবহার করি তাহলে কোডটিও দ্রুত চলবে। তোমরা যখন বিভিন্ন রকম অ্যালগরিদম শিখবে তখন সেই অ্যালগরিদমগুলোর মধ্যে যেটি এখানে সবচেয়ে উপযোগী সেটিই ব্যবহার করবে।

সমস্যা ৩৭ – সংখ্যা বিপর্যয়

সমস্যার বিবরণ

এমন একটি প্রোগ্রাম লিখতে হবে যেটি একটি সংখ্যাকে উল্টোভাবে দেখাবে।
[\(<http://bit.ly/1-52-problem-37>\)](http://bit.ly/1-52-problem-37)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে একটি করে পূর্ণসংখ্যা N ($N \leq 100000000$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে N কে উল্টোভাবে প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	216
612	6501341
1431056	29914001
10041992	

সমাধান

সমস্যাটির বর্ণনায় বলা আছে একটি সংখ্যাকে উল্টো করে প্রিন্ট করতে হবে। এর মানে হচ্ছে যদি একটি সংখ্যা হয় 123 তাহলে আউটপুটে আমাদের দেখাতে হবে 321।

উদাহরণস্বরূপ: ধরা যাক, আমরা যে সংখ্যাটিকে উল্টো করে প্রিন্ট করতে চাইছি সেটি 123, অর্থাৎ $N = 123$ । যদি আমরা 123-কে আমরা যদি 10 দ্বারা ভাগ করি তাহলে ভাগশেষ পাব 3 এবং ভাগফল পাব 12। ভাগশেষ 3 হচ্ছে আমাদের নেওয়া সংখ্যাটির শতকের ঘরের অঙ্ক এবং সংখ্যাটি উল্টো করলে আমরা এই অঙ্কটি প্রথমে পাব।

ইনপুটের বর্ণনায় বলা আছে যে, T -সংখ্যক লাইনে ইনপুট নিতে হবে একটি করে পূর্ণসংখ্যা। অর্থাৎ, T -এর মান যত ততগুলো সংখ্যা আমাদের ইনপুট নিতে হবে। যে সংখ্যাটি আমরা ইনপুট নিব সেটিকে উল্টোভাবে প্রিন্ট করতে হবে।

এবার একটা হিসাব করা যাক এই 123 সংখ্যাটি নিয়ে।

আমরা দুটি ভেরিয়েবল নেই rev (reverse-এর জন্য) এবং N = 123। এই হিসাবে তিনটি ধাপ থাকবে।

- প্রথমে rev এর কোনো ভ্যালু থাকবে না, কারণ আমরা সংখ্যাটি উল্টে দেইনি। তাহলে rev = 0।
- $123 \% 10 = 3$, 123-কে 10 দিয়ে ভাগ করলে ভাগশেষ থাকবে 3। rev এর ভ্যালু 0 বলে rev = 3 হবে এখন।
- এবার N-এর ভ্যালু আপডেট করতে হবে। $N = N/10$; এখন N-এর ভ্যালু হবে $N = 123/10 = 12$

এই কাজটি আমরা যদি N-এর মান শূন্য না হওয়া পর্যন্ত বারবার করি তাহলে আমরা একসময় আমরা আমাদের কাজিক্ষিত আউটপুট পেয়ে যাব। এর জন্য আমাদের কোডটি হবে এমন:

```
while(num != 0)
{
    rev = rev * 10;
    rev = rev + N%10;
    N = N / 10;
}
```

এই কোডটির ইটারেশন হবে এমন:

ধাপ	num != 0	rev	N
0		0	123
1	true	$rev * 10 = 0$	$N / 10$ $= 123 / 10$ $= 12$
		$rev + N \% 10$ $= 0 + 123 \% 10$ $= 0 + 3 = 3$	
2	true	$rev * 10$ $= 3 * 10 = 30$	$N / 10$ $= 12 / 10$ $= 1$
		$rev + N \% 10$ $= 30 + 12 \% 10$ $= 30 + 2 = 32$	
3	true	$rev * 10$ $= 320$	$N / 10$ $= 1 / 10$ $= 0$
		$rev + N \% 10$ $= 320 + 1 \% 10$ $= 320 + 1 = 321$	
4	false	stop	stop

আমরা প্রথম ধাপে 123 উল্টো করে পাওয়া প্রথম অঙ্ক 3 পেয়ে গেলাম।

আমরা দ্বিতীয় ধাপে পেলাম $rev = 32$ এবং $N = 1$.

আমরা তৃতীয় ধাপে পেলাম $rev = 321$ এবং $N = 0$.

তোমরা কি খেয়াল করেছ যে আমরা rev -এর মানে আমাদের কাঙ্ক্ষিত আউটপুট পেয়ে গেছি এবং N এর মান শূন্য হয়ে গিয়েছে।

এখন তোমাদের কাজ হচ্ছে আউটপুট প্রিন্ট করে পুরো কোড লিখে অনলাইন জাজে জমা দেওয়া।

এবারে তোমরা সমস্যাটি আরেকভাবে সমাধান করার চেষ্টা করতে পারো। সংখ্যাকে স্ট্রিংয়ে রূপান্তর করে স্ট্রিংটি উল্টে দেওয়া।

সমস্যা ৩৮ – হীরক রাজ্য

সমস্যার বিবরণ

হীরক আকৃতি (Diamond shape) প্রিন্ট করার একটি প্রোগ্রাম লিখতে হবে।
[\(<http://bit.ly/1-52-problem-38>\)](http://bit.ly/1-52-problem-38)

ইনপুট

প্রোগ্রামটির প্রথমে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) ইনপুট নিতে হবে এবং পরবর্তী T সংখ্যক লাইনে দুটি পূর্ণসংখ্যা n ও m ($1 \leq n \leq 40, 1 \leq m \leq 9$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে $2n-1$ সংখ্যক লাইন m উপাদান দিয়ে হীরক আকৃতি প্রিন্ট করতে হবে।
প্রতিটি হীরক আকৃতির শেষে একটি নিউলাইন প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2	1
3 1	1 1
5 2	1 1 1 1 1 1
	2
	2 2
	2 2 2
	2 2 2 2
	2 2 2 2
	2 2 2
	2 2
	2

সমাধান

এই সমস্যাটি হঠাতে দেখে একটু কঠিন লাগতে পারে। কিন্তু আসলে এটি বেশ সহজ একটি সমস্যা।
তোমার লুপের ব্যবহারে দক্ষ হতে হবে। নেস্টেড লুপ দিয়ে সমস্যাটি সহজেই সমাধান করা যায়।

আমার পরামর্শ হবে, তুমি এই সমস্যার সমাধান করতে গিয়ে ঝামেলায় পড়লেও লেগে থাকো।
সমাধান হবেই।

একটি হিন্টস্ দিয়ে দিই। ধরা যাক, তোমাকে নিচের মতো একটি আকৃতি প্রিন্ট করতে হবে :

1
11
111
11
1

এটি তুমি দুই ধাপে করতে পারো। প্রথম ধাপে নিচের অংশটুকু প্রিন্ট করবে :

1
11
111

পরের ধাপে বাকি অংশটুকু প্রিন্ট করবে :

11
1

বাকি কাজ আশা করি তোমরা করে ফেলতে পারবে। এটির সমাধান করতে গিয়ে নেস্টেড লুপ ছাড়াও তোমরা আরেকটি জিনিস লক্ষ করো। আমরা একটি হীরক আকৃতি প্রিন্ট করার জন্য দুই ধাপে কাজ করেছি। মাঝে মধ্যে কোনো কোনো সমস্যাকে আমরা যদি ছোট ছোট ভাগে ভেঙে ফেলতে পারি, তাহলে সমাধান অনেক সহজ হয়ে যায়।

সমস্যা ৩৯ – প্যালিনড্রোম

সমস্যার বিবরণ

প্যালিনড্রোম হুচে সেসব শব্দ যেটি সোজা বা উল্টো যেভাবে করেই পড়া হোক না কেন একই থাকে। যেমন: রমাকান্তকামার, এই শব্দটি ডানদিক বা বামদিক যেকোনো দিক থেকেই পড়া হোক না কেন একই থাকবে। তাই "রমাকান্তকামার" প্যালিনড্রোম। এমন আরেকটি শব্দ হচে WOW. (<http://bit.ly/1-52-problem-39>)

তোমার কাজ হচ্ছে, এমন একটি প্রোগ্রাম লেখা যেটি একটি শব্দ প্যালিনড্রোম কি না সেটি শনাক্ত করবে।

ইনপুট

প্রোগ্রামটির প্রথমে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) ইনপুট নিতে হবে এবং পরবর্তী T সংখ্যক লাইনে ইনপুট হবে একটি করে স্ট্রিং S (S এর দৈর্ঘ্য 1000 এর বেশী নয়)।

আউটপুট

প্রোগ্রামটির আউটপুটে ইনপুট স্ট্রিংটি প্যালিনড্রোম কি না সেটি দেখাতে হবে। যদি প্যালিনড্রোম হয় তাহলে প্রিন্ট করতে হবে "Yes! It is Palindrome!" অন্যথায় প্রিন্ট করতে হবে "Sorry! It is not Palindrome!"।

উদাহরণ

ইনপুট	আউটপুট
3	Yes! It is palindrome!
wow	Sorry! It is not palindrome!
string	Yes! It is palindrome!
civic	

সমাধান

তোমরা যদি সি প্রোগ্রামিং ভাষার কোনো বই পড়ে থাকো, তোমরা নিশ্চয়ই জানো কীভাবে একটি স্ট্রিংকে উল্টো দিতে হয়। মানে স্ট্রিংটি যদি হয় abcd, উল্টো স্ট্রিংটি হবে dcba। এজন্য তোমাকে যেই ক্যারেন্টার অ্যারেটি দেওয়া হবে, একই দৈর্ঘ্যের আরেকটি অ্যারে নিতে হবে। তারপর একটি লুপ ব্যবহার করে প্রথম অ্যারের শেষ উপাদানটি দ্বিতীয় অ্যারের প্রথম উপাদানে অ্যাসাইন করবে, প্রথম অ্যারের শেষ উপাদানের আগের উপাদানটি দ্বিতীয় অ্যারের দ্বিতীয় উপাদানে অ্যাসাইন

করবে, এ রকম করে চলবে। তাহলে এই সমস্যার সমাধান করতে গেলে তোমার কোড কী রকম হবে?

```
int main()
{
    char string1[1001], string2[1001];
    int T;

    scanf("%d", &T);
    while(T--)
    {
        scanf("%s", string1);

        // now reverse the string into string2
        if (strcmp(string1, string2) ==0)
        {
            printf("Yes! It is Palindrome!\n");
        }
        else
        {
            printf("Sorry! It is not Palindrome\n");
        }
    }
    return 0;
}
```

তোমরা কোডটি সম্পূর্ণ করে তারপরে অনলাইন জাজে জমা দিয়ে দেখো। আশা করি উভয় সঠিক হবে।

- এখন তোমরা চাইলে একটি অ্যারে দিয়েও কাজটি করতে পারো। ধরা যাক, যে স্ট্রিংটি তোমরা প্যালিনড্রোম কি না তা পরীক্ষা করবে, সেটিতে দশটি ক্যারেন্টার রয়েছে। একটি লুপ চালাবে যেটি প্রথম ক্যারেন্টারের সঙ্গে দশম ক্যারেন্টার, দ্বিতীয় ক্যারেন্টারের সঙ্গে নবম ক্যারেন্টার, তৃতীয় ক্যারেন্টারের সঙ্গে অষ্টম ক্যারেন্টার, এভাবে তুলনা করবে এবং যদি কোনো জায়গায় অমিল পাওয়া যায়, তাহলে সরাসরি বলে দেবে যে এটি প্যালিনড্রোম নয়। আর যদি সবকিছুই মিলে যায়, তবে তা প্যালিনড্রোম। এখন তোমরা চটপট প্রোগ্রাম লিখে ফেলো। নতুনদের ঘণ্টাখানেকের মতো (বা আরোও বেশি) সময় লাগতে পারে, তাতে হতাশ হয়ে না, এটিই স্বাভাবিক।

সমস্যা ৪০ – ধারার যোগফল ১

সমস্যার বিবরণ

$x^0 + x^1 + x^2 + x^3 + x^4 + \dots + x^k$ ধারার ফলাফল বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-40>)

ইনপুট

প্রোগ্রামটির প্রথমে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$) ইনপুট নিতে হবে এবং পরবর্তী T সংখ্যক লাইনে দুটি পূর্ণসংখ্যা X, K ($X \leq 10$ এবং $K \leq 6$) নেওয়া হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে উপরোক্ত সিরিজের ফলাফল প্রিন্ট করতে হবে। এখানে " $=$ " এর আগে ও পরের স্পেসটি লক্ষ্যীয়।

উদাহরণ

ইনপুট	আউটপুট
2	Result = 31
5 2	Result = 63
2 5	
3	Result = 2047
2 10	Result = 12207031
5 10	Result = 111111
10 5	

সমাধান

এটি একটি যোগফল নির্ণয়ের ধারা যেখানে দুটি ভেরিয়েবল X, K রয়েছে। এখানে X এর power বা শক্তি K। সমস্যাটির বর্ণনায় প্রথম লাইনে একটি পূর্ণসংখ্যা নেওয়ার কথা বলা হয়েছে। এখানে আমরা পূর্ণসংখ্যা হিসেবে নিব T। পরবর্তী T সংখ্যক লাইনে আরও দুটি পূর্ণসংখ্যা ইনপুট নিতে হবে X, K। যেহেতু পূর্ণসংখ্যা নিয়েই কাজ এবং বলা আছে K এর সর্বোচ্চ মান 6, তাই এখানে আমাদের ডেটা টাইপ হবে int।

```
int X, K, T;
```

প্রথমেই আমরা T সংখ্যক লাইনে ইনপুট নেওয়ার কাজটি করে ফেলি। এর জন্য আমরা scanf() ফাংশন ব্যবহার করে T এর জন্য ইনপুট নিব এবং T সংখ্যক লাইনে ইনপুট নেওয়ার জন্য T পর্যন্ত একটি for লুপ ব্যবহার করব।

```
scanf("%d", &T);
for(int i = 0; i < T; i++) {}
```

এবার আমাদের এই লুপটির ভেতরে X এবং K এর ইনপুট নেওয়ার কাজটি করতে হবে। এখানেও আমরা scanf() ফাংশন ব্যবহার করে ইনপুট নেওয়ার কাজটি করব।

```
scanf("%d %d", &X, &K);
```

এবার আসি আমাদের মূল হিসাবনিকাশের কাজে। যেহেতু একটি ধারায় একটি কাজ বারবার হতে থাকে তাই আমরা এখানে অবশ্যই লুপের সাহায্য নিব। ধারাটি খেয়াল করলে আমরা দেখতে পাই, এখানে X এর পাওয়ার এক এক করে বেড়ে K পর্যন্ত পৌছেছে। তাহলে এখানে আমাদের একটি for লুপ চালাতে হবে যেটি K পর্যন্ত চলবে। লক্ষ করো:

$$x^{k-1} \times x = x^k$$

আমরা যদি লুপ চালানোর সময় X^{k-1} এর মান অন্য একটি ভ্যারিয়েবলে রেখে দেই, তাহলে প্রতিবার তার সঙ্গে X এর মান গুণ করলেই আমরা ধারার পরের পদটি পেয়ে যাব।

```
int sum = 1, power = 1;
for(int i = 1; i <= K; i++)
{
    power = power * X;
    sum += power;
}
```

এই কোডটি কি ঠিকমতো আউটপুট দিবে? না দিলে এর কোথায় সমস্যা আছে ঝুঁজে বের করো তো। লক্ষ করো, আমি কিন্তু, sum ও power এর মান 0 না দিয়ে শুরুতেই 1 দিয়েছি, আর লুপটি চালানোর সময়ও আমি $i = 0;$ না করে $i = 1;$ করেছি।

আরেকটি বিষয় বলে দেই তোমাদের, তোমরা যারা ইতিমধ্যে বিভিন্ন গাণিতিক সমস্যার সমাধান করেছ তারা নিশ্চয়ই pow() ফাংশনের সঙ্গে পরিচিত। pow() ফাংশনটি দুটি double টাইপের ডেটা ইনপুট নেয় ও একটি double টাইপ ডেটা রিটার্ন করে। X^K -এর মান বের করার জন্য অনেকে হয়তো এই ফাংশন ব্যবহার করতে চাইবে। তবে এখানে এই ফাংশনটি ব্যবহার করা ঠিক হবে না। সাধারণত, যেসব সমস্যার সমাধান পূর্ণসংখ্যা, সেগুলোর সমাধানে পারতপক্ষে

`double` বা `float` জাতীয় ডেটা টাইপ ব্যবহার করা উচিত নয়। কারণটা নিশ্চয়ই তোমরা বুঝে ফেলেছ, সেটা হচ্ছে প্রিসিশন (Precision) সমস্যা। `double` বা `float` জাতীয় সংখ্যা যোগ, বিয়োগ, গুণ, ভাগের সময় সাধারণত দশমিকের পরে কিছু অঙ্ক এলোমেলো হয়ে যায়। তাই, নিতান্ত দরকার না পরলে এই এই ডেটা টাইপগুলো ব্যবহার করব না।

আরও একটি নিষয় খেয়াল করো, প্রদত্ত ধারাটি একটি গুনোভর ধারা বা Geometric progression। তোমরা যারা, গুনোভর ধারার সাথে পরিচিত তারা কিন্তু চাইলে ধারার সমষ্টির সুত্র ব্যবহার করেও এই সমস্যাটি সমাধান করে ফেলতে পারো।

এখন তোমরা কোডটি গুছিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো। তারপর এই সমস্যার লিঙ্কে গিয়ে সাবমিট করে দেখো।

সমস্যা ৪১ – ধারার যোগফল ২

সমস্যার বিবরণ

$\frac{1}{1!} + \frac{2}{2!} + \frac{3}{3!} + \dots + \frac{n}{n!}$ এই ধারার যোগফল বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-041>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে ইনপুট হবে একটি পূর্ণসংখ্যা n ($n \leq 15$).

আউটপুট

প্রোগ্রামটির আউটপুটে n পর্যন্ত উপরোক্ত ধারাটির যোগফল প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	2.7083
5	2.7183
8	2.7183
10	

সমাধান

সমস্যাটির বর্ণনায় একটি ধারা দেওয়া আছে, আমাদের কাজ হবে ধারাটির যোগফল বের করা। প্রোগ্রামটির ইনপুট হবে দুটি সংখ্যা। প্রথমে একটি পূর্ণসংখ্যা T এবং পরবর্তী T সংখ্যক লাইনে ইনপুট হবে আরও একটি পূর্ণসংখ্যা n. যেহেতু, n এর রেঞ্জ দেওয়া আছে 15, সেখানে int ডেটা টাইপ নেওয়াটা যুক্তিসঙ্গত হলেও যেহেতু আমাদের ফ্যাক্টরিয়াল নিয়ে এই কোডটিতে কাজ করতে হবে তাই আমরা n এর জন্য ডেটা টাইপ নির long long int (15! বেশ বড় সংখ্যা!)

```
int T;
long long int n;
```

এবার ইনপুট নেওয়ার পালা। যেহেতু T সংখ্যক লাইনে ইনপুট নিতে হুবে তাই প্রথমে আমাদের ইনপুট নিতে হবে T। আমরা T এর মান ইনপুট নেওয়ার জন্য scanf() এর সাহায্য নিব এবং

T সংখ্যক লাইন পর্যন্ত N এর মান ইনপুট নেওয়ার জন্য আমরা 0 থেকে T-এর আগে পর্যন্ত একটি লুপ চালাব।

```
scanf("%d", &T);
for(int i = 0; i < T; i++) {}
```

ধারাটি খেয়াল করলে আমরা দেখতে পাই এখানে n পর্যন্ত প্রতিটি ক্রমবর্ধমান সংখ্যাকে সেই সংখ্যার ফ্যাক্টরিয়াল দিয়ে ভাগ করা হয়েছে। তাহলে আমাদের প্রথম কাজ হবে ফ্যাক্টরিয়ালের জন্য একটি ফাংশন লিখে ফেলা। আমরা জানি, কোনো সংখ্যার ফ্যাক্টরিয়াল সেই সংখ্যাটিসহ তার আগের ধনাত্মক পূর্ণসংখ্যাগুলোর গুণফল। যেমন: 5 এর ফ্যাক্টরিয়াল $120 (1 \times 2 \times 3 \times 4 \times 5 = 120)$ । আমাদের ফাংশনটি হবে এমন:

```
long long int fact(long long int n)
{
    long long int i, product = 1;
    for(i = 1; i <= n; i++)
    {
        product *= i;
    }
    return product;
}
```

এবার আমরা আসি main() ফাংশন লেখার কাজে। main() ফাংশনে আমরা T সংখ্যকবার n এর মান নিব এবং ধারাটির ফলাফল প্রিন্ট করব। যেহেতু, ধারাটিতে ভাগ করার কাজ আছে তাই আউটপুট দেখানোর সময় আমাদের ভাগফল রাখার জন্য double বা float টাইপের ভ্যারিয়েবল ব্যবহার করতে হবে। প্রথমেই আমরা n ইনপুট নেওয়ার কাজটি করব scanf() ফাংশন ব্যবহার করে।

```
scanf("%lld", &n);
```

ধারাটি যেহেতু n পর্যন্ত চলবে, তাই আমরা এখন n এর মান পর্যন্ত একটি লুপ চালাব যেটি ধারাটির পুরো হিসাবটি সহজেই করতে আমাদের সাহায্য করবে।

```
for(long long int i = 1; i <= n; i++)
```

ধারাটি একটি যোগফল নির্ণয়ের ধারা, তাই আমরা নতুন একটি ভেরিয়েবল "sum" এ ধারাটির ফলাফল জমা রাখাব। sum এর ডেটা টাইপ হবে double। এখানে আমরা i এর মানকে fact(i) দিয়ে ভাগ করে সেটি যোগ করতে থাকব n পর্যন্ত। আমাদের কোডটি হবে এমন:

```

double sum = 0.0;
for(i = 1; i <= n; i++)
{
    sum += ( (double)i / fact(i) );
}

```

লক্ষ করে দেখো, এখানে আমরা `i` এর মান `double` এ টাইপকাস্টিং করেছি। কারণ `i` একটি পূর্ণসংখ্যা এবং `sum` এ আমরা দশমিক সংখ্যা হিসেবে রেখেছি। ডেটা টাইপের হেরফেরের কারণে যেন আউটপুট ভুল না আসে সেজন্য এরূপ করা হয়েছে।

এবার আমরা `printf()` ফাংশনের সাহায্যে আউটপুট প্রিন্ট করলেই আমাদের কাজ শেষ। নমুনা আউটপুট খেয়াল করে আমরা দেখতে পাই এখানে আউটপুট দশমিকের পর চার ঘর পর্যন্ত প্রিন্ট করেছে। আমাদের প্রিন্ট statement হবে এমন:

```
printf("%0.4lf\n", sum);
```

এখন তোমরা কোডটি গুছিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো। তারপর এই সমস্যার লিঙ্কে গিয়ে সাবমিট করে দেখো।

সমস্যা ৪২ – ধারার যোগফল ৩

সমস্যার বিবরণ

$$\sum_{k=n}^0 2^k$$

উপরোক্ত সমীকরণটিকে নমুনা আউটপুটের মত করে প্রিন্ট করতে হবে।
[\(http://bit.ly/1-52-problem-042\)](http://bit.ly/1-52-problem-042)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে ইনপুট হবে একটি পূর্ণসংখ্যা N ($N \leq 50$)।

আউটপুট

প্রোগ্রামটির আউটপুটে n পর্যন্ত উপরোক্ত সমীকরণটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
3	$2^5 + 2^4 + 2^3 + 2^2 + 2 + 1$
5	$2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1$
8	$2^{10} + 2^9 + 2^8 + 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2 + 1$
10	

সমাধান

সমস্যাটির বর্ণনায় একটি সমীকরণ দেওয়া আছে, আমাদের কাজ হবে সমীকরণটির সমাধান বের করা। প্রোগ্রামটির ইনপুট হবে দুটি সংখ্যা। প্রথমে একটি পূর্ণসংখ্যা T এবং পরবর্তী T সংখ্যক লাইনে ইনপুট হবে আরও একটি পূর্ণসংখ্যা N। যেহেতু, N এর রেঞ্জ দেওয়া আছে 50, তাই আমরা ডেটা টাইপ নিব int।

```
int T, N;
```

আমরা প্রথমে T এর মান ইনপুট নেওয়ার জন্য scanf() ব্যবহার করবো এবং T সংখ্যক লাইন পর্যন্ত N এর মান ইনপুট নেওয়ার জন্য আমরা T পর্যন্ত একটি লুপ ব্যবহার করবো।

```
scanf("%d", &T);
for(int i = 0; i < T; i++) {}
```

সমীকরণটি খেয়াল করলে আমরা দেখতে পাই, এখানে k এর মান N থেকে শুরু হয়ে 0-তে পৌঁছেছে। অর্থাৎ reverse লুপ দিয়ে আমাদের কাজটি করতে হবে। আমরা এমন একটি লুপ চালাব যেটির মান N থেকে শুরু হয়ে এক এক করে কমে 0 তে পৌঁছাবে। এই কাজটি করার আগে আমাদের N এর মান ইনপুট নিতে হবে। N-এর মান T সংখ্যক লাইনে থাকবে, তাই আমরা N এর মান ইনপুট নিব প্রথম লুপটির ভেতরে।

```
scanf("%d", &N);
for(int k = N; k >= 0; k--) {}
```

এখন আমাদের মূল কোডটি লেখার পালা। আমাদের প্রতিটি k এর মানের জন্য প্রিন্ট করতে হবে 2^k । কিন্তু যখন, k এর মান হয়ে যাবে 1 তখন আমরা আর 2^1 প্রিন্ট না করে প্রিন্ট করব শুধু 2 আর যখন k এর মান হয়ে যাবে 0 তখন আমরা 2^0 এর বদলে প্রিন্ট করব 1।

```
if(k == 1) printf("1\n");
else if(k == 2) printf("2 + ");
else printf("2^%d + ", k);
```

এখন তোমরা কোডটি শুনিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো। N এর মান যদি 1 অথবা 2 হয় তখন কি এই কোডটি সঠিক আউটপুট দেয়?

সমস্যা ৪৩ – হিসাব-কিতাব

সমস্যার বিবরণ

$p^q \bmod c$ এর মান নির্ণয় করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-43>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে ইনপুট হবে তিনটি পূর্ণসংখ্যা p, q, c ($1 \leq p, q, c \leq 100$)

আউটপুট

প্রোগ্রামটির আউটপুটে "Result" কথাটি প্রিন্ট করতে হবে। আউটপুটে "=" চিহ্নের আগের এবং পরের স্পেসটি লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
3	Result = 2
2 3 3	Result = 4
2 10 5	Result = 1
50 2 3	

সমাধান

এই সমস্যাটি একটি গাণিতিক সমস্যা। সমস্যার বর্ণনায় বলা আছে প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T এবং পরবর্তী T সংখ্যক লাইনে ইনপুট নিতে হবে আরও তিনটি পূর্ণসংখ্যা p, q, c। যেহেতু বলা আছে ($p, q, c \leq 100$), সেক্ষেত্রে আমরা p, q, c ভেরিয়েবলের ডেটা টাইপ হবে int।

```
int p, q, c, T;
```

এবার ইনপুট নেওয়ার পালা। আমাদের প্রথম ইনপুট হবে T। এরপর T সংখ্যক লাইনে p, q, c নেওয়ার জন্য একটি লুপ চালাতে হবে T পর্যন্ত।

```
scanf("%d", &T);
for(int i = 0; i < T; i++)
{
    scanf("%d %d %d", &p, &q, &c);
}
```

সমীকরণটিতে p এর power হচ্ছে q। power এর কাজটি করার জন্য যদি আমরা এখানে math.h হেডার ফাইলের অন্তর্গত pow() ফাংশনটি ব্যবহার করি তাহলে কিন্তু কাজ হবে না। কেন বলো তো? কারণ, pow() ফাংশনের রিটার্ন টাইপ হচ্ছে double আর double টাইপের সংখ্যার ওপর মডুলাস অপারেটর ব্যবহার করে ভাগশেষ বের করা যায় না। তাই এটিকে আবার আমাদের ইন্টিজারে টাইপকাস্ট করতে হবে, যেটা মাঝে মধ্যে ক্রটিপূর্ণ ফলাফল দিতে পারে (error-prone)। আমরা আগেও বলেছি যদি int ব্যবহার করে সমাধান করা যায় তবে আমরা

চেষ্টা করব `double` বা `float` ব্যবহার না করার। তাই আমরা লুপ ব্যবহার করে এই পাওয়ারটির মান নির্ণয় করব।

```
int temp = 1;
for(int j = 1; j <= q; j++)
{
    temp *= p;
}
```

এবার আমরা `temp` কে `c` দিয়ে মডুলাস করব।

```
int res = temp % c;
```

এখন আউটপুট প্রিন্ট করার পালা।

```
printf("Result = %d\n", res);
```

এখানে খেয়াল করো, আমি আউটপুট প্রিন্ট করার সময় একটি নিউলাইন প্রিন্ট করেছি এবং আউটপুটের নির্দেশনা অনুযায়ী `"=` চিহ্নের আগে পরে স্পেস ক্যারেক্টার দিয়েছি।

এই সমাধানে কিন্তু একটি ভুল আছে, বলা হয়েছে, `p` ও `q` এর সর্বোচ্চ সীমা হতে পারে 100।

যখন, $p = 100$ এবং $q = 100$, তখন, $p^q = 10^{200}$

এখন এই সংখ্যাটি তো `int` এর সর্বোচ্চ সীমার চেয়ে অনেক বড়। তাহলে এত বড় সংখ্যা আমরা `int` বা `long long int` দিয়েও রাখতে পারব না। এখানে ওভারফ্লো হবে এবং আমরা ভুল উত্তর পাব। তাহলে কী করা যায়? চলো, আবারও একটি বুদ্ধি শিখে নিই:

$$\begin{aligned} p^{n-1} &= a \pmod{m} \dots \dots \dots \quad (1) \\ p &= b \pmod{m} \dots \dots \dots \quad (2) \\ \hline p^n &= a * b \pmod{m} \dots \dots [(1) \text{ ও } (2) \text{ গুণ করে পাই}] \end{aligned}$$

উপরের সমীকরনটি অনেকের কাছে কঠিন মনে হতে পারে। তব পাওয়ার কিছু নেই। এখানে বলা হয়েছে যে, p^{n-1} এর সঙ্গে p গুণ করলে যেমন p^n পাওয়া যায়, তেমনি p^{n-1} কে m দিয়ে ভাগ করে পাওয়া ভাগশেষের সঙ্গে, p কে m দিয়ে ভাগ করে পাওয়া ভাগশেষ গুণ করলেও, p^n কে m দিয়ে ভাগ করার পরের ভাগশেষ পাওয়া যাবে। এই লাইনটি পড়ে যদি খটমট লেগে যায়, তবে ঠান্ডা মাথায় আরও কয়েকবার পড়, ঠিক বুঝে ফেলবে। তার মানে বোঝা গেল p^q কে `c` দিয়ে ভাগ করে

কত ভাগশেষ হয়, তা বের করার জন্য আমাদের p^q এর প্রকৃত মান বের করার দরকার নেই। আমরা প্রতি ধাপে শুধু ভাগশেষ বের করতে থাকলেই সঠিক ফলাফল পাব।

```
for(int j = 1; j <= q; j++)  
{  
    temp = (temp * p) % c;  
}
```

এখন প্রতি ধাপেই আমরা temp কে c দিয়ে মড করে ভাগশেষ বের করে ফেলছি, তাই temp এর মান কখনোই c এর চেয়ে বড় হবে না। অর্থাৎ, ওভারফ্লো হওয়ার কোনো সম্ভাবনা নেই।

এই সমাধানটিতে একটি অপটিমাইজেশনও করা সম্ভব। সেটা আমি করে দেব না তবে বুদ্ধিটা তোমাদের শিখিয়ে দেব। খেয়াল করে দেখো p^q এর মান নির্ণয় করার জন্য আমাদের q সংখ্যকবার লুপ চালাতে হয়েছে। আমরা কি এর কম সংখ্যক বার লুপ চালিয়েই p^q এর মান বের করতে পারি? ধরা যাক, q এর মান যদি হয় 100 আমরা লিখতে পারি:

$$p^{100} = p^{50+50} = p^{50} \cdot p^{50} = (p^{50})^2$$

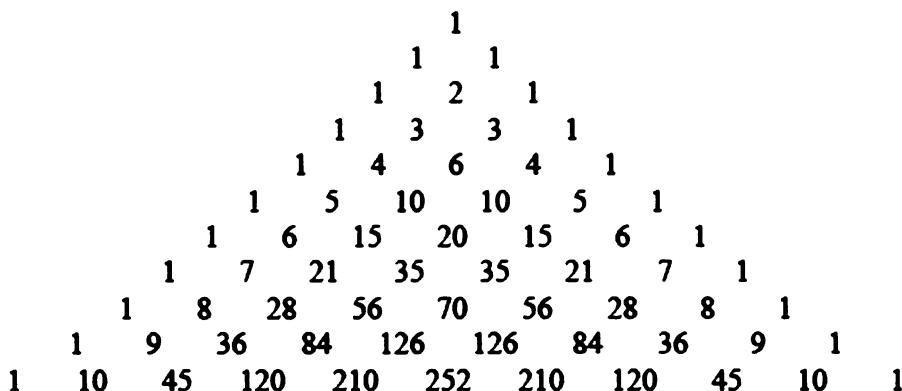
অর্থাৎ, দেখো আমরা যদি p⁵⁰ এর মান বের করে ফেলি তাহলে কিন্তু ওই সংখ্যাটির বর্গ করলেই p¹⁰⁰ এর মান পাওয়া যায়। আরও 50 বার লুপ চালানোর দরকার পড়ে না। একইভাবে, p⁵⁰ এর মান বের করার জন্য আবার কিন্তু p²⁵ এর মানকে বর্গ করলেই হয়ে যায়। তাহলে তোমরা একটু দেখো তো এই বুদ্ধি ব্যবহার করে কোডটিকে অপটিমাইজ করা যায় কি না? এই সমস্যায় যদিও p ও q এর সর্বোচ্চ সীমা খুবই কম (মাত্র 100) কিন্তু এই বুদ্ধি ব্যবহার করলে অনেক বড় বড় পাওয়ারও খুব অল্প সময়ে গণনা করা যাবে।

এখন তোমরা কোডটি শুনিয়ে লিখে ফেলো। প্রথমে কম্পাইল ও রান করবে। যেই নমুনা ইনপুট দেওয়া আছে, সেগুলো দিয়ে দেখবে যে নমুনা আউটপুটের সঙ্গে মিলে কি না। নিজে কিছু ইনপুট দিয়েও পরীক্ষা করতে পারো।

সমস্যা ৪৪ – প্যাসকেলের ত্রিভুজ ১

সমস্যার বিবরণ

প্যাসকেলের ত্রিভুজ গণিত শাস্ত্রের একটি আকর্ষণীয় সংখ্যার উদাহরণের নাম। ফরাসি গণিতবিদ এবং দার্শনিক ব্রেইজ প্যাসকেলের নামানুসারে এই সংখ্যা নমুনার (Number pattern) নাম প্যাসকেলের ত্রিভুজ। এই ত্রিভুজের উল্লেখিত যেকোনো সংখ্যা এর ওপরের সারির দৃটি সংখ্যার যোগফল। উদাহরণস্বরূপ: পরের ছবির সপ্তম সারির চতুর্থ কলামের সংখ্যাটি 20, যেটি ৬ষ্ঠ সারির তৃতীয় এবং ৪ৰ্থ কলামের সংখ্যা 10 এবং 10 এর যোগফলের সমান।



তোমার কাজ হচ্ছে এমন একটি প্রোগ্রাম লেখা, যেটি একটি নির্দিষ্ট সারি পর্যন্ত প্যাসকেলের ত্রিভুজ প্রিন্ট করবে।

(<http://bit.ly/1-52-problem-44>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 20$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে একটি করে পূর্ণসংখ্যা N ($N \leq 20$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে N তম সারি পর্যন্ত প্যাসকেলের ত্রিভুজ প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2	1
3 4	1 1 1 2 1 1 3 3 1
	1
	1 1 1 2 1 1 3 3 1 1 4 6 4 1

সমাধান

সমস্যার বর্ণনা যদি মনোযোগ দিয়ে পড়, তাহলে লক্ষ করবে যে প্রতি লাইনের সংখ্যাগুলো দিয়ে পরের লাইনের সংখ্যাগুলো তৈরি করা যায়। এটি অবশ্য বলেও দেওয়া আছে। আর প্রথম লাইন বাদে বাকি সব লাইনের শুরুর সংখ্যা ও শেষের সংখ্যা হচ্ছে 1।

প্রথম দুটি লাইন আমরা বিশেষভাবে বিবেচনা করতে পারি। যে N-এর মান 1 হলে কী আউটপুট হবে আর N-এর মান 2 হলে কী আউটপুট হবে, সেটি একটি কন্ডিশন দিয়ে পরীক্ষা করতে পারি।

এখন যদি N-এর মান 2-এর চেয়ে বেশি হয়, তাহলে কী হবে? N-এর যেকোনো মানের জন্যই প্রথম ও শেষ সংখ্যা হবে 1। আর অন্য যেকোনো সংখ্যা, ধরা যাক m-তম সংখ্যা, হবে আগের লাইনের $(m-1)$ -তম সংখ্যা ও m-তম সংখ্যার যোগফল। বিষয়টি পরিষ্কার না হলে কাগজে একটু লিখে দেখো, বুঝতে সমস্যা হবে না। তাহলে N-তম লাইনের সংখ্যাগুলো বের করতে হলে $(N-1)$ -তম (অর্থাৎ আগের লাইন) লাইনের সংখ্যাগুলো জানা থাকতে হবে। আবার আগের লাইনের সংখ্যাগুলো কী, সেটি জানতে হলে তার আগের লাইনে কী কী সংখ্যা ছিল, সেটি জানতে হবে। তাহলে নতুন লাইনের জন্য একটি অ্যারে আর তার আগের লাইনের জন্য একটি অ্যারে ব্যবহার করতে হবে। ধরা যাক, একটি অ্যারের নাম odd, আরেকটির নাম even।

শুরুতে even-এর ভেতর 1, 1 রেখে দিলাম $\boxed{\text{even}[0] = 1; \text{even}[1] = 1;}$ । প্যাসকেলের ত্রিভুজের দিকে তাকালে দেখবে যে এটি হচ্ছে দ্বিতীয় লাইন। এখন আমি যদি তৃতীয় লাইনের সংখ্যাগুলো বের করতে চাই, তাহলে সেগুলো odd অ্যারেতে রাখতে হবে। আমরা জানি যে odd অ্যারেতে তিনটি উপাদান থাকবে (কেন? প্যাসকেলের ত্রিভুজের দিকে আবার তাকাও)। প্রথম উপাদান সবসময়ই 1, তাহলে আমরা লিখে দিতে পারি যে $\text{odd}[0] = 1;$ । এখন আমরা

একটি লুপ চালিয়ে odd অ্যারের শেষ উপাদান বাদে বাকি উপাদানগুলো বের করব। আর লুপের শেষে odd অ্যারের শেষ উপাদান 1 অ্যাসাইন করে দিব। এখন odd অ্যারেতে আমরা তৃতীয় লাইনের সংখ্যাগুলো পেয়ে গেলাম। তারপর এই odd অ্যারে থেকে আবার even অ্যারে তৈরি করব, যেখানে চতুর্থ লাইনের সংখ্যাগুলো থাকবে। তোমরা নিজেরা কোড করার চেষ্টা করো, তবে প্রোগ্রামিংয়ে একেবারে নতুনদের জন্য কাজটি একটু কঠিনই হবে।

```

int odd[21], even[21];
int i, j, N;

scanf("%d", &N);

even[0] = 1;
even[1] = 1;

for(i = 3; i <= N; i++)
{
    if(i % 2 == 0)
    {
        even[0] = 1;
        for (j = 1; j < i - 1; j++)
        {
            even[j] = odd[j - 1] + odd[j];
        }
        even[j] = 1;
    }
    else
    {
        odd[0] = 1;
        for (j = 1; j < i - 1; j++)
        {
            odd[j] = even[j - 1] + even[j];
        }
        odd[j] = 1;
    }
}
    
```

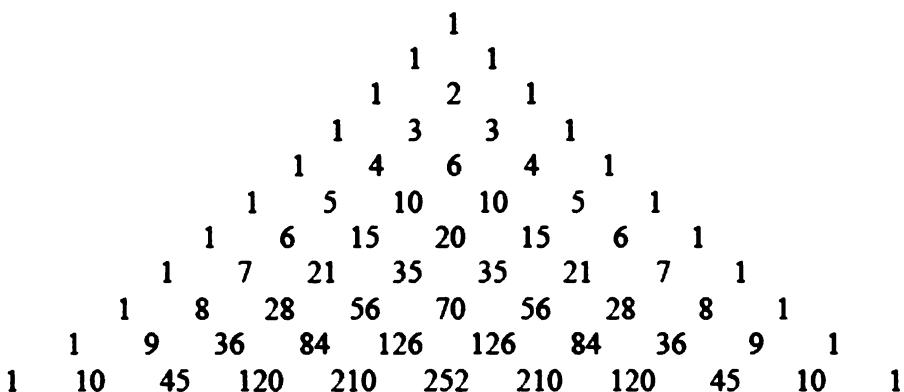
ওপরের কোডে তোমরা লক্ষ করো, আমি কন্ডিশনাল লজিক ব্যবহার করে নির্ণয় করেছি যে এখন odd থেকে even তৈরি করব নাকি even থেকে odd? আর শুরুর ও শেষের 1 কীভাবে অ্যারেতে বসিয়ে দিয়েছি, সেটিও খেয়াল কর। কোড ঠিকঠাক বোঝার পর তোমার একটু কোড লিখতে হবে, অ্যারে প্রিন্ট করার জন্য। এখন তোমাদের জন্য প্রশ্ন, কোন অ্যারে প্রিন্ট করবে? সেটি কীভাবে বুঝবে?

তোমরা কোডটি ঠিকঠাকভাবে লিখে আগে দেখো উদাহরণে যে ইনপুট-আউটপুট দেওয়া আছে, তার সঙ্গে মিলে কি না। যদি মিলে, তাহলে কোড সাবমিট করে দাও। আশা করি তোমার কোড সঠিক হবে।

সমস্যা ৪৫ – প্যাসকেলের ত্রিভুজ ২

সমস্যার বিবরণ

প্যাসকেলের ত্রিভুজ গণিত শাস্ত্রের একটি আকর্ষণীয় সংখ্যার উদাহরণের নাম। ফরাসি গণিতবিদ এবং দার্শনিক ব্রেইজ প্যাসকেলের নামানুসারে এই সংখ্যা নমুনার (Number pattern) নাম প্যাসকেলের ত্রিভুজ। এই ত্রিভুজের উল্লেখিত যেকোনো সংখ্যা এর ওপরের সারির দুটি সংখ্যার যোগফল। উদাহরণস্বরূপ: পরের ছবির সপ্তম সারির চতুর্থ কলামের সংখ্যাটি 20, যেটি ষষ্ঠ সারির তৃতীয় এবং ৪র্থ কলামের সংখ্যা 10 এবং 10 এর যোগফলের সমান।



তোমার কাজ হচ্ছে এমন একটি প্রোগ্রাম লেখা, যেটি একটি নির্দিষ্ট সারি পর্যন্ত প্যাসকেলের ত্রিভুজ প্রিন্ট করবে।

(<http://bit.ly/1-52-problem-45>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 100000$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে একটি করে পূর্ণসংখ্যা N ($N \leq 50$) ইনপুট নিতে হবে।

আউটপুট

প্রোগ্রামটির আউটপুটে N তম সারি পর্যন্ত প্যাসকেলের ত্রিভুজ প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2	1
3 4	1 1 1 2 1 1 3 3 1

ইনপুট	আউটপুট
	1
	1 1 1 2 1 1 3 3 1 1 4 6 4 1

সমাধান

প্রথমেই তোমরা প্যাসকেলের ত্রিভুজ - 1 সমস্যার সমাধানের জন্য যেই প্রোগ্রাম লিখেছ, সেটি আবার সাবমিট করো। এখানে দেখবে টাইম লিমিট শেষ হয়ে গেছে দেখাচ্ছে। মানে তোমার কোড চলতে বেশি সময় নিচ্ছে।

এখন, তোমরা একটি ব্যাপার খেয়াল করো। প্যাসকেলের ত্রিভুজের 10তম লাইন প্রিন্ট করতে গেলে তৃতীয় লাইন থেকে লুপ শুরু হয়ে 10তম লাইন পর্যন্ত চলবে, আবার 20-তম লাইন বের করতে হলেও আবার সেই তৃতীয় লাইন থেকেই কাজ শুরু হবে। এভাবে একই কাজ বার বার করা লাগচ্ছে। আর এই সমস্যার বর্ণনায় টেস্ট কেসের সংখ্যাও বেশি। তাই সময়ে কুলাচ্ছে না। কী করা যায়?

আমাদের যদি 50তম লাইন বের করতে হয়, তাহলে কিন্তু 3 থেকে 47তম লাইনের প্রতিটি লাইন বের করেই আসতে হবে। তো এই লাইনগুলো যদি আমরা একটি অ্যারেতে রেখে দিতে পারি, তাহলে কিন্তু বার বার হিসাব করা লাগবে না। কেবল কোন লাইনটি বের করতে হবে জানলে

আমরা অ্যারের সেই লাইনটি বের করে দিতে পারব। তোমরা নিশ্চয়ই বুঝতে পারছ যে এখানে দুই মাত্রার অ্যারে (2D অ্যারে) ব্যবহার করতে হবে। কোডের মূল অংশ আমি লিখে দিলাম।

```
long long int pascal[51][51];
int i, j;

pascal[1][0] = 1;
pascal[2][0] = 1;
pascal[2][1] = 1;

for(i = 3; i <= 10; i++)
{
    pascal[i][0] = 1;

    for(j = 1; j < i - 1; j++)
    {
        pascal[i][j] = pascal[i-1][j-1] + pascal[i-1][j];
    }

    pascal[i][j] = 1;
}
```

এখন N-এর মান ইনপুট নিয়ে pascal[N] রো-টি প্রিন্ট করতে হবে। সবার শেষে কিছু অতিরিক্ত স্পেস থাকলে চলবে না।

এই পদ্ধতি বা টেকনিক যদি তোমরা আয়ত্তে এনে ফেলতে পারো, তাহলে বিরাট সুবিধা। অনেক সমস্যাই তখন তোমাদের জন্য সমাধান করা সহজ হয়ে যাবে। তোমরা কেবল চিন্তা করবে, কোনো সমস্যায় এ রকম অ্যারেতে আগেভাগে সবকিছু রেখে দেওয়া যায় কি না। যেমন ফ্যাক্টরিয়াল নাম্বার, ফিবোনাচি নাম্বার, কিংবা সিভ পদ্ধতিতে মৌলিক সংখ্যা বের করা - এসব সমস্যার কথা চিন্তা করতে পারো।

আরেকটি বিষয়। কোডে `long long int` ব্যবহার করেছি কারণ একটি উপাদানের সর্বোচ্চ দৈর্ঘ্য 15 হতে পারে, মানে সংখ্যায় 15টি অঙ্ক থাকতে পারে, যেটি `int` টাইপের ভ্যারিয়েবলে ধারণ করা যাবে না। আর প্রিন্ট করার জন্য `%lld` ব্যবহার করতে হবে।

সমস্যা ৪৬ – ত্রিভুজের ক্ষেত্রফল

সমস্যার বিবরণ

একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য দেওয়া আছে। ত্রিভুজের ক্ষেত্রফল বের করার প্রোগ্রাম লিখতে হবে।

(<http://bit.ly/1-52-problem-46>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 1000$), যা টেন্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে থাকবে তিনটি করে পূর্ণসংখ্যা a, b, c ($1 \leq a, b, c \leq 100$)। এখানে লক্ষণীয়, ত্রিভুজের যেকোনো দুই বাহুর যোগফল অবশ্যই তৃতীয় বাহু অপেক্ষা বড় হবে।

আউটপুট

প্রোগ্রামটির আউটপুট হবে ত্রিভুজের ক্ষেত্রফল "Area", যেটি দশমিকের পর 3 ঘর পর্যন্ত প্রিন্ট করবে। এখানে '=' চিহ্নের আগের ও পরের স্পেসটি লক্ষণীয়।

উদাহরণ

ইনপুট	আউটপুট
3	Area = 216.000
24 30 18	Area = 95.917
13 18 15	Area = 173.205
20 20 20	

সমাধান

ত্রিভুজের ক্ষেত্রফল বের করার সমস্যা এটি। একটি ত্রিভুজের তিন বাহুর দৈর্ঘ্য দেওয়া থাকবে ত্রিভুজটির ক্ষেত্রফল বের করার প্রোগ্রাম লিখতে হবে। এই সমস্যাটির সমাধান ত্রিভুজের ক্ষেত্রফলের সূত্র দিয়ে সমাধান করা যায়। সূত্রটি তোমরা নবম-দশম শ্রেণির গণিত বইয়ে পেয়ে যাবে। জানা না থাকলে বই থেকে দেখে নিবে।

ইনপুটের বর্ণনায় বলা আছে T সংখ্যক লাইনে a, b, c -এর ইনপুট নিতে হবে; যেখানে a, b, c এবং T প্রত্যেকেটিই পূর্ণসংখ্যা।

-

```

scanf("%d", &T);
while(T--)
{
    scanf("%d %d %d", &a, &b, &c);
}

```

এখনো পর্যন্ত যদি তোমরা ত্রিভুজের ক্ষেত্রফলের সূত্রটি জেনে না থাকো তাহলে আমি এটি এখানে উল্লেখ করে দিচ্ছি:

$$\text{অর্ধপরিসীমা}, s = \frac{(a + b + c)}{2}$$

$$area = \sqrt{s \times (s - a) \times (s - b) \times (s - c)}$$

সূত্রটিতে বর্গমূলের প্রয়োগ আছে তোমরা দেখতে পারছো। এর আগেও তোমরা পূর্ণবর্গ সংখ্যার প্রোগ্রাম করে এসেছো। তাই এখানে আর বিস্তারিত কিছু দিলাম না।

আউটপুটে আমাদের ক্ষেত্রফল প্রিন্ট করতে হবে। সেজন্য আউটপুটের বর্ণনায় বলা না থাকলেও এখানে ক্ষেত্রফল এর জন্য **double** ডাটা টাইপ ব্যবহার করতে হবে।

সমস্যা ৪৭ – অ্যারের জ্ঞেট

সমস্যার বিবরণ

দুটি ছোট থেকে বড় ক্রমে সাজানো অ্যারে দেওয়া থাকবে। অ্যারে দুটিকে যুক্ত করে একটি ক্রমানুসারে সাজানো অ্যারে তৈরি করতে হবে।

(<http://bit.ly/1-52-problem-47>)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($T \leq 1000$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তীতে T সংখ্যক ইনপুট কেস থাকবে। প্রতিটি ইনপুট কেসে দুটি করে লাইন থাকবে।

প্রথম লাইনে প্রথমে একটি সংখ্যা n1 ($1 \leq n1 \leq 100$) থাকবে এবং তারপরে n1 সংখ্যক পূর্ণসংখ্যা থাকবে। একইভাবে দ্বিতীয় লাইনে প্রথমে একটি সংখ্যা n2 ($1 \leq n2 \leq 100$) থাকবে এবং তারপরে n2 সংখ্যক পূর্ণসংখ্যা থাকবে। দুক্ষেত্রেই সংখ্যাগুলো ছোট থেকে বড় ক্রমে সাজানো থাকবে। অ্যারেদুটির সবগুলো উপাদানই 100000-এর চেয়ে ছোট্টো হবে।

আউটপুট

প্রতিটি ইনপুট কেসের জন্য একটি লাইনে $n1 + n2$ সংখ্যক সংখ্যা আউটপুট দিতে হবে। যেখানে দুটি অ্যারের সবগুলো সদস্য থাকবে ছোট থেকে বড় ক্রমে সাজানো।

উদাহরণ

ইনপুট	আউটপুট
2 3 1 3 5 2 4 10 5 10 20 30 40 50 3 15 21 22	1 3 4 5 10 10 15 20 21 22 30 40 50

সমাধান

দুটি সর্টেড (ছোট থেকে বড় ক্রমে) অ্যারে দেওয়া থাকবে, তোমার কাজ হচ্ছে অ্যারে দুটি যুক্ত করা। প্রথমে তোমরা নিচের কোড দেখো, তারপরে সমাধান পুরোপুরি লিখে ফেলো।

```

int main()
{
    int a[] = {1, 2, 4, 7, 9, 10};
    int b[] = {1, 2, 3, 4, 7, 8, 9, 10};
    int c[20];
    int i = 0, j = 0, k = 0, n1 = 6, n2 = 8, n;

    while(1)
    {
        if (a[i] < b[j])
        {
            c[k] = a[i];
            k++;
            i++;
        }
        else
        {
    }
}

```

```

        c[k] = b[j];
        k++;
        j++;
    }
}
n = k;

printf("%d", c[0]);
for(i = 1; i < n; i++)
{
    printf(" %d", c[i]);
}
printf("\n");
return 0;
}

```

আমি যতটুকু কোড লিখেছি, সেটি কেবল দেখানোর জন্য যে সমস্যার সমাধান কীভাবে করতে হবে। কোডে যে লুপ আছে, সেটি কিন্তু আপনাআপনি বন্ধ হবে না, তোমার নিজে থেকে শর্ত জুড়ে দিতে হবে, যেন কাজ শেষ হলে লুপ থেকে ব্রেক করা যায়। আমার কোড অসম্পূর্ণ, তোমরা সেটি সম্পূর্ণ করবে। নিচে আর প্রোগ্রামিংয়ের আনন্দ কোথায়!

এখন তোমাদের জন্য আরেকটি সমস্যা। অ্যারের জোট সমস্যায় যদি এমন শর্ত থাকত যে, দুটি অ্যারে জোটবন্ধ হওয়ার পরে যেই অ্যারেটি তৈরি হবে, সেখানে কোনো সংখ্যা একাধিকবার থাকতে পারবে না, তাহলে সমাধান কী?

সমস্যা 48 – নিখৌজি সংখ্যা

সমস্যার বিবরণ

একটি অ্যারেতে 1 থেকে শুরু করে n পর্যন্ত যতগুলো সংখ্যা আছে তার একটি বাদে সবগুলো সংখ্যা দেওয়া থাকবে। কোন উপাদানটি অনুপস্থিত সেটি বলতে হবে।
[\(http://bit.ly/1-52-problem-48\)](http://bit.ly/1-52-problem-48)

ইনপুট

প্রোগ্রামটির প্রথম ইনপুট হবে একটি পূর্ণসংখ্যা T ($1 \leq T \leq 100$), যা টেস্ট কেসের সংখ্যা নির্দেশ করে। পরবর্তী T সংখ্যক লাইনে একটি করে পূর্ণসংখ্যা N ($1 \leq N \leq 100000$) ইনপুট নিতে হবে, যা অ্যারের উপাদান সংখ্যা নির্দেশ করে। পরবর্তীতে N-1 সংখ্যক অ্যারে উপাদান ইনপুট নিতে হবে। অ্যারের কোনো উপাদানই N-এর চেয়ে বড় হবে না।

আউটপুট

প্রোগ্রামটির আউটপুটে কোন অ্যারের কোন উপাদানটি নেই সেটি খুঁজে বের করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
2	7
7 2 1 4 6 5 3	3
9 9 4 5 8 6 1 7 2	

সমাধান

সমস্যার বর্ণনায় বলা আছে যে 1 থেকে N পর্যন্ত সংখ্যাগুলোর মধ্যে N-1টি সংখ্যা আছে, একটি নির্খোঁজ। সেই নির্খোঁজ সংখ্যাটি আমাদের খুঁজে বের করতে হবে। কীভাবে করব?

প্রথমে একটি অ্যারেতে সংখ্যাগুলো ইনপুট নেই:

```
scanf("%d", &N);
for(i = 0; i < N - 1; i++)
{
    scanf("%d", &ara[i]);
}
```

এবারে 1 থেকে N পর্যন্ত প্রতিটি সংখ্যা অ্যারের মধ্যে খুঁজব। যেই সংখ্যাটি খুঁজে পাব না, সেটিই হচ্ছে নির্খোঁজ। এজন্য নেস্টেড লুপ ব্যবহার করতে হবে।

```
for (i = 1; i <= N; i++)
{
    found = 0;
    for(j = 0; j < N - 1; j++)
    {
        if (i == ara[j])
```

```

    {
        found = 1;
    }
}
if (found == 0)
{
    printf("Missing number: %d\n", i);
}
}

```

তোমরা এখন পুরো কোডটি লিখে ফেল। ঠিকঠাক কোড করে সাবমিট করলে দেখবে টাইম লিমিট (Time Limit Exceeded) দেখাচ্ছে। কোডটি আরও ইফিশিয়েন্ট করা যায়। তোমরা যদি একটু চিন্তা করো, দেখবে ওপরের কোডে দুই জায়গায় break ব্যবহারের সুযোগ রয়েছে। কোনো সংখ্যা অ্যারেতে খুঁজে পেলে আর লুপ চালানোর দরকার নেই। আর নিখোঁজ সংখ্যা খুঁজে পেলেও লুপ চালানোর দরকার নেই। তাই প্রয়োজনীয় জায়গায় break ব্যবহার করে তুমি প্রোগ্রামের রান্টাইম অনেক কমিয়ে ফেলতে পারো। প্রোগ্রামটি ঠিক করে আবার অনলাইনে জমা দিয়ে দেখো।

এবারে আমরা দ্বিতীয় একটি সমাধান দেখব। যেহেতু N-এর সর্বোচ্চ মান হতে পারে 100000, আমরা 100001 উপাদান ধারণ করতে পারে এমন একটি অ্যারে নেব। শুরুতে সেই অ্যারের সব উপাদান 0 করে দেব। তারপর যেসব সংখ্যা ইনপুট নেওয়া হবে, অ্যারের সেই উপাদানটিতে 1 অ্যাসাইন করব। যেমন ইনপুটে যদি 7 থাকে, তাহলে ara[7] = 1;। তারপর সবগুলো সংখ্যা ইনপুট নেওয়া শেষ হলে অ্যারেতে শুরু থেকে শেষ পর্যন্ত একটি লুপ চালিয়ে দেখব যে কোন ঘরাটি 0 আছে। কেবলমাত্র একটি ঘরাই 0 থাকবে, আর সেই ঘরাটিই হচ্ছে আমাদের নিখোঁজ সংখ্যা। এবারে কোড লেখার পালা :

```

int ara[100001];
int i, N, num, missing;

scanf("%d", &N);

for(i = 1; i <= N; i++) ara[i] = 0;

for(i = 0; i < N; i++)
{
    scanf("%d", &num);
    ara[num] = 1;
}

for(i = 1; i <= N; i++)
{
    if (ara[i] == 0)

```

```

{
    missing = i;
    break;
}
printf("Missing number: %d\n", missing);

```

আশা করি, তোমরা কোড দেখে বুঝতে পেরেছ। এবারে সম্পূর্ণ প্রোগ্রাম নিজে নিজে লিখে ফেলো আর অনলাইন জাজে জমা দাও।

আচ্ছা, কেউ কি অ্যারে ব্যবহার না করে সমাধান করতে পারবে? বইটি বন্ধ করে কয়েক ঘণ্টা চিন্তা করে দেখোই না, পারো কি না।

চিন্তা করে না পারলে আমি এবারে হিন্টস্ দিয়ে দিই। ধরা যাক 1 থেকে 10 পর্যন্ত সংখ্যাগুলোর মধ্যে একটি বাদে বাকি ৭টি সংখ্যা দেওয়া আছে: 10, 3, 2, 9, 5, 8, 4, 7, 1। এখন 1 থেকে 10 পর্যন্ত সংখ্যাগুলোর যোগফল হচ্ছে 55। আর যেসব সংখ্যা দেওয়া আছে, সেগুলোর যোগফল 49। 55 থেকে 49 বিয়োগ করলে থাকে 6, আর এই 6-ই হচ্ছে আমাদের নিখোঁজ সংখ্যা। তোমরা যারা ধারার যোগফল কীভাবে বের করতে হয় জানো, তাদের সমাধান লিখতে কোনো সমস্যা হওয়ার কথা নয়। আর যারা জানো না, তারা স্কুলের গণিত বইতে দেখে নাও (নবম-দশম শ্রেণির বইতে আছে, অষ্টম শ্রেণির বইতেও থাকার কথা)।

দেখলে তো, একটি সমস্যা কতভাবে সমাধান করা যায়। কীভাবে রানটাইম কমিয়ে প্রোগ্রামের গতি বাঢ়ানো যায়। আর কীভাবে যেমোরিও কম খরচ করে প্রোগ্রাম লেখা যায়। আসলেই প্রোগ্রামিংয়ের জগতটি বড়ই আশ্চর্য!

সমস্যা ৪৯ – মৌলিক কি না

সমস্যার বিবরণ

একটি সংখ্যা মৌলিক কি না বের করতে হবে। মৌলিক সংখ্যা হচ্ছে ১ এর চেয়ে বড় পূর্ণসংখ্যা যা শুধুমাত্র ১ এবং নিজেকে দিয়ে বিভাজ্য।

(<http://bit.ly/1-52-problem-49>)

ইনপুট

ইনপুট ফাইলের প্রথম লাইনে থাকবে টেস্ট কেসের সংখ্যা T ($T \leq 10$), এরপরে T সংখ্যক লাইন থাকবে যাদের প্রতিটিতে একটি করে পূর্ণসংখ্যা N ($2 \leq N \leq 1000000000000$) থাকবে।

আউটপুট

প্রতিটি টেস্ট কেসের জন্য, যদি N মৌলিক হয়, প্রথমে প্রিন্ট করবে N , তারপরে "is a prime" স্ট্রিংটি কোনো কোটেশন ছাড়া প্রিন্ট করবে। N মৌলিক না হলে প্রথমে প্রিন্ট করবে N , তারপরে "is not a prime" স্ট্রিংটি কোনো কোটেশন ছাড়া প্রিন্ট করবে। নমুনা আউটপুটে আরও বিস্তারিত দেখতে পারো।

উদাহরণ

ইনপুট	আউটপুট
3	2 is a prime
2	6 is not a prime
6	
11	11 is a prime

সমাধান

তোমরা ইতিমধ্যে কোনো সংখ্যা মৌলিক সংখ্যা কি না, সেটি বের করা শিখে গেছ নিশ্চয়ই। তোমরা প্রথমে করবে কী, এতক্ষণ পর্যন্ত যতটুকু জানো, সেই জ্ঞান প্রয়োগ করে এই সমস্যাটির সমাধান করে ফেলো। খুব সন্তুষ্ট তোমাদের বেশিরভাগেরই সমাধানটি অনলাইন জাজে টাইম লিমিট এর দিবে। মানে আরও ইফিশিয়েন্ট সমাধান দরকার।

এই সমস্যার মূল চ্যালেঞ্জ হচ্ছে, যে সংখ্যাটি মৌলিক কি না বের করতে হবে, তার সর্বোচ্চ মান হতে পারে 1000000000000 । তাই আমাদের কিছু জ্ঞান-বৃদ্ধি প্রয়োগ করতে হবে। তোমাদের যদি "কম্পিউটার প্রোগ্রামিং ১ম খণ্ড" বইটি পড়া না থাকে, তাহলে তোমরা

<http://cpbook.subeen.com> ওয়েবসাইটে গিয়ে মৌলিক সংখ্যা অধ্যায়টি পড়ে নিতে পারো, সেখানে বিস্তারিত আলোচনা আছে। তাহলে আমরা সমাধানের দিকে কীভাবে অগ্রসর হব এবং কোন কাজটি কেন করছি, সেটি বুঝতে সহজ হবে।

1000000000000 -এর বর্গমূল হচ্ছে 1000000। তাই আমরা একটি অ্যারেতে 1000000 পর্যন্ত সব মৌলিক সংখ্যা বের করে রেখে দেব। আর এই কাজটি করার জন্য আমরা সিভ মেথডের সাহায্য নেব। সিভ মেথডের বিস্তারিত "কম্পিউটার প্রোগ্রামিং - ১ম খণ্ড" বইতে আছে।

তারপর আমরা ইনপুট নেওয়া শুরু করব আর তখন সংখ্যাটি মৌলিক কি না, সেটি পরীক্ষা করে আউটপুট প্রিন্ট করব। প্রোগ্রামের গঠন এ রকম হবে:

```
#include<stdio.h>
#include<math.h>

void find_prime_number(int prime_ara[], int n)
{
    // write code here
}

int main()
{
    int prime_ara[]; // এখানে যথাযথভাবে অ্যারের সাইজ উল্লেখ করতে হবে।
    int T, sqrt_n;
    long long int N;

    scanf("%d", &T);

    while(T--)
    {
        scanf("%lld", &N);
        sqrt_n = sqrt(N);
        find_prime_number(prime_ara, sqrt_n);
        is_prime = 1;
        for(i = 0; prime_ara[i] <= sqrt_n; i++)
        {
            if (N % prime_ara[i] == 0)
            {
                is_prime = 0;
                break;
            }
        }

        if (is_prime == 1)
```

```

    {
        printf("%lld is a prime\n", N);
    }
    else
    {
        printf("%lld is not a prime\n", N);
    }
}

return 0;
}

```

সমস্যা ৫০ – লেফট-রাইট

সমস্যার বিবরণ

একটি স্ট্রিং দেওয়া থাকবে। স্ট্রিংটি কিছু দশমিক ডিজিট এবং 'L', 'R' এ দুইটি ক্যারেক্টার দিয়ে গঠিত। স্ট্রিংটির যেসব অবস্থানে 'L' পাওয়া যাবে সেগুলোকে তার ঠিক বামের ক্যারেক্টার দিয়ে বদলে ফেলতে হবে এবং যেসব স্থানে 'R' পাওয়া যাবে সেগুলোকে বদলে ফেলতে হবে তার ডানের ক্যারেক্টার দিয়ে। অর্থাৎ, ইনপুট 34R92L6 থাকলে হয়ে যাবে 3499226।

(<http://bit.ly/1-52-problem-50>)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। T-এর মান যত, এর পরে ততগুলো লাইনে একটি করে ইনপুট স্ট্রিং থাকবে। স্ট্রিংটি শুরু হবে একটি ডিজিট দিয়ে এবং শেষ হবে একটি ডিজিট দিয়ে। মধ্যবর্তী কোনো স্থানে ডিজিট ব্যতীত অন্য কোনো ক্যারেক্টার পাশাপাশি থাকবে না। প্রতিটি ইনপুট স্ট্রিং এর দৈর্ঘ্য 50 বা তার কম হবে।

আউটপুট

প্রতিলাইনের জন্য সেই লাইনে দেওয়া স্ট্রিংটিকে নিয়ম অনুযায়ী পরিবর্তন করলে যে নতুন স্ট্রিং পাওয়া যাবে সেটি প্রিন্ট করতে হবে।

উদাহরণ

ইনপুট	আউটপুট
5	007
0L7	45559
4R5L9	71
71	84400
8R4R0	3499226
34R92L6	

সমাধান

এই সমস্যায় দুটি শর্তের কথা বলা আছে। একটি হচ্ছে স্ট্রিংয়ে L পেলে কী করতে হবে আর R পেলে কী করতে হবে। যদি সি ভাষায় সেটি অনুবাদ করি, তাহলে মূল কাজ দাঁড়ায় :

```
if(ara[i] == 'L') ara[i] = ara[i-1];
if(ara[i] == 'R') ara[i] = ara[i+1];
```

এখন তোমরা একটি লুপ চালিয়ে পুরো সমাধানটি লিখে ফেলো। প্রথম প্রচেষ্টায় অনেকের সমাধান ভুল হওয়ার সম্ভাবনা আছে। সেক্ষেত্রে তোমাদের লক্ষ করতে হবে যে য্যারের ইনডেক্স ব্যবহার করার সময় কোথাও গঙ্গোল করেছ কি না।

সমস্যা ৫১ – খুঁজ দ্য সার্ট ১

সমস্যার বিবরণ

দুটি স্ট্রিং দেওয়া থাকবে যার দ্বিতীয়টি প্রথমটির সাবস্ট্রিং। খুঁজে বের করতে হবে প্রথমটিতে সাবস্ট্রিংটি সর্বপ্রথম কোথা থেকে শুরু হয়েছে। উল্লেখ্য, কোনো স্ট্রিংের একটানা কোনো অংশকে বলে তার সাবস্ট্রিং। যেমন, banana এর একটা সাবস্ট্রিং ana এবং এটা সর্ব প্রথম শুরু হয়েছে

1 তম স্থান থেকে। আরেকটি সাবস্ট্রিং ban যা শুরু হয়েছে 0 তম স্থান থেকে। ওদিকে anna প্রথম স্ট্রিংটির বৈধ সাবস্ট্রিং নয়।
[\(http://bit.ly/1-52-problem-51\)](http://bit.ly/1-52-problem-51)

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। T-এর মান যত, এর পরে ততগুলো লাইনে দুটি করে স্ট্রিং থাকবে। প্রতিটি স্ট্রিংয়ের দৈর্ঘ্য 128 এর কম এবং স্ট্রিং দুটি একটি স্পেস দিয়ে আলাদা।

আউটপুট

প্রতিলাইনের জন্য সেই লাইনের দ্বিতীয় স্ট্রিংটি প্রথম স্ট্রিংতে সর্বপ্রথম কোথা থেকে শুরু হয়েছে তা বলতে হবে।

উদাহরণ

ইনপুট	আউটপুট
4	1
banana ana	0
banana ban	11
aquickbrownfoxjumpsoverthelazydog fox	0
foobar foobar	

সমাধান

বেশ সহজ একটি সমস্যা, সমাধানও সহজ। ধরা যাক তোমরা প্রথম স্ট্রিং ইনপুট নিছ str1 আর দ্বিতীয় স্ট্রিং ইনপুট নিছ str2-তে। এখন str2 এর প্রথম ক্যারেক্টর, অর্থাৎ str2[0]-কে str1-এর মধ্যে খোঁজা শুরু করবে। যদি খুঁজে পাও, তাহলে পরবর্তী ক্যারেক্টরগুলো মিলে কি না, সেটি পরীক্ষা করতে হবে। যদি মিলে যায়, তখনই ঘোষণা করে দিবে যে দ্বিতীয় স্ট্রিংটি প্রথম স্ট্রিংতে সর্বপ্রথম কোথা থেকে শুরু হয়েছে, আর খোঁজাখুঁজি বন্ধ করবে।

```

str1_len = strlen(str1);
str2_len = strlen(str2);

for (i = 0; i < (str1_len - str2_len); i++)
{
    if (str2[0] == str1[i])
    {
        for (j = 1; j < str2_len; j++)
    }
}

```

```

    {
        if (str2[j] != str1[i+j]) break;
    }
    if (j == str2_len)
    {
        printf("%d\n", i);
        break;
    }
}

```

ওপরের কোড আমি প্রতিটি লাইন ধরে ধরে ব্যাখ্যা করলাম না। তোমাদের বুঝতে একটু সময় লাগতে পারে তবে মনোযোগ ও সময় দিলে অবশ্যই বুঝতে পারবে। না বুঝে কোড লিখে সেটা অনলাইন জাজে অ্যাকসেপ্ট করিয়ে কোনো লাভ হবে না, তাতে তুমি আজীবনের জন্য ঠকে যাবে।

সমস্যা ৫২ – রেঞ্জ দ্য সার্চ ২

সমস্যার বিবরণ

দুটি স্ট্রিং দেওয়া থাকবে যার দ্বিতীয়টি প্রথমটির সাবস্ট্রিং। খুঁজে বের করতে হবে প্রথমটিতে সাবস্ট্রিংটি কতবার আছে। উল্লেখ্য, কোনো স্ট্রিংয়ের একটানা কোনো অংশকে বলে তার সাবস্ট্রিং। যেমন, banana এর একটা সাবস্ট্রিং ana এবং এটা সর্ব প্রথম শুরু হয়েছে 1-তম স্থান থেকে, আবার 3-তম স্থান থেকেও এটি আরেকবার আছে।

```

banana
↓↓↓↓↓
ana↓↓
  ↓↓↓
ana

```

অর্থাৎ, মোট 2 বার সাবস্ট্রিংটিকে পাওয়া যাচ্ছে। ওদিকে anna প্রথম শব্দটির কোনো বৈধ সাবস্ট্রিংই নয়। তাই পাওয়া যাচ্ছে 0 বার।

ইনপুট

প্রথম লাইনে একটি সংখ্যা T ($1 \leq T \leq 100$) থাকবে। T -এর মান যত, এর পরে ততগুলো লাইনে দুটি করে স্ট্রিং থাকবে। প্রতিটি স্ট্রিংয়ের দৈর্ঘ্য 128-এর কম এবং স্ট্রিং দুটি একটি স্পেস দিয়ে আলাদা।

আউটপুট

প্রতিলাইনের জন্য সেই লাইনের দ্বিতীয় স্ট্রিংটি প্রথম স্ট্রিংটির মধ্যে কতবার আছে তা বলতে হবে।

উদাহরণ

ইনপুট	আউটপুট
5	2
banana ana	0
banana anna	1
aquickbrownfoxjumpsoverthelazydog fox	3
ddddd ddd	1
foobar foobar	

সমাধান

আমরা এখন বইয়ের একেবারে শেষ সমস্যায় চলে এসেছি। তোমরা নিচয়ই এতক্ষণে প্রোগ্রামিং সমস্যা সমাধানের মজা পেয়ে গিয়েছ। আশা করি, অনেকেই এই সমস্যার সমাধানও নিজে নিজে করে ফেলেছ।

খোঁজ দ্য সার্চ ১ সমস্যাটির সমাধান যদি তুমি করে থাক, তাহলে এটি তোমার জন্য সহজ হবে বৈকি। আর ওই সমস্যার সমাধান করা না থাকলে আমার পরামর্শ হবে ওটি আগে সমাধান করা।

খোঁজ দ্য সার্চ ১-এর সমাধান করার সময়, আমরা যখনই সার্চিং পেয়ে গিয়েছিলাম, আমরা লুপ থেকে ব্রেক করেছি। কারণ আমাদের দরকার ছিল সার্চিং পাওয়া গেলে সেটি প্রথম স্ট্রিংয়ের কোন ঘর থেকে শুরু হয়েছিল। আর এখন আমাদের বের করতে হবে যে সার্চিংটি কতবার পাওয়া গেল। তাই একটি count ভেরিয়েবল ব্যবহার করতে হবে (int টাইপের) যার মান প্রথমে 0 করে দেব। সার্চিং পাওয়া গেলে আমরা ভ্যারিয়েবলটির মান এক বাড়াব কিন্তু লুপ থেকে ব্রেক করব না। লুপটি যখন নিজে নিজে বন্ধ হবে, তখন আমরা count-এর মান প্রিন্ট করব। আশা করি প্রোগ্রামটি নিজে নিজে লিখতে পারবে। কারও কারও ক্ষেত্রে count-এর মান কোন জায়গায় শূন্য সেট করবে সেটা বের করতে একটু সমস্যা হতে পারে, কিন্তু লেগে থাকলে সমাধান হবেই!

পরিশিষ্ট

অনলাইন জাজ কী এবং কেনো?

গণিত অলিম্পিয়াড বা এই জাতীয় প্রতিযোগিতার সঙ্গে প্রোগ্রামিং প্রতিযোগিতার একটা বড় পার্থক্য হচ্ছে অন্যান্য সব প্রতিযোগিতার ক্ষেত্রেই সাধারণত উভরপত্র যাচাই-বাছাই ও মূল্যায়ন করা হয়ে থাকে শিক্ষকদের দিয়ে। কিন্তু প্রোগ্রামিং প্রতিযোগিতার ক্ষেত্রে একটি উভর সঠিক হলো কি না, তা যাচাই করা হয় অন্য আরেকটি কম্পিউটার প্রোগ্রাম দিয়ে। আর এই প্রোগ্রামকেই বলা হয় অনলাইন জাজ। যিনি প্রোগ্রামিং প্রবলেম বা সমস্যাটি তৈরি করেন, তাকে বলা হয় প্রবলেম সেটার। সাধারণত প্রবলেম সেটার নিজের তৈরিকৃত সমস্যার জন্য একটি ইনপুট সেট তৈরি করেন (হাতে কলমে বা কম্পিউটার প্রোগ্রাম লিখে), তারপর তিনি সেই সমস্যার একটি সমাধান প্রোগ্রাম লিখেন। এরপরে, আগে তৈরি করা ইনপুটের জন্য নিজের সমাধান প্রোগ্রাম চালিয়ে আউটপুট তৈরি করেন। কখনো কখনো একাধিক ইনপুট-আউটপুট সেটও তৈরি করা হয়। এরপর, অনলাইন জাজে নিজের প্রোগ্রামিং সমস্যা ও তার ইনপুট আউটপুট ফাইলসহ জমাদান (submit) করেন। যখন কোনো প্রতিযোগী একটি সমস্যার সমাধান করে অনলাইন জাজে জমাদান (submit) করেন, তখন অনলাইন জাজ সফটওয়্যারটি প্রথমে নিজস্ব কম্পাইলার ব্যবহার কোডটির এক্সিকিউটিভেল ফাইল তৈরি করে। পরে সেই প্রোগ্রামটি চালু করে প্রবলেম সেটারের তৈরি ইনপুট ফাইল থেকে ইনপুট প্রদান করে। প্রোগ্রামটি চলার পরে যে আউটপুট তৈরি হয়, সেটাকে প্রবলেম সেটারের জমা দেওয়া আউটপুট ফাইলের সঙ্গে তুলনা করে যদি হুবহু মিলে যায়, তবেই কেবল অনলাইন জাজ সিদ্ধান্ত দেয় যে প্রোগ্রামটি সঠিক হয়েছে।

এখন তোমাদের মনে প্রশ্ন জাগতে পারে, পরীক্ষার খাতা দেখার জন্য আবার কম্পিউটার প্রোগ্রাম লেখার দরকার কী ছিল? শিক্ষকরা নিজেরাই তো মূল্যায়ন করতে পারেন। অনলাইন জাজের একটি বড় সুবিধা হচ্ছে, একটি সমাধান যাচাইয়ের এই পুরো কাজটা করতে সাধারণত কয়েক সেকেন্ডের বেশি সময় লাগে না। ফলে একটি প্রতিযোগিতায় যদি হাজার হাজার পরীক্ষার্থীও অংশগ্রহণ করে সমস্যার সমাধান জমা দিতে থাকে তাহলেও মোটামুটিভাবে সঙ্গে সঙ্গেই ফলাফল ও র্যাঙ্কিংস্ট তৈরি করা যায়। আর অনলাইন জাজের একটি অসুবিধাও রয়েছে, যেহেতু পুরো প্রক্রিয়াটি কম্পিউটারে করা হয়, তাই তোমার আউটপুট যদি পুরোটা ঠিক হয়, অথচ, সামান্য একটু ভুল হয়, তাতেও কম্পিউটার তোমার পুরো সমাধানকেই ভুল বলে ফলাফল দেবে এবং কোনো আংশিক নম্বর দিবে না। (যদিও কিছু কিছু প্রতিযোগিতায় আংশিক সঠিক ফলাফলের জন্য আংশিক নম্বর দেওয়া হয়)।

অনলাইন জাজে সমাধান সাবমিট করার পরে কী কী ধরনের ফলাফল আসছে পারে তার একটা বিবরন এখানে দেওয়া হলো:

১। রং অ্যানসোর (Wrong Answer) – অনলাইন জাজে সাবমিট করা সমাধানের আউটপুট যদি প্রবলেম সেটারের আউটপুটের সঙ্গে না মিলে তাহলে এই ধরনের উত্তর আসে। পরে এ সংক্রান্ত বিস্তারিত আলোচনা করা হয়েছে।

২। টাইম লিমিট এক্সেডেড (Time Limit Exceeded) – সাধারণত প্রবলেম সেটার প্রতিটি প্রবলেমের জন্য একটি নির্দিষ্ট সময়সীমা বেঁধে দেন। অর্থাৎ, সমাধানটি এমন হতে হবে যেন, ওই নির্দিষ্ট সময়ের মধ্যেই প্রোগ্রামটি ইনপুট নিয়ে প্রয়োজনীয় আউটপুট তৈরি করবে। যদি প্রোগ্রামটি চলতে এর চেয়ে বেশি সময় নেয়, তখন অনলাইন জাজ এই ধরনের ফলাফল তৈরি করে। এক্ষেত্রে বুঝতে হবে, কোডটিতে যেই অ্যালগরিদম ব্যবহার করা হয়েছে, তার থেকেও দ্রুততর কোনো অ্যালগরিদম ব্যবহার করতে হবে অথবা সমস্যাটি ভিন্নভাবে সমাধান করতে হবে।

৩। মেমোরি লিমিট এক্সেডেড (Memory Limit Exceeded) – কিছু কিছু সমস্যার ক্ষেত্রে সমাধানটি মেমোরি বা RAM-এ কত বাইট জায়গা ব্যবহার করতে পারবে তার ওপরেও একটা সীমা বেঁধে দেওয়া হয়। যদি সমাধানটি তারচেয়ে বেশি মেমোরি ব্যবহার করে তখন সাধারণত এই ধরনের ফলাফল আসে।

৪। প্রেজেন্টেশন এরর (Presentation Error) – এই ফলাফলের অর্থ হচ্ছে সমাধানের উত্তরগুলো সঠিক হয়েছে, তবে উদাহরণের ঠিক যেই ফরম্যাট বা প্যাটার্নে আউটপুট দিতে বলা হয়েছে, তা অনুসরন করা হয়নি। অর্থাৎ, কোথাও হয়তো প্রয়োজনের চেয়ে বেশি বা কম স্পেস, নিউলাইন ('\\n') ইত্যাদি ক্যারেক্টার প্রিন্ট করা হয়েছে।

৫। কম্পাইলেশন এরর (Compilation Error) – যদি সমাধানটি অনলাইন জাজ সঠিকভাবে কম্পাইল করতে না পারে, তখন এ ধরনের এরর দেওয়া হয়। এ রকম এরর পেলে বুঝতে হবে, কোডের কোথাও কোনো ত্রুটি রয়েছে, যেমন: সেমিকোলন-কমা ঠিক মতো না দেওয়া বা বিভিন্ন রকম বন্ধনী সঠিকভাবে বন্ধ না করা, ভ্যারিয়েবল ডিক্লেয়ার না করেই ব্যবহার করা ইত্যাদি।

৬। রানটাইম এরর (Runtime Error) – যদি সাবমিট করা সমাধানটি চলতে গিয়ে কোনো রকম এরর বা এক্সেপশন তৈরি করে বন্ধ হয়ে যায়, তখন এই ধরনের ফলাফল দেওয়া হয়। কখনো কখনো অ্যারের ইনডেক্স ব্যবহারে ভুল হলে বা ভুল অ্যাড্রেসে পয়েন্টার ব্যবহার করলে এই ধরনের ত্রুটি বা এরর হতে পারে।

৭। রেস্ট্রিটেড ফাংশন (Restricted Function) – অনেক ক্ষেত্রে অনলাইন জাজে বিশেষ কিছু ফাংশন ব্যবহারে নিষেধাজ্ঞা থাকে। সাবমিটকৃত সমাধান যদি ওইসব ফাংশন ব্যবহার করে তখন এ ধরনের ফলাফল দেওয়া হয়।

৮। আউটপুট লিমিট এক্সেডেড (Output Limit Exceeded) – যদি জমা দেওয়া সমাধানটি চলার সময় অতিরিক্ত আউটপুট দিতে থাকে, তখন এই ফলাফল দেওয়া হয়। ধরা যাক,

কেউ যদি ভুলক্রমে একটি ইনফিনিট লুপ ব্যবহার করে আউটপুট দিতে থাকে তখন এই ফলাফল দেখা যাবে।

৮। **অ্যাক্সেপ্টেড (Accepted)** – যদি সাবমিট করা সমাধানটি চালানোর পরে উপরের কোনো ক্রটি না হয় এবং তার তৈরিকৃত আউটপুট প্রবলেম সেটারের আউটপুটের সঙ্গে হ্রাস মিলে যায় তখন এই ফলাফল-দেওয়া হয়। এর অর্থ হচ্ছে, জমা দেওয়া সমাধানটি সম্পূর্ণ সঠিক হয়েছে।

ৱং অ্যানসার (Wrong Answer)

যারা বিভিন্ন প্রোগ্রামিং সমস্যা সমাধানের ওয়েবসাইটে কেবল চৰ্চা করা শুরু করেছ, তাদের জন্য সবচেয়ে সাধারণ ব্যাপার হচ্ছে এই রং অ্যানসার (Wrong Answer)। “স্যাম্পল ইনপুট-আউটপুটের সঙ্গে তো আমার মিলে, তাহলে অনলাইন জাজ কেন রং অ্যানসার দিচ্ছে”?

নমুনা ইনপুট-আউটপুট (Sample Input-Output) মিললেই হবে না। ধরা যাক, কোনো একটা মৌলিক সংখ্যা বের করার প্রোগ্রামে নমুনা ইনপুট দেওয়া আছে 3, 5, 7 যার জন্য আউটপুট হচ্ছে প্রাইম। এখন এটা দেখে কেউ যদি এমন প্রোগ্রাম লিখে যে বিজোড় সংখ্যা মাত্রই প্রাইম, তাহলে তো হবে না, তাই না? তাই নিজে নিজে টেস্ট কেইস তৈরি করে আগে নিজে বের করতে হবে যে আউটপুট কী হবে। তারপরে প্রোগ্রামের আউটপুটের সঙ্গে মিলিয়ে দেখবে।

আরেকটা ব্যাপার হচ্ছে আউটপুট ফরম্যাটিং। প্রতিটা টেস্ট কেইসের শেষে নিউলাইন ('\\n') দিতে হবে, যেটা আসলে সবসময় বলে দেওয়া থাকে না। আর দেখতে হবে নমুনা আউটপুটের সঙ্গে যেন হ্বহ্ব মিলে। ছোটহাত-বড়হাতের অক্ষর (Uppercase / lowercase) এবং স্পেস ক্যারেক্টার ঠিকমতো মিলতে হবে। যেমন প্রবলেমের আউটপুটে যদি Case লিখতে বলে আর তুমি case লিখ, তাহলে হবে না। আবার Case 1: লিখতে বললে তুমি যদি Case 1 : লিখ (: এর আগে একটি স্পেস), তাহলেও রং অ্যানসার।

কর্ণার কেস টেস্ট করতে হবে। ধরা যাক তোমাকে বলা হলো, a^n এর মান বের করতে। এখন তুমি চট করে প্রোগ্রাম লিখে ফেললে যেটা খুব সহজেই কাজটি করে দেয়। স্যাম্পল আউটপুট তো মিলেই, তুমি নিজেও যেসব ইনপুট-আউটপুট তৈরি করে টেস্ট করলে, সেগুলোও মিলে যায়। কিন্তু এখানে n -এর মান 0 হলে কী হবে? আর ঝগত্তাক হলেই বা কী হবে? এগুলোও কিন্তু তোমার প্রোগ্রামে ঠিকঠাক আউটপুট দিতে হবে।

আরেকটি কারণ হচ্ছে সমস্যার বর্ণনা ঠিকমতো বুঝতে না পারা। আমরা অনেক সময় একবার বা দুবার পড়ে ঠিকঠাক বুঝতে পারি না। তাই রং অ্যানসার হলে সমস্যার বর্ণনা আবার মনোযোগ দিয়ে পড়তে হবে। অনেক সময় দেখবে তখন তুমি উপলব্ধি করবে যে প্রোগ্রাম লিখার সময় তুমি সমস্যাটি ঠিকঠাক বুঝতে পারোনি।

অনেক সময় ফ্রোটিং পয়েন্ট সংখ্যা নিয়ে হিসাবের সময় ঝামেলা হয়। সবকিছু ঠিকঠাক থাকলেও ফ্রোটিং পয়েন্ট ক্যালকুলেশনের কারণে তুমি রং অ্যানসার পেতে পারো। তাই এই কাজটি ঠিকঠাক করতে হবে। একটি উদাহরণ দিচ্ছি (পাইথন কোড):

>>> 2.1 - 1.9
০.২০০০০০০০০০০০১৮

তাই ফ্লোটিং পয়েন্ট নিয়ে কাজ করার সময় সাবধান!

আরেকটা কারণ হচ্ছে ভুল অ্যালগরিদমের ব্যবহার। এটাও আসলে সমস্যার বর্ণনা ঠিকভাবে বুঝতে না পারার ফলেই ঘটে থাকে।

এসব সমস্যা থেকে পরিত্রাণ পাওয়ার উপায় হচ্ছে রং অ্যানসার পেলে সেটা যে তোমারই ভুল, এটা স্বীকার করে নেওয়া (মাঝে-মধ্যে জাজেরও ভুল হতে পারে, তবে সেটা খুবই কম ঘটে)। নিজে আরও চিন্তাভাবনা করা, না পারলে বঙ্গুদের সঙে সেটা নিয়ে আলাপ করা। তাতেও কাজ না হলে অনলাইন ফোরাম ঘেঁটে দেখা যে আগে কেউ এ রকম সমস্যায় পরেছে কি না এবং সেটা কীভাবে সমাধান হয়েছে।

প্রোগ্রামিং চর্চার কয়েকটি অনলাইন জাজ

অনেক ওয়েবসাইট আছে, যেখানে অনেক প্রোগ্রামিং সমস্যা দেওয়া আছে এবং সেগুলোর সমাধান করে জমা দিলে স্বয়ংক্রিয়ভাবে সেটা জাজ করে জানিয়ে দেওয়া হয় যে সমাধানটি সঠিক হয়েছে কি না। প্রোগ্রামিং শিখতে যেহেতু চর্চা করাটা সবচেয়ে জরুরি, তাই অনলাইন জাজগুলো অনেক গুরুত্বপূর্ণ। এ রকম বেশকিছু অনলাইন জাজ ওয়েবসাইটের লিঙ্ক দিয়ে দিচ্ছি, এর বাইরেও আরও অনেক আছে।

প্রোগ্রামিং চর্চার জন্য অনলাইন জাজ:

- 1) http://cpbook.subeen.com/p/blog-page_11.html: এখানে বাংলায় বেশ কিছু প্রোগ্রামিং সমস্যা দেওয়া আছে, যা প্রোগ্রামিংয়ে একেবারে যারা নতুন, তাদের জন্য ভালো।
- 2) <http://lightoj.com>: এটি তৈরি করেছেন বাংলাদেশের জানে আলম জান। এখানে অভিজ্ঞ প্রোগ্রামারদের সঙ্গে সঙ্গে বিগিনারদের জন্যও কিছু সমস্যা দেওয়া আছে।
- 3) <http://projecteuler.net>: এখানে অনেক মজার সমস্যা আছে যেগুলোর বেশিরভাগই প্রোগ্রাম লিখে সমাধান করতে হয়। এখানে প্রোগ্রাম জমা দেওয়া লাগে না, কেবল প্রোগ্রাম দিয়ে বের করা উত্তরটা জমা দিতে হয়।
- 4) <http://train.usaco.org/usacogate>: এটি যদিও আমেরিকার ইনফরমেটিক্স অলিম্পিয়াড ট্রেনিং প্রোগ্রাম, কিন্তু সাইটে যেকোনো দেশের প্রোগ্রামাররাই রেজিস্ট্রেশন করে অনুশীলন করতে পারে। তোমরা যারা প্রোগ্রামিং প্রতিযোগিতায় ভালো করতে চাও, তাদের অবশ্যই এখানে অনুশীলন করা উচিত।
- 5) <https://www.topcoder.com/>: এখানেও নিয়মিত অনলাইন প্রোগ্রামিং প্রতিযোগিতা অনুষ্ঠিত হয়। এখানে ভালো ফলাফল করলে আবার টাকাও দেয় (কী আনন্দ!)। এ ছাড়া এখানে অনেক ভালো টিউটোরিয়াল ও আর্টিকেল আছে। এটি অভিজ্ঞ প্রোগ্রামারদের জন্য বেশ ভালো একটি সাইট।
- 6) <https://uva.onlinejudge.org/index.php>: এই সাইটে নিয়মিত অনলাইন প্রোগ্রামিং প্রতিযোগিতার আয়োজন করা হয়। এছাড়াও অনুশীলনের জন্য প্রচুর সমস্যা দেওয়া আছে। নতুন প্রোগ্রামারদের জন্য এটি বেশ ভালো জায়গা। তবে একই ধরনের অনেক প্রবলেম আছে। তাই খেয়াল করতে হবে যেন একই রকম অনেক প্রবলেম সলভ করে সময় নষ্ট না হয়।
- 7) <http://www.spoj.com/>: এখানেও অনেক ভালো সমস্যা আছে। সমাধান করে প্রোগ্রাম জমা দিলে প্রোগ্রাম সঠিক হয়েছে কি না তা জানা যায়। এই ওয়েবসাইটের একটি বৈশিষ্ট্য হচ্ছে

সি, সি প্লাস প্লাস, জাভা, পার্স, পাইথন, ক্লবি, পিএইচপি এ রকম আরও অনেক ল্যাঙুয়েজ ব্যবহার করে প্রোগ্রাম লেখা যায়।

৮) <http://codeforces.com/>: এই সাইটে নিয়মিত বিভিন্ন ধরনের প্রোগ্রামিং কন্টেস্ট হয়। অভিজ্ঞ প্রোগ্রামারদের জন্য ভালো।

৯) <https://www.codechef.com/>: এটিও প্রোগ্রামিং প্রতিযোগিতার জন্য একটি ভালো ওয়েবসাইট এবং অভিজ্ঞ প্রোগ্রামারদের জন্য।

১০) <http://acm.timus.ru/>: এখানে চমৎকার কিছু প্রবলেম আছে, বেশিরভাগই বিভিন্ন রাশিয়ান কন্টেস্টের।

১১) <http://www.hsin.hr/coci/>: এটি ক্রোয়েশিয়ান ইনফরমেটিক্স অলিম্পিয়াডের ওয়েবসাইট। এখানে মাঝে মধ্যে প্রোগ্রামিং প্রতিযোগিতা আয়োজন করা হয়ে থাকে।

১২) <http://ipsc.ksp.sk/>: এটি হচ্ছে IPSC (Internet Problem Solving Contest) এর ওয়েবসাইট, এখানে প্রতিবছর বিভিন্ন বৈচিত্রময় বিষয়ের ওপরে প্রতিযোগিতা আয়োজিত হয়।

১৩) কিছু চীনা অনলাইন জাজ: চীনা বেশ কিছু অনলাইন জাজ রয়েছে:

http://acm.pku.edu.cn/icpc_pku2015/,

<http://acm.zju.edu.cn/onlinejudge/>,

<http://acm.tju.edu.cn/toj/>

এই সাইটগুলোতে বেশ কিছু প্রোগ্রামিং সমস্যা রয়েছে। এই সাইটগুলোতেও নিয়মিত প্রতিযোগিতার আয়োজন করা হয়।

বইটি কাদের জন্য

দক্ষ প্রোগ্রামার হওয়ার সহজ উপায় হচ্ছে প্রোগ্রামিং সমস্যা সমাধানের চর্চা করা। তাই যারা নতুন প্রোগ্রামিং শিখছে, এই বইতে তাদের জন্য উপযোগী ৫২টি প্রোগ্রামিং সমস্যা দেওয়া হয়েছে এবং সেগুলোর সমাধান নিয়ে আলোচনা করা হয়েছে। সমাধান দেখানোর সময় সি প্রোগ্রামিং ভাষা ব্যবহার করা হয়েছে। যারা ভবিষ্যতে বিভিন্ন প্রোগ্রামিং প্রতিযোগিতায় অংশ নিতে চায় - যেমন জাতীয় হাই স্কুল প্রোগ্রামিং প্রতিযোগিতা, ইনফরমেটিক্স অলিম্পিয়াড, এসিএম আইসিপিসি, তারা প্রোগ্রামিং সমস্যা সমাধানের চর্চা এই বই দিয়েই শুরু করতে পারে। এছাড়া বইটি প্রোগ্রামিংয়ের জগতে নতুন যে কারো জন্য উপযোগী হবে।