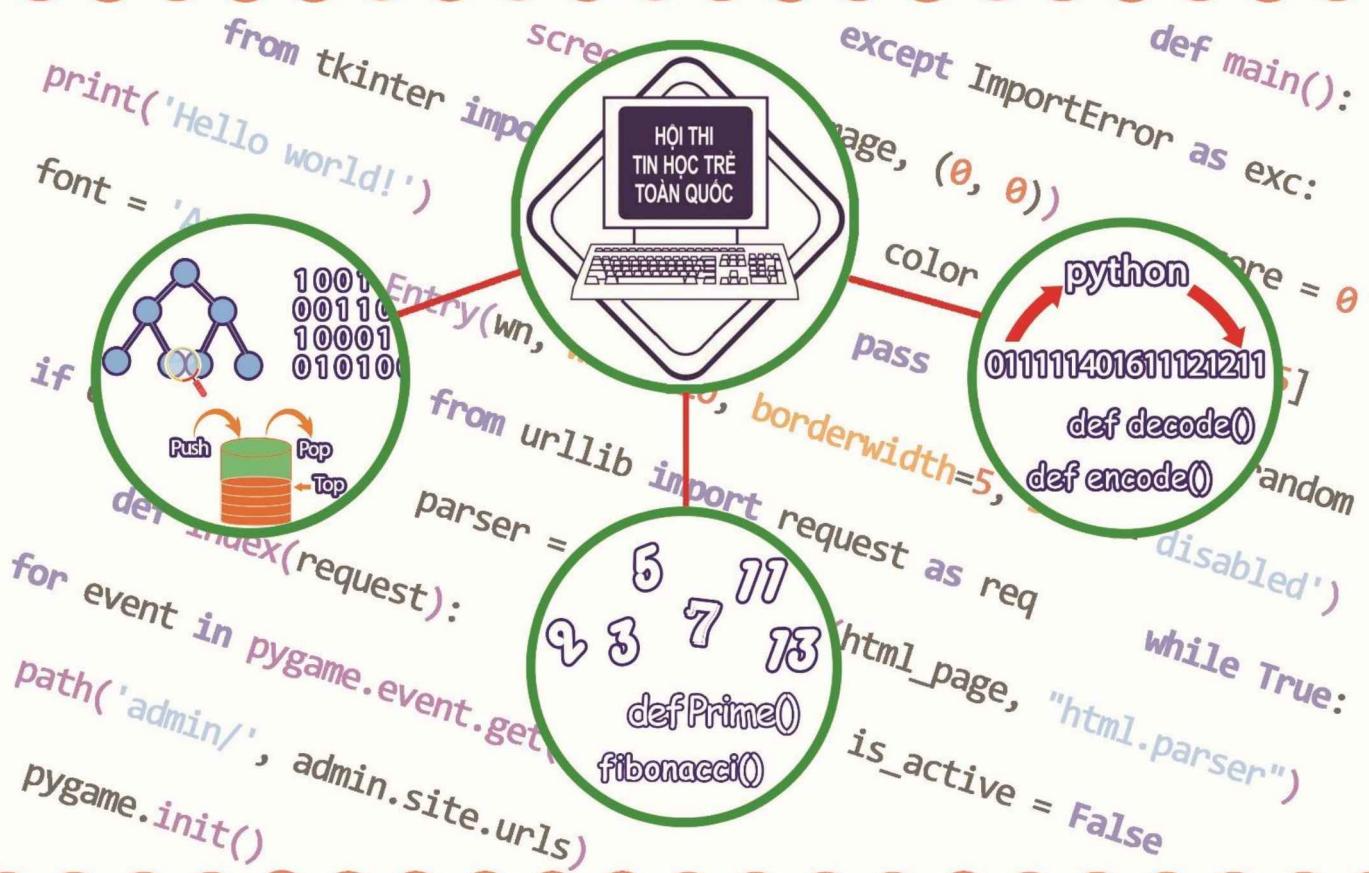


# SÁCH LUYỆN THI HỘI THI TIN HỌC TRẺ VỚI python

BẢNG B: THI KỸ NĂNG LẬP TRÌNH CẤP TRUNG HỌC CƠ SỞ



SÁCH LUYỆN THI  
HỘI THI TIN HỌC TRẺ  
VỚI  
**python**

BẢNG B: THI KỸ NĂNG LẬP TRÌNH CẤP TRUNG HỌC CƠ SỞ



# MỤC LỤC

<b>LỜI NÓI ĐẦU .....</b>	<b>8</b>
<b>CHƯƠNG I. GIỚI THIỆU VỀ HỘI THI TIN HỌC TRẺ .....</b>	<b>10</b>
Giới thiệu về Hội thi .....	10
Nội dung và hình thức .....	11
Đăng ký tham dự .....	14
Giải thưởng .....	15
<b>CHƯƠNG II: GIẢI ĐỀ THEO CÁC DẠNG BÀI .....</b>	<b>16</b>
<b>Dạng bài số học .....</b>	<b>17</b>
Bài 1. Giá trị biểu thức (TP. Vũng Tàu, 2020) .....	17
Bài 2. Tựa bài toán cổ (Bắc Giang, 2020) .....	21
Bài 3. Tính tổng (Yên Bai, 2020) .....	22
Bài 4. Số mạnh mẽ (Hải Dương, 2020) .....	24
Bài 5. Bội chính phương (Quốc gia, 2020) .....	25
Các bài tập thực hành .....	27
<b>Dạng bài xử lý xâu và chuỗi .....</b>	<b>30</b>
Bài 1. Số đảo ngược (Tỉnh Tây Ninh, 2019) .....	30
Bài 2. Nén xâu (Bắc Giang, 2020) .....	30
Bài 3. Tính nhân (Hậu Giang, 2020) .....	32
Bài 4. Đối gương .....	34
Bài 5. Mã hóa và giải mã .....	38
Bài tập thực hành .....	41
<b>Dạng bài mảng một chiều, danh sách .....</b>	<b>45</b>
Bài 1. Đặt hàng (TP. Vũng Tàu, 2020) .....	45
Bài 2. Dãy Fibonacci (Yên Bai, 2020) .....	48
Bài 3. Đếm số nguyên tố (An Giang, 2020) .....	50
Bài 4. Cấp số cộng (Sóc Trăng, 2019) .....	52
Bài 5. Phát khẩu trang (Bình Dương, 2020) .....	55
Bài tập thực hành .....	58
<b>Dạng bài ma trận .....</b>	<b>63</b>
Bài 1. Cách ly (Bình Dương, 2020) .....	64
Bài 2. Xây dựng bệnh viện dã chiến (Hải Phòng, 2020) .....	67
Bài 3. Tứ giác đồng hồ cạnh K (Đông Triều, 2019) .....	71
Bài 4. Ma trận hợp lệ .....	74
Bài tập thực hành .....	76
<b>Dạng bài sắp xếp .....</b>	<b>84</b>
Bài 1. Dãy số (Bắc Giang, 2019) .....	85
Bài 2. Dãy không giảm (Vị Thanh – Hậu Giang, 2019) .....	87
Bài tập thực hành .....	89
<b>Dạng bài sử dụng thuật toán .....</b>	<b>93</b>

Bài 1. Thuật toán đệ quy - RECURSION .....	93
Bài tập thực hành.....	94
Bài 2. Thuật toán tìm kiếm – SEARCH ALGORITHMS.....	97
Bài tập thực hành.....	100
<b>CHƯƠNG III: GIẢI ĐỀ THI CHÍNH THỨC ..... 102</b>	
<b>Đề thi Quốc gia năm 2019 ..... 103</b>	
Bài 1. Trung bình cộng .....	103
Bài 2. Phân số.....	104
<b>Đề thi sơ khảo Quốc gia năm 2020 ..... 108</b>	
Bài 1. Phần thưởng .....	108
Bài 2. Đề thi .....	110
<b>Đề thi Thành phố Hà Nội năm 2020 ..... 113</b>	
Bài 1. SQ.....	113
Bài 2. SX10 .....	114
Bài 3. HCN .....	116
<b>Đề thi thành phố Hồ Chí Minh năm 2020 ..... 119</b>	
Bài 1. Mẫu số chung nhỏ nhất.....	119
Bài 2. Chữ số .....	121
Bài 3. Đổi hướng.....	123
<b>Đề thi Thành phố Đà Nẵng năm 2021..... 127</b>	
Bài 1. Số giàu có .....	127
Bài 2. Dịch cúm .....	128
Bài 3. Cắt dây.....	130
Bài 4. Sắp xếp theo modul K .....	131
<b>Đề thi tỉnh Nghệ An năm 2020 ..... 133</b>	
Bài 1. Số chính phương .....	133
Bài 2. Hình chữ nhật .....	134
Bài 3. Điểm danh.....	135
Bài 4. Đoạn con.....	137
<b>CHƯƠNG IV: THƯ VIỆN ĐỀ THI CHỌN LỌC ..... 140</b>	
Đề thi Quốc gia năm 2018 .....	141
Đề thi sơ khảo Quốc gia vòng tự do năm 2021 .....	144
Đề thi thành phố Hà Nội năm 2021 .....	147
Đề thi Thành phố Hà Nội năm 2019 .....	150
Đề thi Thành phố Đà Nẵng năm 2020 .....	152
Đề thi Thành phố Đà Nẵng năm 2019 .....	154
Đề thi Bắc Giang năm 2021 .....	156
Đề thi Đồng Triều – Quảng Ninh năm 2021.....	158
Đề thi Hà Tĩnh năm 2020 .....	161
Đề thi Quảng Ninh năm 2020 .....	163
Đề thi Vĩnh Phúc năm 2020.....	165
Đề thi Đồng Nai năm 2020 .....	167



Đề thi Thanh Hóa năm 2019 .....	170
Đề thi Đồng Tháp năm 2019 .....	172
Đề thi Bình Dương năm 2021 .....	174
Đề thi Gia Lai năm 2019 .....	178
<b>PHỤ LỤC: THÔNG TIN MỞ RỘNG .....</b>	<b>180</b>
Các cuộc thi lập trình trên thế giới .....	180
Giới thiệu một số website luyện thi trực tuyến .....	184
Giới thiệu cộng đồng lập trình viên trên thế giới và kho mã nguồn .....	185
Mã hóa, BlockChain và Tiền Ảo .....	186

# LỜI NÓI ĐẦU

## Hội thi Tin học trẻ và ngôn ngữ lập trình Python

Hội thi Tin học trẻ do Trung ương Đoàn TNCS Hồ Chí Minh chủ trì, phối hợp với Bộ Khoa học và Công nghệ, Bộ Thông tin và Truyền thông, Bộ Giáo dục và Đào tạo, Đài Truyền hình Việt Nam và Hội Tin học Việt Nam phối hợp tổ chức từ năm 1995. Đây là cuộc thi lâu đời và uy tín nhất về tin học cho học sinh phổ thông, mỗi năm có hàng chục nghìn học sinh trên cả nước tham gia, là sân chơi giúp phát hiện và bồi dưỡng những tài năng trẻ về công nghệ thông tin cho đất nước trong suốt 25 năm qua.

Trong năm 2021, với cú pháp đơn giản, dễ học, dễ hiểu,... Python đã vươn lên trở thành ngôn ngữ lập trình bậc cao được sử dụng phổ biến nhất trên thế giới, đặc biệt trong lĩnh vực trí tuệ nhân tạo, máy học và phân tích dữ liệu. Do vậy, trong những năm qua Python đã được đưa vào các đề thi Hội thi Tin học trẻ nhằm từng bước thay thế những ngôn ngữ lập trình trước kia như Pascal.

## Mục đích của cuốn sách

Cung cấp tới các thầy cô và các bạn học sinh tài liệu hướng dẫn ôn luyện “Hội thi Tin học trẻ” (tập trung vào Bảng B: Thi kỹ năng lập trình cấp Trung học cơ sở).

## Nội dung cuốn sách

Cuốn sách gồm 04 chương:

Chương 1: Giới thiệu thông tin về “Hội thi Tin học trẻ”.

Chương 2: Cung cấp các dạng bài thường gặp trong các đề thi, mỗi dạng bài đều có hướng dẫn chi tiết một số bài mẫu, khái quát cách giải và thư viện bài tập giúp học sinh tự ôn luyện cho từng dạng câu hỏi.

Chương 3: Hướng dẫn giải một số đề thi chính thức của Hội thi Tin học trẻ cấp Quốc gia và một số cuộc thi cấp tỉnh, thành phố tiêu biểu.

Chương 4: Cung cấp thư viện đề thi chọn lọc qua các năm giúp học sinh thực hành tự ôn luyện nâng cao kỹ năng giải đề.

Phụ lục: Giới thiệu một số cuộc thi và bài thi quốc tế dành cho học sinh trung học cơ sở; một số trang web luyện thi trực tuyến và cộng đồng lập trình viên với kho mã nguồn mở trên toàn thế giới.



## **Yêu cầu kiến thức khi sử dụng sách**

Để sử dụng sách hiệu quả, người học cần có kiến thức cơ bản về lập trình và ngôn ngữ lập trình Python. Tìm hiểu thêm trong cuốn “Lập trình với Python” (Học viện VIETSTEM, NXB Đại học Quốc gia Hà Nội).

## **Đối tượng sử dụng sách**

Học sinh muốn nâng cao về lập trình Python.

Học sinh cũng như giáo viên hướng dẫn ôn luyện “Hội thi Tin học trẻ toàn quốc - Bảng B: Thi kỹ năng lập trình cấp Trung học cơ sở”.

## **Lời cảm ơn**

Chúng tôi xin chân thành cảm ơn cộng đồng giáo viên tin học, các bạn học sinh đã luôn ủng hộ VIETSTEM và đội ngũ tác giả. Những phản hồi tích cực trong thời gian qua là động lực để chúng tôi tiếp tục hành trình tạo ra những sản phẩm ngày một sáng tạo và chất lượng, giúp phổ biến lập trình cũng như phổ biến giáo dục STEM đến mọi học sinh Việt Nam, góp phần tạo nên một thế hệ trẻ yêu công nghệ.

Để có thêm thông tin và tài liệu hỗ trợ, các bạn có thể truy cập vào mục Trợ giúp và tải tài liệu trên website: <https://vietstem.com/>.

## Giới thiệu về Hội thi Tin học trẻ

### Giới thiệu về Hội thi

Hội thi Tin học trẻ do Trung ương Đoàn TNCS Hồ Chí Minh, Bộ Khoa học và Công nghệ, Bộ Giáo dục và Đào tạo, Bộ Thông tin và Truyền thông, Đài Truyền hình Việt Nam và Hội Tin học Việt Nam phối hợp tổ chức từ năm 1995 nhằm phát hiện các tài năng tin học trẻ, đồng thời hình thành phong trào thanh thiếu niên cả nước học tập, làm chủ công nghệ thông tin, góp phần thiết thực phát triển đội ngũ nhân lực, nhân tài trẻ của đất nước.

Qua 25 năm tổ chức, Hội thi đã trở thành hoạt động truyền thống trong lĩnh vực công nghệ thông tin dành cho thanh thiếu nhi, mỗi năm thu hút hàng trăm nghìn học sinh các cấp trong cả nước tham dự.

Hội thi nhằm tuyên truyền, vận động và phát huy tiềm năng sáng tạo, nâng cao ý thức học tập, trình độ tin học và ứng dụng công nghệ thông tin trong học tập, sinh hoạt, vui chơi giải trí của thanh thiếu nhi, qua đó phát hiện, bồi dưỡng tài năng tin học trẻ góp phần đẩy mạnh việc ứng dụng và phát triển công nghệ thông tin phục vụ sự nghiệp công nghiệp hóa, hiện đại hóa đất nước.

## Nội dung và hình thức

### Nội dung và hình thức thi

#### 1. Phần thi kiến thức, kỹ năng tin học.

Học sinh Tiểu học, Trung học cơ sở (THCS), Trung học phổ thông (THPT) đăng ký dự thi theo các bảng sau:

- **Bảng A: Thi kỹ năng lập trình theo hướng tạo ra sản phẩm cấp Tiểu học**

Thi cá nhân sử dụng ngôn ngữ Scratch và thư viện hình ảnhâm thanh do Ban Giám khảo cung cấp để lập trình giải bài toán theo hướng tạo ra sản phẩm sáng tạo gắn với thực tế.

Ngoài nội dung thi kỹ năng lập trình, một số quận/huyện, tỉnh/thành có yêu cầu thí sinh thực hành thêm kỹ năng sử dụng những tính năng cơ bản trong các phần mềm MS Word, Excel, PowerPoint, Paint để soạn thảo văn bản, trình chiếu... Tuy nhiên với xu hướng hiện nay, các đề có thêm phần thực hành sử dụng những tính năng đang giảm dần và có thể sẽ không được đưa vào đề thi trong các năm tới.

- **Bảng B: Thi kỹ năng lập trình cấp THCS**

Thi cá nhân sử dụng ngôn ngữ Free Pascal, DevC++, CodeBlock, Python... để lập trình online theo thể thức quốc tế (ICPC) với các thuật toán thuộc chương trình THCS.

- **Bảng C: Thi kỹ năng lập trình tập thể cấp THPT**

Thi theo đội với tối đa 02 thí sinh/đội, mỗi đội tuyển tự trang bị 01 máy tính xách tay được cài sẵn Free Pascal, DevC++, CodeBlock, Python... để lập trình online theo thể thức quốc tế (ICPC) với các thuật toán thuộc chương trình THPT.

#### 2. Phần thi Sản phẩm sáng tạo (SPST)

- Tùy thuộc vào từng khu vực sẽ có tên bảng thi phần SPST khác nhau. Ví dụ tại TP. Hà Nội tên bảng thi SPST các cấp Tiểu học, THCS, THPT tương ứng các bảng D1, D2 và D3. Tuy nhiên tại TP. Hồ Chí Minh tên bảng SPST các cấp THCS, THPT lại là B1 và C1. Do đó thí sinh đăng ký phần thi SPST chú ý tên bảng đăng ký dự thi dựa vào thể lệ cuộc thi tại khu vực đăng ký.

- Phần thi dành cho thí sinh có SPST (phần mềm, phần cứng hoặc sản phẩm tích hợp) nhằm phục vụ học tập, giải trí và các nhu cầu thực tiễn khác. Thí sinh tự trang bị máy tính để triển lãm và bảo vệ SPST đã được thiết kế bằng tất cả các công cụ (phần mềm, phần cứng) hiện có.
- Khuyến khích xây dựng SPST tích hợp ứng dụng trong lĩnh vực tự động hóa, giáo dục, giao thông, môi trường, nông nghiệp, đặc biệt là nông nghiệp ứng dụng công nghệ cao. Nếu SPST đã tham gia cuộc thi, hội thi cấp quốc gia nhưng chưa đạt giải hoặc tham khảo mã nguồn mở thì trong bản thuyết minh, mô tả sản phẩm phải chỉ rõ những nội dung đã được nâng cấp mở rộng so với phiên bản trước đó hay những phần tham khảo, địa chỉ tham khảo, những ý tưởng tham khảo bộ mã nguồn mở.
- Phần thi không giới hạn số lượng sản phẩm tham gia của các đơn vị. Các thí sinh có thể làm SPST và đăng ký dự thi theo nhóm, mỗi nhóm tối đa 03 thí sinh/sản phẩm. Một thí sinh có thể đăng ký dự thi nhiều SPST.
- Tiêu chí chung để đánh giá phần mềm sáng tạo:
  - + Tính sáng tạo về ý tưởng, giải pháp của sản phẩm.
  - + Triển vọng ứng dụng của sản phẩm trong thực tế (tiêu chí ưu tiên).
  - + Khả năng thị trường, thương mại hóa sản phẩm.
  - + Tâm nhìn của sản phẩm (công nghệ, ứng dụng, thương mại).
  - + Hồ sơ thuyết minh, phong cách trình bày báo cáo.

**Lưu ý:** Thí sinh tham gia dự thi có thể đăng ký 01 hoặc cả 02 nội dung thi nêu trên.

## Hình thức tổ chức

Hội thi có 2 hình thức chính: thi theo các cấp cơ sở và thi tự do. Tùy thuộc từng khu vực sẽ có các vòng thi cơ sở từ cấp trường, cấp quận/huyện/thị xã, tỉnh/thành phố để chọn ra các thí sinh/đội tham gia vào vòng thi cấp Quốc gia. Mỗi vòng thi sẽ có thể lệ và số lượng thí sinh/đội tham gia dự thi riêng. Trong trường hợp thí sinh muốn đăng ký dự thi nhưng tại khu vực không tổ chức thi hoặc các cá nhân/đội đã đăng ký tham gia dự thi các cấp cơ sở nhưng chưa đạt được kết quả, thành tích như mong muốn. Khi đó cá nhân/đội có thể đăng ký tham gia thi tự do.

### 1. Thí sinh đăng ký thi các cấp cơ sở

Tùy thuộc từng trường sẽ tổ chức hoặc không tổ chức vòng loại cấp trường. Các thí sinh được trường cử tham gia dự thi sẽ tham gia các vòng thi sau:

#### Giai đoạn 1: Vòng sơ tuyển cấp quận, huyện, thị xã.

Đoàn Thanh niên, Phòng Giáo dục và Đào tạo sẽ phối hợp xây dựng kế hoạch và tổ chức Hội thi Tin học trẻ cấp cơ sở, tuyển chọn những thí sinh đạt kết quả cao để thi cấp tỉnh, thành phố.



Trường hợp các quận, huyện, thị xã có số thí sinh đăng ký dự thi chưa vượt quá số lượng thí sinh tối đa, các thí sinh đăng ký sẽ được tuyển thẳng vào vòng thi cấp tỉnh, thành phố (nếu tỉnh, thành phố có tổ chức Hội thi). Thời gian tổ chức vòng sơ tuyển cấp quận, huyện, thị xã thường diễn ra từ tháng 3 đến tháng 5.

### **Giai đoạn 2: Vòng thi cấp tỉnh, thành phố**

Đoàn Thanh niên, Sở Giáo dục và Đào tạo sẽ phối hợp xây dựng kế hoạch và tổ chức Hội thi Tin học trẻ cấp tỉnh, thành phố, trao giải những thí sinh đạt giải và ôn luyện những thí sinh đạt kết quả cao để tham dự Hội thi Tin học trẻ. Vòng thi cấp tỉnh, thành phố thường được tổ chức trong khoảng từ tháng 3 đến tháng 5.

### **Giai đoạn 3: Vòng sơ khảo Quốc gia**

Trung ương Đoàn TNCS Hồ Chí Minh, Bộ Khoa học và Công nghệ, Bộ Giáo dục và Đào tạo, Bộ Thông tin và Truyền thông, Đài Truyền hình Việt Nam, Hội Tin học Việt Nam phối hợp tổ chức Vòng sơ khảo Quốc gia.

Vòng sơ khảo quốc gia gồm các **thí sinh xuất sắc được lựa chọn từ vòng thi cấp tỉnh, thành phố** và các **thí sinh tự do xuất sắc** vượt qua các bài thi online do Trung tâm Phát triển Khoa học, Công nghệ và Tài năng trẻ - Trung ương Đoàn TNCS Hồ Chí Minh tổ chức.

Vòng thi sơ khảo Quốc gia được tổ chức vào khoảng tháng 8 trong năm. Các thí sinh tham gia dự thi tập trung tại 3 khu vực: Hà Nội, TP. Hồ Chí Minh và Đà Nẵng.

### **Giai đoạn 4: Vòng thi chung kết Quốc gia**

Trung ương Đoàn TNCS Hồ Chí Minh, Bộ Khoa học và Công nghệ, Bộ Giáo dục và Đào tạo, Bộ Thông tin và Truyền thông, Đài Truyền hình Việt Nam, Hội Tin học Việt Nam phối hợp tổ chức Hội thi toàn quốc.

Thời gian tổ chức sẽ do các cơ quan thống nhất và quyết định. Theo các năm, Hội thi toàn quốc được tổ chức trong khoảng từ tháng 7 đến tháng 8. Tuy nhiên trong năm 2020 Hội thi các cấp đều tổ chức muộn hơn do ảnh hưởng bởi dịch bệnh Covid-19.

Các thí sinh tham gia dự thi vòng chung kết Quốc gia gồm các thí sinh xuất sắc được lựa chọn từ Vòng sơ khảo Quốc gia. Vòng chung kết quốc gia gồm tối đa 30 thí sinh bảng A, tối đa 30 đội mỗi bảng B và C, tối đa 15 SPST.

## **2. Thí sinh đăng ký thi tự do**

Trường hợp tại khu vực thí sinh muốn tham gia dự thi nhưng quận, huyện, thị xã không tổ chức thi hoặc các thí sinh đã tham gia dự thi các cấp nhưng **không thuộc đội tuyển được các tỉnh, thành Đoàn cử tham dự Vòng sơ khảo Quốc gia** mà muốn tham gia thi Vòng sơ khảo Quốc gia để thử sức và cọ xát cũng có thể đăng ký tuyển chọn vào Vòng thi sơ khảo Quốc gia bằng cách đăng ký thi tự do.

Các thí sinh đăng ký thi tự do tại <http://tainangviet.vn>. Ban Tổ chức Hội thi sẽ tuyển chọn tối đa 20 thí sinh/đội mỗi bảng tham gia dự thi vào Vòng sơ khảo Quốc gia theo kết quả kiểm tra đạt yêu cầu và xét duyệt từ cao xuống thấp.

Thời hạn đăng ký thi tự do phụ thuộc thể lệ Hội thi Tin học trẻ toàn quốc. Thông thường thời hạn đăng ký thi tự do sẽ sau các Vòng thi cấp tỉnh, thành phố.

Các thí sinh được lựa chọn vào Vòng sơ khảo Quốc gia tập trung tại 3 khu vực: Hà Nội, TP. Hồ Chí Minh và Đà Nẵng để tham gia dự thi cùng các thí sinh được các tỉnh, thành Đoàn lựa chọn tham gia dự thi.

## **Đăng ký tham dự**

### **Hình thức 1: Đăng ký thi các cấp cơ sở**

Hội thi Tin học trẻ là cuộc thi không mang tính bắt buộc với tất cả các trường. Phụ huynh và học sinh có thể liên hệ với nhà trường để nhận được hướng dẫn đăng ký và nộp phiếu dự thi theo mẫu.

Hồ sơ đăng ký dự thi bao gồm:

- Thí sinh đăng ký dự thi phần kiến thức và kỹ năng tin học:
  - + Phiếu đăng ký dự thi theo mẫu (dán ảnh 4 x 6 chụp không quá 6 tháng).
- Thí sinh dự thi SPST:
  - + 02 bản thuyết minh trên khổ giấy A4 (theo mẫu) có đóng quyển, có xác nhận của đơn vị chọn cử (riêng Phần mềm sáng tạo gửi kèm thêm 01 đĩa CD/DVD hoặc Flash RAM chứa sản phẩm được đóng cùng mã nguồn, bản thuyết minh để Hội đồng Giám khảo theo dõi, đánh giá).
  - + Đôi với các SPST đã dự thi các năm trước, hoặc tham khảo mã nguồn mở thì trong bản thuyết minh, mô tả sản phẩm phải chỉ rõ những nội dung đã được nâng cấp mở rộng so với phiên bản trước đó hay những phần tham khảo, địa chỉ tham khảo, những ý tưởng tham khảo trong bộ mã nguồn mở.

### **Hình thức 2: Đăng ký thi tự do**

Thí sinh tự do đăng ký dự thi trực tuyến tại <http://tainangviet.vn> và nộp lệ phí thi trước hạn đăng ký. Mức phí và thời hạn đăng ký có thể thay đổi theo từng năm. Năm 2021 mức phí đăng ký thi tự do là 400.000 VNĐ/thí sinh, đội; thời gian đăng ký vào khoảng tháng 7.

## Giải thưởng

Số lượng, cơ cấu và giá trị giải thưởng phụ thuộc điều kiện, thể lệ thi từng vòng và từng khu vực. Các thí sinh đạt giải từ cấp tỉnh, thành phố trở lên sẽ nhận được bằng khen của Ban Chấp hành Thành đoàn hoặc Trung ương Đoàn và phần thưởng bằng tiền mặt hoặc hiện vật.

Tất cả các thí sinh dự thi Vòng sơ khảo Quốc gia và Vòng chung kết Quốc gia sẽ được Ban Tổ chức Hội thi cấp Giấy chứng nhận tham dự Hội thi.

Cụ thể số lượng, cơ cấu giải thưởng của Hội thi Tin học trẻ cấp toàn quốc năm 2020 như sau:

<b>Giải</b>	<b>Bảng</b>	<b>A</b>	<b>B</b>	<b>C</b>	<b>D2</b>	<b>D3</b>
Giải nhất		1	1	1	1	1
Giải nhì		3	3	3	2	2
Giải ba		6	6	6	5	5
Giải khuyến khích		10	10	10	7	7
<b>Tổng cộng</b>		<b>20</b>	<b>20</b>	<b>20</b>	<b>15</b>	<b>15</b>

Giải Nhất, Nhì, Ba đồng đội được trao cho các đơn vị có kết quả thi toàn đoàn cao nhất theo tổng sắp giải thưởng.

Với mỗi vòng thi, tùy thuộc vào kết quả thi, Ban Giám khảo có thể đề xuất với Ban Tổ chức Hội thi tăng hoặc giảm các giải thưởng.

Thí sinh đạt giải Nhất các bảng thi vòng chung kết Quốc gia được tặng Huy hiệu Tuổi trẻ sáng tạo của Ban Chấp hành Trung ương Đoàn và Bằng khen của Bộ trưởng Bộ Giáo dục và Đào tạo; các giải Nhất đồng đội và giải Nhì, Ba (các bảng thi và đồng đội) được tặng Bằng khen của Ban Chấp hành Trung ương Đoàn.

Thí sinh đạt giải Nhất, Nhì, Ba vòng chung kết Quốc gia sẽ được lưu hồ sơ cá nhân trong Hệ thống cơ sở dữ liệu Tài năng trẻ Việt Nam.

Đặc biệt, thí sinh đạt giải cao (cấp Tỉnh/Thành phố hoặc cấp Quốc gia) có thể được xét tuyển trực tiếp hoặc được cộng điểm vào các trường THCS, THPT (tùy thuộc vào quy định từng trường và từng năm học), ví dụ trường THCS và THPT Nguyễn Tất Thành có điều kiện xét tuyển thẳng vào lớp 6 năm học 2019 – 2020 là đạt giải Nhất, Nhì cuộc thi Tin học trẻ cấp Quốc gia.

## Giải đề theo các dạng bài

Sáu dạng bài phổ biến nhất trong đề thi gồm Số học; Xử lý xâu và chuỗi; Sử dụng mảng một chiều/danh sách; Ma trận; Sắp xếp và Thuật toán sẽ được tổng hợp trong chương này. Mỗi dạng sẽ gồm các bài giải mẫu và bài tập tự thực hành. Đa phần các bài được lấy trong đề thi của các tỉnh, thành phố và các quận, huyện trên cả nước.

Mỗi bài đều được chia nhỏ thành các bước hoặc các phần. Khi ghép các đoạn mã lệnh với nhau thành đoạn chương trình (hoặc toàn bộ chương trình), với một số đoạn chương trình sẽ có kích thước rất lớn, vì vậy đoạn chương trình sau khi lập trình có thể sẽ không được đưa vào sách, các bạn sẽ tự mình sắp xếp vị trí dòng lệnh dựa theo số thứ tự đứng trước từng dòng lệnh để hoàn thiện và chạy chương trình. Ngoài cách giải được nêu trong sách, bạn đọc được khuyến khích suy nghĩ và tìm nhiều cách giải khác cho bài toán.



## DẠNG BÀI SỐ HỌC

### Bài 1. Giá trị biểu thức (TP. Vũng Tàu, 2020)

Cho 3 số nguyên dương  $a, b, c$ .

**Yêu cầu:**

Tính giá trị của biểu thức:

$$S = \frac{a^2 + b^2 + c^2}{abc} + \sqrt{abc}$$

**Dữ liệu vào:**

File ROOT.INP chứa 3 số nguyên dương  $a, b, c$ . Mỗi số trên một dòng.

**Kết quả:**

Ghi vào File ROOT.OUT kết quả  $S$  tính được (làm tròn lấy 2 chữ số sau phần thập phân).

**Ví dụ:**

<b>ROOT.INP</b>	<b>ROOT.OUT</b>
2	4.25
1	
2	

Ta thấy, trong biểu thức có chứa căn bậc hai của tích 3 số  $a, b$  và  $c$ . Để sử dụng được hàm căn bậc hai trong Python, ta cần thực hiện import thư viện math vào chương trình.

```
1 from math import *
```

Tiếp đó, ta cần mở file dữ liệu vào và dữ liệu ra của bài toán. Với file dữ liệu vào ta chỉ mở lên để đọc dữ liệu, còn file dữ liệu ra ta cần mở để thay đổi dữ liệu trong file.

```
2 file = open("ROOT.INP")
3 file2 = open("ROOT.OUT", "w")
```



### Bạn có biết: Đọc và ghi dữ liệu trong Python

Để có thể đọc và ghi dữ liệu vào file, việc đầu tiên cần làm là mở file lên. Python cho phép chúng ta mở file với hàm open.

Cú pháp:

```
open(fileName, mode)
```

Trong đó:

- fileName: Tên của tập tin cần mở.
- mode: Quy định cách thức mở tập tin, nếu mode không điền gì, chương trình sẽ hiểu mặc định mở file để đọc.

**Yêu cầu:** File cần mở ở cùng thư mục với file code.

Các mode gồm có:

Mode	Mô tả
r	Mở file chỉ để đọc
r+	Mở file để đọc và ghi
w	Tạo một file mới để ghi, nếu file đã tồn tại thì sẽ bị ghi mới
w+	Tạo một file mới để đọc và ghi, nếu file đã tồn tại thì sẽ bị ghi mới
a	Mở file để ghi thêm vào cuối file, nếu không tìm thấy file sẽ tạo mới một file
a+	Mở file để đọc và ghi thêm vào cuối file, nếu không tìm thấy file sẽ tạo mới một file

Dữ liệu trong file ROOT.INP được viết trên từng dòng, để đọc dữ liệu theo dòng, ta sử dụng phương thức readline(). Dữ liệu đọc được ta gán vào biến, ví dụ biến **a**.

```
4 a = file.readline()
```

Tương tự ta đọc tiếp dòng dữ liệu tiếp theo và gán vào hai biến **b** và **c**.

```
5 b = file.readline()
```

```
6 c = file.readline()
```

Các biến **a**, **b**, **c** hiện tại được gán bằng các dòng dữ liệu đọc từ file, vậy kiểu dữ liệu hiện tại của 3 biến này là kiểu chuỗi (string, ký hiệu 'str'). Do vậy để có thể tính toán với 3 biến **a**, **b**, **c**, ta cần chuyển đổi 3 biến này sang kiểu dữ liệu số. Việc chuyển đổi kiểu dữ liệu này ta còn gọi là ép kiểu cho biến.



```
8  a = int(a)
9  b = int(b)
10 c = int(c)
```

**Lưu ý:** Để phân biệt chức năng của từng đoạn mã lệnh (mở file, đọc dữ liệu, tính toán,...), ta nên chèn 1 dòng lệnh trống giữa đoạn mã lệnh đó. Trường hợp chỉ số đầu dòng cuối cùng của đoạn lệnh bên trên nhỏ hơn 2 giá trị so với chỉ số đầu dòng số dòng đoạn lệnh bên dưới, các bạn ngầm hiểu sẽ có một dòng lệnh trống giữa 2 đoạn lệnh đó.



### Bạn có biết: Các phương thức đọc dữ liệu

#### 1. Phương thức read()

Phương thức cho phép ta đọc một lượng dữ liệu tương ứng với tham số size và trả về một chuỗi, khi không điền size, mặc định chương trình sẽ đọc toàn bộ dữ liệu trong tệp tin.

Cú pháp:

```
file.read(size)
```

#### 2. Phương thức readline()

Phương thức cho phép ta đọc một lượng dữ liệu tính theo byte (size) trong 1 dòng từ tệp tin và trả về một chuỗi, khi không điền giá trị size, mặc định chương trình sẽ đọc hết 1 dòng dữ liệu.

Cú pháp:

```
file.readline(size)
```

#### 3. Phương thức readlines()

Phương thức cho phép ta đọc toàn bộ file và trả về một danh sách. Mỗi phần tử trong danh sách là một dòng của file.

Cú pháp:

```
file.readlines()
```

**Lưu ý:** Các phương thức đọc dữ liệu đều trả về kiểu dữ liệu chuỗi (str), vậy nên khi đọc giá trị các số, ta cần ép kiểu dữ liệu số (int – số nguyên, float – số thực) cho dữ liệu sau khi đọc từ tệp tin.

Tiếp theo ta tính tử số và mẫu số của biểu thức, sau đó gán vào biến **tu** và **mau**.

```
12 tu = a**2 + b**2 + c**2
13 mau = a * b * c
```

Từ tử số và mẫu số đã tính được, ta thực hiện tính giá trị biểu thức và lưu vào biến **s**.

```
14 s = tu/mau + sqrt(mau)
```

Để bài yêu cầu kết quả sẽ ghi vào file ROOT.OUT với 2 chữ số phần thập phân, để làm tròn tới 2 chữ số phần thập phân, ta sử dụng hàm round(). Ngoài ra, để ghi dữ liệu vào file, ta cần chuyển kiểu dữ liệu của biến **s** từ kiểu dữ liệu số thành kiểu chuỗi.

```
15 s = str(round(s, 2))
16
17 file2.write(s)
```

Cuối cùng, sau khi hoàn thiện việc đọc dữ liệu và ghi dữ liệu vào file, ta đóng 2 file dữ liệu bằng phương thức close().

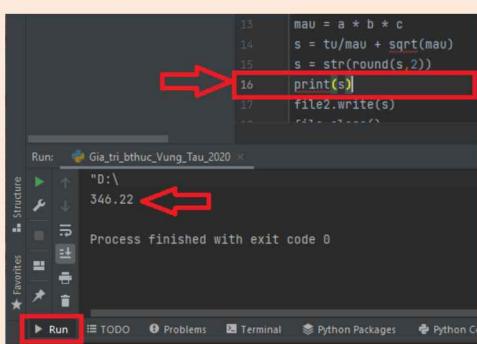
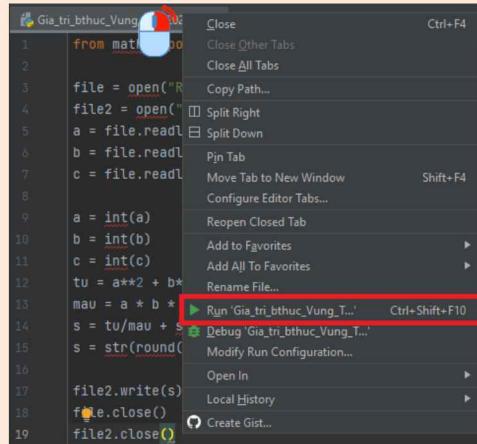
```
18 file.close()
19 file2.close()
```



### Bạn có biết: Cách chạy chương trình và kiểm tra kết quả trên cửa sổ Run

#### 1. Cách chạy chương trình

Để chạy file hiện tại đang được mở trong PyCharm, ta nhấn chuột phải lên tên tệp tin sau đó chọn Run, hoặc sử dụng tổ hợp phím **Ctrl + Shift + F10**.



#### 2. Kiểm tra kết quả trên cửa sổ Run

Để có thể kiểm tra kết quả đã tính được có chính xác hay không, ngoài cách mở tệp tin ROOT.OUT, các bạn có thể thêm lệnh print(s) vào vị trí dòng thứ 16 trong dự án để kiểm tra kết quả trong cửa sổ Run.

## Bài 2. Tự bài toán cỗ (Bắc Giang, 2020)

Nhà An có một trang trại rộng lớn. Do sở thích của An nên bố An chỉ nuôi gà và chó. Một hôm bố an đố con gái nhà mình nuôi bao nhiêu con gà, bao nhiêu con chó? Bố An cho biết nhà có tổng số gà và chó là x con. Do số lượng nhiều và khó đếm từng loại nên An chỉ đếm được tổng số chân của gà và chó là y chân. Em hãy giúp An trả lời câu đố của bố.

Dữ liệu vào: đọc từ file văn bản TOANCO.INP gồm 2 số nguyên dương x, y trên một dòng. Hai số cách nhau một khoảng trắng ( $x \leq 10^5$ ,  $y \leq 4 \cdot 10^5$ ).

Kết quả: ghi ra file văn bản TOANCO.OUT gồm 2 số tương ứng là số gà và số chó tìm được. Hai số cách nhau một khoảng trắng. Giải sử bài toán luôn có nghiệm.

<b>TOANCO.INP</b>	<b>TOANCO.OUT</b>
36 100	22 14

Tương tự bài toán trên, trước tiên ta cần mở 2 file lưu dữ liệu vào và dữ liệu ra của bài toán, sau đó đọc dữ liệu từ file bằng phương thức `readline()`.

```
1 file = open("TOANCO.INP")
2 file2 = open("TOANCO.OUT", "w")
3 s = file.readline()
```

Vì dữ liệu trong bài có cấu trúc gồm 2 số trên 1 dòng mà không phải mỗi số trên 1 dòng, vậy sau khi đọc dữ liệu, ta sử dụng phương thức `split()`. Với phương thức này, ta có thể tách chuỗi thành một danh sách, ta có thể chỉ định ký tự phân cách khi tách chuỗi (ví dụ: dấu ‘,’, dấu ‘;’, dấu cách,...). Ký tự phân cách mặc định là dấu cách (khoảng trắng). Sau khi phân tách, ta gán giá trị đã phân tách vào 2 biến **x** và **y**.

```
4 s = s.split()
5 tong = int(s[0])
6 chan = int(s[1])
```



### Bạn có biết: Danh sách trong Python

Danh sách (List) là một tập hợp các thông tin được lưu trữ theo thứ tự. Danh sách cho phép chứa dữ liệu trùng lặp và có khả năng lưu trữ các kiểu dữ liệu khác nhau (kiểu string, int, ...).

Chỉ mục	Thông tin
0	‘Name’
1	‘Address’
2	2021

Mỗi thông tin trong danh sách ta gọi là phần tử và các phần tử sẽ có chỉ mục (index) tăng dần từ 0. Ta có thể lấy và thay đổi thông tin tại bất kỳ chỉ mục nào trong danh sách.

Sau khi lưu được dữ liệu, ta tạo hai biến **a**, **b** tương ứng số gà và số chó. Ban đầu ta gán giá trị cho 2 biến là 0.

```
8  a = b = 0
```

Để giải bài toán, ta sử dụng vòng lặp for. Ta cho **a** chạy từ 1 tới tổng số gà và chó đã cho. Với mỗi giá trị **a**, ta có  $(x - a)$  là số chó, từ đó ta cộng số chân gà và chân chó. Nếu giá trị tính được là **y** – tổng số chân gà và chó mà đề bài đã cho thì **a** và **x - a** là số gà và số chó cần tìm.

```
9  for a in range (tong):
10     b = tong - a
11     if (a * 2 + b * 4) == chan:
12         break
```

Sau khi đã tìm được số gà và chó, ta ghi dữ liệu vào file2 và đóng 2 file dữ liệu.

```
14 file2.write(str(a) + " " + str(b))
15 file.close()
16 file2.close()
```

**Lưu ý:**

- + Vòng lặp for và cấu trúc điều kiện if cần có dấu ":" phía sau dòng lệnh.
- + Độ thụt lề của các dòng lệnh trong vòng lặp và trong cấu trúc điều kiện.

### Bài 3. Tính tổng (Yên Bá, 2020)

Viết chương trình nhập vào 02 số nguyên  $A$ ,  $B$  ( $0 < A < B$ ).

- Tìm và tính tổng các số nguyên tố của dãy số từ  $A$  đến  $B$ .
- Xuất ra màn hình các số chia hết cho 5 của dãy số từ  $A$  đến  $B$ .

Ví dụ: nhập  $A = 6$ ,  $B = 22$ .

Kết quả tổng các số nguyên tố trong dãy số từ 6 đến 22 là:

$$7 + 11 + 13 + 17 + 19 = 67$$

Các số chia hết cho 5 của dãy số từ 6 đến 22 là: 10, 15, 20.

Trước tiên, ta import thư viện math vào dự án.

```
1  from math import *
```



Đề bài yêu cầu nhập vào 2 số nguyên A và B, để nhập dữ liệu vào từ cửa sổ Run, ta sử dụng hàm input().

```
3 a = input("Nhập vào số nguyên A: ")
4 b = input("Nhập vào số nguyên B: ")
```

Sau khi nhập dữ liệu, ta cần ép kiểu dữ liệu cho 2 biến **a** và **b** từ kiểu chuỗi thành kiểu số nguyên. Tiếp đó tạo biến lưu giá trị tổng các số nguyên tố và chuỗi các số chia hết cho 5 cần tìm theo yêu cầu của đề bài.

```
5 a = int(a)
6 b = int(b)
7 tong = 0
8 s = ""
```

Trong bài yêu cầu tính tổng các số nguyên tố, để kiểm tra một số có phải là số nguyên tố hay không, ta tạo hàm **nguyento** với giá trị truyền vào là số cần kiểm tra tính nguyên tố **x**.

Trong hàm kiểm tra tính nguyên tố, trường hợp **x** = 1 không là số nguyên tố, hàm sẽ trả về kết quả False. Trường hợp còn lại ta chạy từ 2 tới căn bậc hai của **x**, nếu **x** chia hết cho **i** thì hàm trả về kết quả False. Nếu chạy hết vòng lặp thì trả về kết quả True. Tuy nhiên vòng lặp for chỉ chạy các số nguyên mà hàm căn bậc hai có thể sẽ trả về giá trị là số thập phân, vậy ta cần ép kiểu int cho phép khai căn và cộng thêm 1 giá trị.

```
10 def nguyento(x):
11     if x == 1:
12         return False
13     for i in range (2, int(sqrt(x))+1):
14         if x % i == 0:
15             return False
16     return True
```

Sau khi tạo được hàm kiểm tra tính nguyên tố, trong đoạn chương trình chính ta sử dụng vòng lặp for và chạy có giá trị từ **a** tới **b** + 1. Với mỗi giá trị **i**, ta kiểm tra **i** có phải là số nguyên tố hay không bằng cách gọi hàm **nguyento(i)**, nếu **i** là số nguyên tố thì tăng giá trị biến tổng thêm . Đồng thời kiểm tra **i** có chia hết cho 5 hay không. Nếu chia hết ta nối thêm **i** và dấu “,” vào chuỗi **s**. Tuy nhiên chúng ta cần ép kiểu str cho **i** khi nối vào chuỗi **s**.

```
18 for i in range (a, b+1):
19     if nguyento(i):
20         tong += i
21     if i % 5 == 0:
22         s += str(i) + ","
```

Khi đã tìm được kết quả bài toán, ta thực hiện in kết quả lên màn hình bằng cách sử dụng lệnh print(). Tuy nhiên với chuỗi **s**, ta đã nối thêm dấu “,” phía sau, vậy ta sẽ in ra chuỗi **s** và bớt lại 2 ký tự cuối cùng của chuỗi.

```
24 print (tong)
25 print (s[:len(s)-2])
```

#### Bài 4. Số mạnh mẽ (Hải Dương, 2020)

*Số mạnh mẽ là số khi nó chia hết cho số nguyên tố thì cũng chia hết cho cả bình phương của số nguyên tố đó.*

Ví dụ: 25 là số mạnh mẽ, vì nó chia hết cho số nguyên tố 5 và chia hết cho cả bình phương của 5 là 25.

Viết chương trình liệt kê các số mạnh mẽ không vượt quá 1000.

**Phân tích:** Để giải bài toán, với mỗi giá trị n trong khoảng từ 2 tới 1000, ta kiểm tra n có chia hết cho i và chia hết cho cả  $i^2$  hay không ( $2 \leq i \leq n$ ) thì thực hiện kiểm tra i có phải số nguyên tố hay không. Nếu i là số nguyên tố thì n là số mạnh mẽ.

Trước tiên, ta import thư viện math vào dự án và tạo hàm NguyenTo(a) để kiểm tra tính nguyên tố của số a, với a là tham số được truyền vào hàm.

Trong hàm, ta kiểm tra 3 trường hợp:

- Khi  $a = 1$ , hàm trả về giá trị False – a không phải số nguyên tố.
- Khi  $a = 2$ , hàm trả về giá trị True – a là số nguyên tố.
- Khi  $a > 2$  ta kiểm tra trong khoảng từ 2 tới  $\sqrt{a} + 1$ :
  - + Nếu a chia hết cho bất kỳ số nào trong khoảng này, hàm trả về giá trị False.
  - + Nếu a không chia hết cho số nào, hàm trả về giá trị True.

```
1 from math import *
2
3 def NguyenTo(a):
4     if a == 1:
5         return False
6     if a == 2:
7         return True
8     for i in range(2, int(sqrt(a))+1):
9         if a % i == 0:
10             return False
11     return True
```

Sau khi đã tạo hàm kiểm tra tính nguyên tố của một số, ta tạo biến n lưu giá trị cần kiểm tra có phải số mạnh mẽ hay không và biến s lưu kết quả của bài toán.



```
13 s = ''
14 n = 2
```

Ta cần tìm các số mạnh mẽ trong khoảng từ 1 tới 1000. Chính vì vậy, ta thực hiện lặp lại việc kiểm tra tới khi n lớn hơn 1000.

```
16 while n <= 1000:
17     for i in range (2, n):
18         if n % i == 0 and n % (i*i) == 0:
19             if NguyenTo(i):
20                 s += str(n) + ' '
21             break
22     n += 1
```

Cuối cùng, ta in kết quả là chuỗi s lên màn hình.

```
24 print(s)
```

### Bài 5. Bộ chính phương (Quốc gia, 2020)

*Cho một dãy số A có N phần tử. Tìm số nguyên dương P nhỏ nhất thỏa mãn: P là số chính phương và P chia hết cho tất cả các phần tử của dãy số A.*

**Yêu cầu:** In ra phần dư của phép chia khi chia P cho  $10^9 + 7$ .

**Dữ liệu:** Vào từ thiết bị theo khuôn dạng sau:

- + Dòng đầu tiên chứa số nguyên dương N là số lượng phần tử của dãy số.
- + Dòng tiếp theo chứa N số nguyên dương là các phần tử của dãy A.

Các số trên một dòng được ghi cách nhau bởi dấu cách.

**Kết quả:** Ghi ra thiết bị ra gồm một số nguyên duy nhất là kết quả của bài toán.

Ví dụ:

Dữ liệu vào	Dữ liệu ra
3	36
2 1 3	

**Phân tích:** Để giải bài toán, ta tạo biến **num** lưu giá trị cần tìm của bài toán. Biến **num** ban đầu ta đặt giá trị là 1 và tăng dần lên bằng cách sử dụng vòng lặp **while**. Với mỗi giá trị **num**, ta tiến hành kiểm tra  $\text{num}^2$  có chia hết cho từng phần tử trong dãy số hay không, nếu không ta tiếp tục tăng **num** thêm 1, còn không thì xuất kết quả vào file dữ liệu ra và thoát vòng lặp.

Trước tiên, ta mở 2 file dữ liệu và đọc dữ liệu vào của bài toán.

```

1  file = open('INPUT.INP')
2  file2 = open('OUTPUT.OUT', 'w')
3
4  n = int(file.readline())
5  data = file.readline()

```

Sau đó ta tách dữ liệu vào thành danh sách và khởi tạo các giá trị biến.

```

6  a = data.split()
7  kq = 0
8  num = 1

```

Ta sử dụng vòng lặp while để lặp lại việc tăng num tới khi tìm được đáp án, sử dụng vòng lặp for để kiểm tra  $\text{num}^2$  có chia hết cho từng số trong dãy số hay không. Khi gặp 1 số mà  $\text{num}^2$  không chia hết, ta thoát vòng lặp.

```

10 while True:
11     for i in range(0, n):
12         temp = int(a[i])
13         if (num**2) % temp != 0:
14             break

```

Sau khi kiểm tra từng số trong dãy số, ta kiểm tra giá trị  $i$  hiện tại nếu bằng  $n - 1$  và  $\text{num}$  chia hết cho số cuối cùng trong dãy số thì  $\text{num}^2$  chính là giá trị P cần tìm trong bài toán. Tuy nhiên, theo yêu cầu của bài toán, ta cần in ra phép chia lấy dư của P cho  $10^9 + 7$ , vậy ra tính kết quả bài toán, lưu vào biến  $kq$  và ghi kết quả vào file dữ liệu ra của bài toán. Trường hợp  $i < n - 1$  hoặc  $\text{num}^2$  không chia hết cho số cuối cùng trong dãy số, nghĩa là giá trị  $\text{num}$  hiện tại không phải giá trị ta cần tìm, vậy ta tăng  $\text{num}$  thêm 1 và tiếp tục vòng lặp kiểm tra giá trị  $\text{num}$ .

```

15     if i == n-1 and (num**2) % temp == 0:
16         kq = num**2 % (10**9 + 7)
17         file2.write(str(kq))
18         break
19     else:
20         num += 1

```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán sau khi ghi xong kết quả vào file dữ liệu ra.

```

22 file.close()
23 file2.close()

```



## Các bài tập thực hành

### Bài tập 1. Số hạnh phúc (Hải Dương, 2020)

Với một số nguyên dương bất kì, thay thế số đó bằng tổng bình phương các chữ số của nó và cứ lặp lại quá trình đó sẽ có những trường hợp sau xảy ra:

Kết thúc bằng 1 – Ta gọi số đó là số hạnh phúc.

Kết thúc bằng 0 – Ta gọi số đó là số buồn bã.

Lặp lại vô hạn lần – Số đó không hạnh phúc cũng không buồn bã.

Ví dụ: Số 44

$$+ Lần 1: 4^2 + 4^2 = 16 + 16 = 32$$

$$+ Lần 2: 3^2 + 2^2 = 9 + 4 = 13$$

$$+ Lần 3: 1^2 + 3^2 = 1 + 9 = 10$$

$$+ Lần 4: 1^2 + 0^2 = 1 + 0 = 1$$

Vậy số 44 là số hạnh phúc.

Em hãy viết chương trình để kiểm tra xem ngày sinh của một người bất kỳ có phải là số hạnh phúc hay không?

### Bài tập 2. Phân số tối giản (Gia Lai, 2019)

Một chuỗi được gọi là có dạng phân số nếu nó có dạng như sau: ‘Tử\_số/Mẫu\_số’.

Viết chương trình nhập vào chuỗi có dạng phân số, sau đó xuất ra dạng tối giản của phân số đó.

Ví dụ:

Chuỗi ‘12/5’ biểu diễn cho phân số. Dạng tối giản của phân số đó là ‘3/5’.

### Bài tập 3. Tổng phần tử lẻ (TP. Tây Ninh, 2019)

Cho một dãy gồm  $N$  số nguyên:  $a_1, a_2, a_3, \dots, a_N$ , mỗi số có giá trị không vượt quá  $10^9$ .

**Yêu cầu:**

+ Tính tổng các số lẻ trong dãy.

**Dữ liệu:**

+ Dòng đầu tiên chứa số nguyên dương  $N$  ( $1 \leq N \leq 10^9$ ).

+  $N$  dòng tiếp theo, dòng thứ  $i$  chứa  $a_i$ .

### Bài tập 4. Lãi suất (Quảng Ngãi, 2020)

Một người gửi tiền vào ngân hàng có kỳ hạn là c tháng với lãi suất mỗi tháng là k%, số tiền gửi ban đầu là A (đơn vị triệu đồng).

Tính số tiền người đó nhận được sau t tháng. Biết rằng tiền lãi mỗi tháng được cộng dồn vào tiền gốc, nếu nhận tiền trước kỳ hạn thì số tiền được tính với lãi suất không kỳ hạn là h% của số tiền ban đầu A nhân với số tháng đã gửi. Trong trường hợp rút tiền sau kỳ hạn thì số tháng sau kỳ hạn sẽ được tính với lãi suất không kỳ hạn là h% so với số tiền thu được đã qua kỳ hạn.

**Dữ liệu vào:** Tệp văn bản BL2.INP ghi 5 số Kỳ hạn (c, nếu c = 0 là gửi không kỳ hạn), thời gian gửi (t), số tiền ban đầu (A), lãi suất có kỳ hạn (k), lãi suất không kỳ hạn (h), các số cách nhau một ký tự trắng.

**Dữ liệu ra:** Tệp văn bản BL2.OUT ghi 1 số là số tiền nhận được (kết quả làm tròn đến 1 số lẻ sau dấu chấm thập phân).

Ví dụ:

BL2.INP	BL2.OUT
12 13 100 1.0 0.2	112.9
0 10 100 1.0 0.2	102.0

### Bài tập 5. Nguyên tố (Nghệ An, 2019)

**Minh đố An:** Cho một số chẵn K ( $2 \leq K \leq 1000$ ) hãy tìm hai số nguyên tố sao cho tổng của chúng bằng số chẵn K đã cho.

Em hãy viết chương trình giúp An trả lời câu hỏi của Minh.

**Dữ liệu vào:** Tệp văn bản NT.INP:

Dòng đầu tiên chứa số nguyên dương N tương ứng số test.

N dòng tiếp theo, mỗi dòng chứa một số K.

**Kết quả:** Tệp văn bản NT.OUT gồm N dòng tương ứng N kết quả. Mỗi kết quả hiển thị tổng hai số nguyên tố bằng số K nhập vào.

Ví dụ:

NT.INP	NT.OUT
2	$8 = 5 + 3$
8	$24 = 19 + 5$
24	



### Bài tập 6. Số hoàn hảo (TP. Tây Ninh, 2019)

Em định nghĩa số hoàn hảo như sau: Số hoàn hảo là một số tự nhiên mà tổng tất cả các ước tự nhiên thực sự của nó bằng chính nó. Trong đó ước thực sự của một số là các ước không bằng số đó.

Em hãy lập trình nhập vào một số tự nhiên có 2 chữ số bất kỳ.

In ra màn hình thông báo số vừa nhập có phải là số hoàn hảo hay không? Nếu là số hoàn hảo thì in tất cả các ước của số đó.

### Bài tập 7. Số may mắn (Đồng Nai, 2020)

Để động viên thành tích học tập xuất sắc của các em học sinh lớp 6/3 trong năm học 2019 - 2020, thầy giáo chủ nhiệm đã chuẩn bị các món quà được đánh số từ 1 đến n. Sau đó thầy giáo sẽ cho các em lên bốc thăm để nhận món quà may mắn của mình.

Đầu tiên thầy giáo sẽ ghi tất cả số nguyên lẻ từ 1 đến n, sau đó sẽ ghi tất cả các số nguyên chẵn từ 2 đến n (theo thứ tự tăng dần) để tạo thành một dãy số phàn thưởng.

Mỗi bạn sẽ bốc thăm một số k ứng với con số của món quà mình đạt được.

**Yêu cầu:** In số của món quà học sinh đạt được.

**Dữ liệu vào:**

- Dòng duy nhất ghi số nguyên n và k ( $1 \leq k \leq n \leq 1000$ ).

**Dữ liệu ra:** In số của món quà học sinh đạt được.

SOMAYMAN.INP	SOMAYMAN.OUT
10 6	2

### Bài tập 8. Ước chung lớn nhất (Ninh Bình, 2019)

Nhập vào 3 số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm ước chung lớn nhất của 3 số trên.

Ví dụ: Nhập vào 3 số: 4, 6, 12 thì kết quả ước chung lớn nhất là 2.

## DẠNG BÀI XỬ LÝ XÂU VÀ CHUỖI

### Bài 1. Số đảo ngược (Tỉnh Tây Ninh, 2019)

Tìm số đảo ngược Y của một số X, biết Y gồm các chữ số của X và viết theo thứ tự ngược lại. Xuất ra kết quả là số Y mod 19.

**Dữ liệu:**

Số nguyên dương X.

**Kết quả:**

Y mod 19 (với Y là số đảo ngược của X).

INPUT	OUTPUT	Giải thích
123	17	đảo ngược của 123 là 321 mod 19 = 17

Trước tiên ta lập trình yêu cầu nhập vào số nguyên dương **x**.

```
1 x = input("Nhập vào số X: ")
```

Tiếp đó ta đảo ngược chuỗi **x** vừa nhập vào và gán vào biến **y**.

```
2 y = x[::-1]
```

Cuối cùng ta tính kết quả của phép chia lấy dư của **y** cho 19 và in kết quả lên màn hình.

```
3 kq = int(y) % 19
4 print(kq)
```

### Bài 2. Nén xâu (Bắc Giang, 2020)

Em hãy viết chương trình nhập vào một xâu ký tự chỉ gồm các chữ cái Tiếng Anh, chữ số, dấu cách và dấu gạch nối. Nén các ký tự liên tiếp giống nhau thành số lượng ký tự và ký tự đó, rồi đưa ra xâu sau khi nén.

**Dữ liệu vào:**

Đọc từ file văn bản NENXAU.INP gồm một xâu S có số lượng ký tự không quá 255 ký tự.

**Kết quả:**

Đưa ra file văn bản NENXAU.OUT gồm xâu S sau khi nén.



Ví dụ:

<b>NENXAU.INP</b>	<b>NENXAU.OUT</b>
aaaababb cc	4a1b1a2b4 2c

Trước tiên ta lập trình mở 2 file dữ liệu, đọc và lưu dữ liệu quét được từ tệp tin “NENXAU.INP”.

```
1  file = open("NENXAU.INP")
2  file2 = open("NENXAU.OUT", "w")
3  s = file.readline()
```

Để giải bài toán, ta tạo biến **để** lưu vị trí ký tự và biến **kq** để lưu kết quả của bài toán. Ban đầu **i** = 0 (vị trí ký tự đầu tiên trong xâu **s**) và **kq** là xâu rỗng.

```
5  i = 0
6  kq = ""
```

Ta sử dụng vòng lặp while để lặp khi đã qua vị trí cuối cùng trong xâu **s**. Bên trong vòng lặp while, ta tạo 2 biến **temp** để lưu ký tự cần nén và biến **dem** để lưu số lượng ký tự đó. Với mỗi ký tự cần nén, ta lặp lại việc tăng và tăng giá trị biến **dem** cho đến khi gặp ký tự khác trong xâu. Sau mỗi lần tăng, ta cần kiểm tra giá trị **i** có phải ký tự cuối cùng của xâu **s** hay không, nếu đúng ta thoát vòng lặp bằng lệnh break.

Khi gặp ký tự khác ta thực hiện nén xâu bằng cách nối biến **dem** với **temp** – ký tự được nén.

```
8  while (i < len(s)):
9      temp = s[i]
10     dem = 0
11
12     while (temp == s[i]):
13         i += 1
14         dem += 1
15         if (i == len(s)):
16             break
17     kq += str(dem) + temp
```

Cuối cùng ta thực hiện ghi kết quả vào file dữ liệu ra và đóng 2 file dữ liệu của bài toán.

```
19 file2.write(kq)
20 file.close()
21 file2.close()
```

### Bài 3. Tính nhân (Hậu Giang, 2020)

Viết chương trình nhập vào 2 số nguyên dương  $a$  và  $b$ . Sau đó thực hiện nhân ( $a \times b$ ) như cách nhân bằng tay thông thường.

**Ví dụ:**

- + Nhập vào thửa số thứ 1: 125
- + Nhập vào thửa số thứ 2: 15

**Kết quả:**

$$\begin{array}{r}
 & 125 \\
 \times & 15 \\
 \hline
 & 625 \\
 & 125 \\
 \hline
 & 1875
 \end{array}$$

Trước tiên ta lập trình yêu cầu nhập vào hai giá trị  $a$  và  $b$  tương ứng hai số cần thực hiện tính nhân.

```
1 a = input ("Nhập vào số thứ nhất: ")
2 b = input ("Nhập vào số thứ hai: ")
```

Để thực hiện phép toán nhân và hiển thị cách nhân bằng tay của  $a$  và  $b$ , trước tiên ta cần xét đến dòng có độ dài lớn nhất khi in lên màn hình để in ra các số với vị trí phù hợp. Dòng có độ dài lớn nhất chính là dòng hiển thị kết quả của phép toán, vậy ta thực hiện tính kết quả của phép toán và đặt độ dài lớn nhất bằng độ dài của kết quả tính được cộng thêm 1 (vì trước kết quả có thêm 1 ký tự dấu cách).

```
4 kq = int(a) * int(b)
5 max_len = len(str(kq)) + 1
```

Ta tạo biến  $s$  để lưu chuỗi cần in lên màn hình. Ở dòng đầu tiên, ta cần in ra giá trị  $a$ , tuy nhiên trước khi in, ta cần thực hiện nối thêm các ký tự dấu cách vào trước giá trị  $a$ .

```
7 s = a
8 while (len(s) < max_len):
9     s = " " + s
10 print(s)
```



Sau khi in giá trị **a**, dòng thứ 2 ta cần in dấu “x” và dòng thứ 3 là giá trị **b**. Cách in giá trị **b** cũng tương tự giá trị **a**.

```
11 print("x")
12
13 s = b
14 while (len(s) < max_len):
15     s = " " + s
16 print(s)
```

Tiếp đó ta in các dấu gạch ngang để thực hiện phép nhân.

```
18 PhanCach = "—" * max_len
19 print(PhanCach)
```

Khi tính nhân, ta sẽ lấy từng số từ phải sang trái của số thứ 2, lần lượt nhân với số thứ nhất, các dòng bên dưới sẽ lùi sang trái 1 đơn vị so với dòng bên trên, vậy ta sẽ tạo biến **kc** để lưu số lượng ký tự khoảng cách cần lùi sang trái của dòng đó. Ban đầu ta đặt **kc** = 0 tương ứng dòng đầu tiên sẽ không lùi sang trái.

Để lấy được từng số từ phải sang trái, ta sử dụng vòng lặp for với giá trị bắt đầu là **len(b)** tới giá trị 0 với bước nhảy là -1. Với mỗi số của **b**, ta thực hiện nhân với **a**. Tuy nhiên trước khi in ra giá trị tính được, ta cần nối các ký tự khoảng cách vào sau và trước giá trị sao cho phù hợp.

```
21 kc = 0
22 for i in range (len(b), 0, -1):
23     s = int(b[i-1]) * int(a)
24     if kc > 0:
25         temp = " " * kc
26         s = str(s) + temp
27     s = str(s)
28     while (len(s) < max_len):
29         s = " " + s
30     print(s)
31     kc += 1
```

Cuối cùng ta cần in ra kết quả của phép toán, tuy nhiên trường hợp số **b** gồm 1 chữ số, kết quả đã được in bên trên. Vậy ta chỉ in dấu phân cách và kết quả bên dưới dấu phân cách khi độ dài của **b** lớn hơn 1.

```
33 if (len(b) > 1):
34     print(PhanCach)
35     print(' ' + str(kq))
```

#### Bài 4. Đôi gương

Một xâu (chuỗi) **đôi gương** là một xâu ký tự mà đọc từ phải qua trái hay ngược lại đều như nhau. Ví dụ, xâu "ABCDEDCBA" là **đôi gương**.

Một xâu **phản gương** là một xâu khi mỗi ký tự của nó đảo chiều và xâu được đọc từ phải qua trái thì kết quả giống xâu gốc. Ví dụ, xâu "3AIAE" là một xâu **phản gương** vì "A" và "I" chính là **phản gương** của nó; "3" và "E" là **phản gương** của nhau.

Một xâu vừa **đôi** vừa **phản** là một xâu có đặc điểm của cả 2 xâu **phản gương** và **đôi gương**. Xâu "ATOYOTA" là một xâu vừa **đôi** vừa **phản gương**. Tất nhiên là, "A", "T", "O", và "Y" chính là **phản gương** của nó.

Danh sách các ký tự và **phản gương** của nó (nếu có) như sau:

Ký tự	Phản	Ký tự	Phản	Ký tự	Phản
A	A	M	M	Y	Y
B		N		Z	5
C		O	O	1	1
D		P		2	S
E	3	Q		3	E
F		R		4	
G		S	2	5	Z
H	H	T	T	6	
I	I	U	U	7	
J	L	V	V	8	8
K		W	W	9	
L	J	X	X		

Chú ý: 0 (số 0) và O (ký tự O) được xem như là 1 ký tự, do đó chỉ có ký tự "O".

**Dữ liệu vào:** "DOIGUONG.INP"

File dữ liệu vào gồm một số xâu gồm từ một đến 20 ký tự (chỉ gồm các ký tự nằm trong danh sách trên), mỗi xâu ghi trên 1 dòng.

**Dữ liệu ra:** "DOIGUONG.OUT"

Ứng với mỗi xâu đầu vào, hiện ra 1 xâu theo định dạng sau:



Xâu	Ý nghĩa
" -- is not a palindrome."	Nếu xâu không phải là đối gương và không là phản gương
" -- is a regular palindrome."	Nếu xâu là đối gương và không phải là phản gương
" -- is a mirrored string."	Nếu xâu không phải là đối gương mà là phản gương
" -- is a mirrored palindrome."	Nếu xâu vừa là phản gương vừa là đối gương

**Ghi chú:**

Phần hiện file có bao gồm “--” chính xác như định dạng trên với phần trước là xâu đã cho (tham khảo ví dụ bên dưới).

Sau mỗi dòng output, bạn phải in một dòng trắng.

Ví dụ:

DOIGUONG.INP	DOIGUONG.OUT
NOTAPALINDROME	NOTAPALINDROME -- is not a palindrome.
ISAPALINILAPASI	ISAPALINILAPASI -- is a regular palindrome.
2A3MEAS	2A3MEAS -- is a mirrored string.
ATOYOTA	ATOYOTA -- is a mirrored palindrome.

Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1  file = open("DOIGUONG.INP")
2  file2 = open("DOIGUONG.OUT", "w")
```

Để giải bài toán, ta tạo danh sách **a**, **b**, **c**, **d** lần lượt lưu các ký tự như sau:

- + Danh sách **a**: Lưu các ký tự phản gương là chính nó.
- + Danh sách **b**: Lưu các ký tự không có phản gương.
- + Danh sách **c**: Lưu các ký tự có phản gương không phải chính nó.
- + Danh sách **d**: Lưu các ký tự là phản gương của danh sách **c**.

```

4  a = ["A", "H", "I", "M", "O", "T", "U", "V", "W", "X", "Y", "0",
5    "1", "8"]
6  b = ["B", "C", "D", "F", "G", "K", "N", "P", "Q", "R", "4", "6",
7    "7", "9", " "]
8  c = ["E", "J", "L", "S", "Z", "2", "3", "5"]
9  d = ["3", "L", "J", "2", "5", "S", "E", "Z"]

```

Để kiểm tra một chuỗi có phải đối gương hay không, ta tạo hàm **doiguong()**, với biến **s** là biến toàn cục lưu chuỗi cần kiểm tra. Trong hàm **doiguong()**, ta tạo biến **temp** lưu chuỗi đảo ngược của **s**, sau đó kiểm tra nếu **temp** bằng **s** thì trả về giá trị 1 – **s** là chuỗi đối gương, còn không thì trả về giá trị 0 – không là chuỗi đối gương.

```

9  def doiguong():
10    global s
11    temp = s[::-1]
12    if temp == s:
13      return 1
14    return 0

```

Để kiểm tra chuỗi **s** có phải chuỗi phản gương hay không, ta tạo hàm **phanguong()**. Trong hàm **phanguong()**, ta tạo hai biến **dau** và **cuoi** lưu vị trí đầu và cuối, biến đầu sẽ tăng từ 0 và cuối sẽ giảm từ vị trí cuối cùng của chuỗi.

```

16 def phanguong():
17   global s
18   dau = 0
19   cuoi = len(s) - 1

```

Khi giá trị biến **dau** chưa lớn hơn **cuoi**, ta thực hiện kiểm tra các trường hợp:

- + Ký tự tại vị trí **dau** và **cuoi** giống nhau. Khi này ta kiểm tra nếu ký tự đó không nằm trong danh sách **a** thì hàm trả về giá trị 0.
- + Ký tự tại vị trí **dau** nằm trong danh sách **b** nhưng ký tự tại vị trí **cuoi** không phải ký tự khoảng cách, hàm sẽ trả về giá trị 0.
- + Ký tự tại vị trí **dau** nằm trong danh sách **c** nhưng ký tự tại vị trí **cuoi** không phải là phản gương của ký tự đó, hàm sẽ trả về giá trị 0.

Sau mỗi lần kiểm tra, ta tăng giá trị **dau** và giảm giá trị **cuoi**. Nếu vòng lặp chạy tới khi **dau** lớn hơn **cuoi**, nghĩa là chuỗi **s** thỏa mãn điều kiện phản gương, hàm trả về giá trị 1.



```

21     while (dau ≤ cuoi):
22         if (s[dau] == s[cuoi]):
23             if not(s[dau] in a):
24                 return 0
25
26         elif (s[dau] in b):
27             if (s[cuoi] != ' '):
28
29                 return 0
30         elif (s[dau] in c):
31             i = c.index(s[dau])
32             if s[cuoi] != d[i]:
33                 return 0
34         dau += 1
35         cuoi -= 1
36     return 1

```

Trong đoạn chương trình chính, ta đọc chuỗi **s** từ file dữ liệu vào, sau đó lặp lại việc kiểm tra tới khi độ dài của **s** lớn hơn 0 (chuỗi quét được không phải chuỗi rỗng). Khi đọc dữ liệu từ file, biến **s** là chuỗi sẽ có thêm ký tự ngắt dòng “\n” phía cuối. Ta dùng hàm `rstrip()` để bỏ các ký tự nhất định phía cuối của chuỗi đi. Cụ thể trong trường hợp này là ký tự “\n”.

```

38 s = file.readline()
39
40 while len(s) > 0:
41     s = s.rstrip("\n")

```

Tiếp đó, thực hiện kiểm tra chuỗi **s** bằng cách gọi 2 hàm **doiguong()** và **phanguong()**, sau đó ghi dữ liệu ra của bài toán theo yêu cầu của đề bài. Lưu ý, mỗi test cần cách nhau một dòng, nên khi ghi dữ liệu ra ta cần ghi 2 ký tự “\n”.

```

43     if doiguong() == 1 and phanguong() == 1:
44         file2.write(str(s) + " -- is a mirrored palindrome.\n\n")
45     elif doiguong() == 1 and phanguong() == 0:
46         file2.write(str(s) + " -- is a regular palindrome.\n\n")
47     elif doiguong() == 0 and phanguong() == 1:
48         file2.write(str(s) + " -- is a mirrored string.\n\n")
49     else:
50         file2.write(str(s) + " -- is not a palindrome.\n\n")
51     s = file.readline()

```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```

53 file.close()
54 file2.close()

```

## Bài 5. Mã hóa và giải mã

Chịu trách nhiệm về phòng Công nghệ thông tin của Cơ quan tình báo quốc tế (Agency of International Espionage), bạn được yêu cầu thực hiện một chương trình cho phép các điệp viên mã hóa và giải mã các thông điệp.

Một thông điệp dài không quá 80 ký tự, bao gồm tất cả các chữ cái in và thường của bảng chữ cái và thêm ký tự khoảng cách và các ký tự trong bảng mã bên dưới.

Bảng ký tự ASCII của các ký tự có thể có trong thông điệp:

“A”	65	“a”	97		“ ”	32	“;”	59
“B”	66	“b”	98		“!”	33	“?”	63
...	...	...	...		“,”	44		
“Y”	89	“y”	121		“.”	46		
“Z”	90	“z”	122		“..”	58		

Thuật toán của bạn cần phải mã hóa thông điệp sử dụng giá trị ASCII của mỗi ký tự, bắt đầu từ ký tự cuối cùng và kết thúc ở ký tự đầu tiên. Sau đó bạn cần đảo ngược thứ tự giá trị ASCII trong thông điệp. Ví dụ, nếu giá trị ASCII là 123, thông điệp được mã hóa sẽ chứa chuỗi “321”. Không có ký tự khoảng cách trong thông điệp được mã hóa.

**Input “encode.inp”**

File input bao gồm một số dòng với thông điệp bình thường (chưa mã hóa) hoặc thông điệp đã mã hóa.

**Output “encode.out”**

File output có cùng số dòng với file input, mỗi dòng là một thông điệp, tương ứng với dòng thông điệp chưa mã hóa thì mã hóa nó và ngược lại.

**Ví dụ:**

<b>encode.inp</b>	<b>encode.out</b>
abc	998979
798999	cba
Have a Nice Day !	332312179862310199501872379231018117927

Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1  file = open("encode.inp")
2  file2 = open("encode.out", "w")
```



Để mã hóa dữ liệu, ta tạo hàm **maho()** với biến **s** là chuỗi cần mã hóa. Ta có thể chuyển các ký tự trong bảng chữ cái tiếng Anh thành mã của ký tự trong bảng mã ASCII bằng cách sử dụng hàm **ord()**. Bởi vậy, trong hàm **maho()**, ta chuyển đổi từng ký tự trong chuỗi **s**. Sau khi chuyển sang mã ASCII, ta tiến hành đảo ngược chuỗi **kq** và ghi kết quả vào file dữ liệu ra của bài toán.

```
4 def mahoa():
5     global s
6     kq = ""
7     for i in range(len(s)):
8         kq += str(ord(s[i]))
9     kq = kq[::-1]
10    print(kq)
11    file2.write(kq)
```

Trường hợp dữ liệu đọc từ file là các số, ta cần giải mã dữ liệu đọc được. Để giải mã, ta tạo hàm **giaima()** với **s** là chuỗi dữ liệu đọc từ file. Trước khi giải mã, ta cần đảo ngược **s**.

```
13 def giaima():
14     global s
15     s = s[::-1]
```

Trong bảng mã đề bài đưa ra, các ký tự có thể sẽ có 2 chữ số hoặc 3 chữ số. Với các số có 3 chữ số đều bắt đầu bằng số 1. Chính vì vậy ta kiểm tra nếu ký tự hiện tại là 1, ta tiến hành cộng với 2 ký tự tiếp theo trong chuỗi **s**. Còn không thì cộng với 1 ký tự tiếp theo trong chuỗi **s**.

```
16     kq = ""
17     i = 0
18     while i < len(s):
19         if s[i] == "1":
20             num = 100 + int(s[i+1])*10 + int(s[i+2])
21             i += 3
22         else:
23             num = int(s[i])*10 + int(s[i+1])
24             i += 2
```

Tiếp đó ta sử dụng hàm **chr()** để chuyển từ mã số cộng được sang ký tự. Sau đó ghi kết quả vào file dữ liệu ra của bài toán.

```
26         kq += chr(num)
27     print(kq)
28     file2.write(kq)
```

Trong đoạn chương trình chính, ta tiến hành đọc dữ liệu của bài toán tới khi hết dữ liệu. Tuy nhiên chúng ta cần lưu ý số dòng trong file dữ liệu ra sẽ bằng với số dòng trong file dữ liệu vào. Chính vì vậy ta cần ghi thêm ký tự xuống dòng “**\n**” phía trước. Nhưng trường hợp dòng đọc được là dòng đầu tiên ta không cần in thêm ký tự “**\n**” phía trước. Chính vì vậy ta tạo biến **firstline** và quy ước:

- + **firstline** = 1: Dòng hiện tại là dòng đầu tiên.
- + **firstline** = 0: Dòng hiện tại không phải là dòng đầu tiên.

Ban đầu biến **firstline** nhận giá trị 1, sau khi mã hóa hoặc giải mã xong dòng đầu tiên, biến **firstline** chuyển thành 0.

```
29 firstline = 1
30 s = file.readline()
31
32 while len(s) > 0:
33     if firstline == 0:
34         file2.write("\n")
```

Khi đọc dữ liệu vào, python sẽ đọc cả ký tự xuống dòng “\n”. Chính vì vậy ta cần kiểm tra nếu ký tự cuối cùng của chuỗi **s** là “\n”, ta bỏ ký tự cuối cùng trong chuỗi **s** đi. Sau đó kiểm tra chuỗi **s** có phải các ký tự số hay không bằng hàm **isdigit()**. Nếu dữ liệu là số, ta gọi hàm giải mã, còn không thì mã hóa.

```
36     if s[-1] == "\n":
37         s = s[:-1]
38     if s.isdigit():
39         gaiima()
40     else:
41         mahoa()
```

Sau khi gọi hàm, ta chuyển biến **firstline** thành 0 và đọc tiếp dữ liệu của bài toán. Khi đã hoàn thành mã hóa và giải mã hết dữ liệu trong file “endecode.inp”. Ta đóng 2 file dữ liệu của bài toán.

```
43     firstline = 0
44     s = file.readline()
45
46 file.close()
47 file2.close()
```



## Bài tập thực hành

### Bài tập 1. Đếm từ (Hậu Giang, 2020)

Viết chương trình cho phép nhập vào một đoạn văn bản (không rỗng) tiếng việt không dấu (kết thúc bằng dấu enter) sau khi nhập xong xuất ra màn hình kết quả đếm như sau:

Đoạn văn bản có bao nhiêu ký tự? bao nhiêu từ? Số lượng ký tự xuất hiện trong văn bản đó (Không phân biệt ký tự Hoa và thường, tính cả khoảng trắng, số và tất cả các dấu câu).

**Ví dụ:** Hoi thi.

Tổng số ký tự: 8

Tổng số từ: 2

Ký tự "h": 2

Ký tự "o": 1

Ký tự "i": 2

Ký tự "t": 1

Ký tự ".": 1

Ký tự " ": 1

### Bài tập 2. Tổng chữ số (Quảng Ngãi, 2020)

Cho số tự nhiên  $N$  ( $0 < N < 10^9$ ). Tính tổng của tất cả các chữ số của số tự nhiên  $N$ .

**Dữ liệu vào:** Tệp văn bản BL1.INP ghi số tự nhiên  $N$ .

**Dữ liệu ra:** Tệp văn bản BL1.OUT ghi 1 số là kết quả tìm được.

**Ví dụ:**

BL1.INP	BL1.OUT
2334	12

### Bài tập 3. Rút gọn xâu (TP. Tây Ninh, 2019)

Cho một xâu  $S$  chỉ gồm các chữ cái in thường với độ dài tối đa 250 ký tự. Em hãy viết chương trình để tạo ra xâu  $S1$  từ xâu  $S$  bằng cách xóa các ký tự liên tiếp giống nhau trong xâu  $S$  và chỉ để lại một ký tự đại diện trong đoạn đó.

**Dữ liệu:** Xâu  $S$  chỉ gồm các chữ cái in thường.

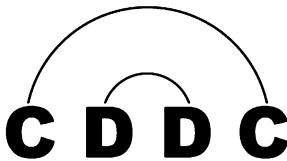
**Kết quả:** Xâu  $S1$  tìm được.

INPUT	OUTPUT
hhooooooooooooocccsssiinnnnnnnnnhhhh	hocsinh

#### Bài tập 4. Chuỗi huyền diệu (Bình Định, 2019)

Bình đang kết nối các cặp ký tự giống nhau có trong một chuỗi bằng cách vẽ các cung ở trên chuỗi đó. Một chuỗi được gọi là huyền diệu nếu mỗi ký tự có thể kết nối với một ký tự khác (ký tự giống nó) thỏa mãn điều kiện tất cả các cung không giao nhau.

Em hãy viết chương trình kiểm tra 1 chuỗi có là chuỗi huyền diệu hay không.



##### INPUT:

- + Dòng đầu tiên chứa giá trị  $N$  ( $N \leq 100$ ).
- +  $N$  dòng tiếp theo, mỗi dòng chứa một chuỗi có độ dài không vượt quá 50.

##### OUTPUT:

- + Dòng đầu tiên chứa giá trị  $M$  là tổng số chuỗi huyền diệu.
- +  $M$  dòng tiếp theo, mỗi dòng chứa một chuỗi huyền diệu.

INPUT	OUTPUT
5	2
ABAB	AABB
AABB	CDDCXX
CDDCXX	
ZLZP	
YYY	

#### Bài tập 5. Tìm xâu (Ninh Bình, 2019)

Nhập 2 xâu ký tự từ bàn phím, kiểm tra xem xâu 2 có nằm trong xâu 1 hay không, nếu có thì chỉ ra vị trí đầu tiên mà xâu 2 nằm trong xâu 1 và số lần xuất hiện của xâu 2.

Ví dụ: Nhập xâu 1 là “abcdefcd”, xâu 2 là “cd” thì thông báo xâu cd nằm trong xâu abcdefcd, vị trí đầu tiên xuất hiện của xâu 2 là 3, số lần xuất hiện của xâu 2 là 2 lần.



## Bài tập 6. Bignum2

Tính hiệu của 2 số nguyên dương, độ lớn của mỗi số có thể lên tới 100 chữ số.

**Dữ liệu vào:** “Bignum2.txt”

Dòng đầu tiên là một số nguyên dương NTEST là số lượng test được ghi trên một dòng. Mỗi bộ test gồm 2 dòng mỗi dòng ghi một số cần tính hiệu, số ghi trước là số bị trừ.

**Dữ liệu ra:**

Hiện ra NTEST số, mỗi số trên một dòng là hiệu của 2 số đã cho tương ứng.

INPUT	OUTPUT
2	12345678909
12345678910	22222222222222222
1	
33333333333333333	
11111111111111111	

## Bài tập 7. Dịch số 4.0

Cho vào một số số nguyên nằm trong khoảng âm 999.999.999 đến 999.999.999. Nhiệm vụ của bạn là dịch những số này thành dạng chữ bằng tiếng Anh. Danh sách sau là tất cả các từ tiếng Anh mà chương trình của bạn cần:

negative, zero, one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve, thirteen, fourteen, fifteen, sixteen, seventeen, eighteen, nineteen, twenty, thirty, forty, fifty, sixty, seventy, eighty, ninety, hundred, thousand, million

**Chú ý:**

- + Các số âm được bắt đầu bằng từ “negative”.
- + Từ “hundred” không được dùng khi “thousand” có thể dùng. Ví dụ, 1500 được viết là “one thousand five hundred”, chứ không phải là “fifteen hundred”.

INPUT	OUTPUT
6	six
-729	negative seven hundred twenty nine
1000101	one million one hundred one

### Bài tập 8. Xâu Palindrome (Trà Vinh, 2019)

Một S được gọi là xâu Palindrome nếu ta đọc xâu S từ trái sang phải cũng giống từ phải sang trái, ví dụ xâu 'madam'. Từ một xâu người ta có thể tạo ra xâu mới bằng cách đẩy vòng một số lần: đưa ký tự cuối xâu về ghi ở đầu xâu. Ví dụ, từ xâu 'array', bằng cách đẩy vòng ta có thể nhận được các xâu:

array -> yarra -> ayarr ->rayar -> rraya

Trong số các xâu nhận được có một xâu là Palindrome. Trong trường hợp này người ta nói 'array' là một xâu palindrome vòng. Bản thân xâu palindrome cũng là xâu palindrome vòng (với số lần đẩy vòng bằng 0).

**Yêu cầu:** Cho xâu S không quá 100 ký tự. Hãy xác định xem S có phải là xâu Palindrome vòng hay không.

**Dữ liệu vào:** Gồm một dòng chứa xâu S.

**Dữ liệu ra:** Ghi số 1 nếu S là xâu Palindrome vòng, ghi số 0 nếu S không là xâu Palindrome vòng.

INPUT	OUTPUT
array	1
mama	0

### Bài tập 9. Nối xâu (Vị Thanh – Hậu Giang, 2019)

Nhập từ bàn phím xâu S có chiều dài n ( $3 < n < 20$ ), biết rằng xâu S chỉ chứa các ký tự thường trong bảng chữ cái tiếng Anh. Từ xâu S ta sinh ra xâu S1 bằng cách thêm bên trái xâu S một hoặc một số ký tự sao cho xâu S1 là xâu đối xứng ngắn nhất.

Dữ liệu vào	Dữ liệu ra
Nhập xâu S: abcde	abculedcba

### Bài tập 10. Xâu đối xứng (TP. Vinh, 2019)

Khi học lập trình Nam rất thích xử lý trên xâu ký tự, nhưng cuộc thi Tin học trẻ năm nay có bài toán khá hóc búa với yêu cầu sau: Cho trước một xâu ký tự bao gồm các chữ cái in thường có độ dài không quá 255, hãy đưa ra xâu con đối xứng dài nhất. Nếu có nhiều xâu con thỏa mãn yêu cầu thì đưa ra tất cả các xâu con đó (Xâu con là xâu chứa liên tiếp các ký tự trong xâu đã cho; xâu đối xứng là xâu đọc từ trái sang phải hay từ phải sang trái đều như nhau VD: abcba). Em hãy giúp Nam giải quyết bài toán trên.

**Dữ liệu vào:** Nhập xâu vào danh sách từ file Xaunhap.txt hoặc nhập trực tiếp.

**Dữ liệu ra:** Màn hình hiển thị danh sách các xâu đối xứng.



## DẠNG BÀI MẢNG MỘT CHIỀU, DANH SÁCH

Mảng là một tập hợp các phần tử có cùng một kiểu dữ liệu (kiểu số nguyên, chuỗi,...), các phần tử trong mảng có chỉ mục tăng dần từ 0. Để sử dụng mảng, ta cần import thư viện array vào dự án.

Không giống như mảng, mỗi danh sách - list có thể lưu trữ phần tử với bất kỳ kiểu dữ liệu nào và làm được mọi thứ mà mảng có thể làm. Chúng ta có thể lưu trữ số nguyên, số thập phân, chuỗi trong cùng một list.

Trong cuốn sách sẽ hướng dẫn các bạn sử dụng mảng hoặc danh sách để giải một số bài toán cần lưu trữ nhiều thông tin, dữ liệu.

### Bài 1. Đặt hàng (TP. Vũng Tàu, 2020)

Trên trang web bán hàng trực tuyến của một cửa hàng điện thoại di động có trưng bày các mẫu điện thoại, mỗi mẫu được gán một mã số để phân biệt là các số nguyên dương từ 1 đến  $n$ . Trong một tháng, có  $N$  khách hàng truy cập đến trang web để xem và đặt mua điện thoại. Mỗi khách hàng có thể đặt mua một mẫu điện thoại bất kì. Vào cuối tháng, nhân viên cửa hàng phải tổng hợp các đơn đặt hàng để biết được mẫu điện thoại được khách hàng ưa chuộng nhất (mẫu điện thoại được ưa chuộng nhất là mẫu có số lượng đặt mua nhiều nhất).

#### Yêu cầu:

Hãy giúp nhân viên cửa hàng đếm số lượng được đặt mua của mẫu điện thoại được ưa chuộng nhất trong tháng.

#### Dữ liệu vào:

File ORDER.INP có nội dung như sau:

- Dòng thứ nhất chứa số nguyên dương  $N$  ( $N \leq 10^6$ ).
- Trong  $n$  dòng tiếp theo, dòng thứ  $i$  chứa số nguyên dương  $a_i$  ( $a_i \leq 10^5$ ).

#### Kết quả:

Ghi vào File ORDER.OUT một số nguyên duy nhất là số lượng đặt mua của mẫu điện thoại được ưa chuộng nhất.

<b>ORDER.INP</b>	<b>ORDER.OUT</b>
4	
3	
2	
3	
1	2

Trước tiên, ta import thư viện array vào dự án để có thể sử dụng mảng và lập trình mở 2 file dữ liệu của bài toán.

```
1 from array import *
2
3 file = open("ORDER.INP")
4 file2 = open("ORDER.OUT", "w")
```

Tiếp đó ta đọc dữ liệu dòng đầu tiên từ tệp tin dữ liệu vào, sau đó khởi tạo mảng **a** để lưu các mẫu điện thoại đã được đặt mua và mảng **dem** để lưu số lượng mẫu điện thoại bán được. Trong mảng **dem** ta quy ước phần tử đầu tiên của mảng lưu số lượng bán được của mẫu số 0, phần tử thứ hai của mảng lưu số lượng bán được của mẫu số 1, ....

Mảng dem	index	Quy ước
	0	Số lượng bán ra của mẫu điện thoại 0
	1	Số lượng bán ra của mẫu điện thoại 1
	2	Số lượng bán ra của mẫu điện thoại 2
	...	...
	n	Số lượng bán ra của mẫu điện thoại n

```
6 n = int(file.readline())
7 a = array('i')
8 dem = array('i')
```

Tiếp đó, ta thực hiện việc đọc **n** dòng tiếp theo trong file dữ liệu vào. Với mỗi dòng dữ liệu, ta lưu vào biến **temp** sau đó thêm **temp** và mảng **a** bằng phương thức **append()**.

```
10 for i in range(n):
11     temp = int(file.readline())
12     a.append(temp)
```





### Bạn có biết: Thêm phần tử vào cuối mảng/danh sách

Ta có thể thêm phần tử vào vị trí cuối cùng trong mảng bằng hai phương thức sau:

- + `append()`: Thêm 1 phần tử vào cuối mảng/danh sách.
- + `extend()`: Thêm nhiều phần tử vào cuối mảng/danh sách.

Cú pháp:

```
name.append(element)
name.extend(iterable)
```

Trong đó:

- `name`: Tên mảng/danh sách cần thêm phần tử.
- `element`: phần tử muốn thêm vào mảng/danh sách.
- `iterable`: các phần tử muốn thêm, mỗi phần tử cách nhau bởi dấu phẩy ','. Ví dụ: `a.extend([1, 4, 6])`.

**Yêu cầu:** Đối với mảng, phần tử muốn thêm cần có kiểu dữ liệu giống với kiểu dữ liệu của các phần tử trong mảng.

Mảng **dem** được sử dụng để lưu số lượng điện thoại được bán ra tương ứng với từng mẫu, vậy ban đầu ta đặt tất cả số lượng bán được là 0, sau đó duyệt từng phần tử trong mảng **a** và tăng số lượng bán được trong mảng **dem** tương ứng.

Tuy nhiên, ta cần tìm giá trị lớn nhất của mã điện thoại đã bán được để từ đó thêm số lượng các phần tử 0 vào mảng **dem**.

```
14 m = max(a)
15 for i in range(m+1):
16     dem.append(0)
```

Sau khi tạo được mảng **dem** ban đầu gồm các giá trị 0, ta thực hiện duyệt từng phần tử trong mảng **a** và tăng giá trị tại vị trí tương ứng bên mảng **dem**.

```
18 for i in range(n):
19     j = a[i]
20     dem[j] += 1
```

Sau khi thực hiện đếm các mẫu điện thoại đã bán được, giá trị lớn nhất trong mảng **dem** chính là kết quả của bài toán.

```
22 kq = max(dem)
23 file2.write(str(kq))
24
25 file.close()
26 file2.close()
```



### Bạn có biết: Hàm max()

Khi sử dụng hàm max() với mảng gồm các phần tử thuộc kiểu dữ liệu số (int, float,...), hàm sẽ trả về cho chúng ta số lớn nhất.

Ví dụ: a = [1, 2, 3, 5, 4, 21] max(a) trả về giá trị 21

Khi sử dụng hàm max() với mảng gồm các phần tử thuộc kiểu dữ liệu chuỗi (str), hàm sẽ trả về giá trị lớn nhất được sắp xếp theo thứ tự bảng chữ cái.

Ví dụ: a = ["1", "2", "3", "5", "4", "21"] max(a) trả về giá trị "5"

## Bài 2. Dãy Fibonacci (Yên Bá, 2020)

Dãy số Fibonacci là dãy số có hai số hạng đầu tiên bằng 1, các số hạng tiếp theo từ số hạng thứ 3 trở đi bằng tổng của hai số hạng đứng trước nó. Một trong những số hạng đầu tiên của dãy Fibonacci là:

1, 1, 2, 3, 5, 8, 13, 21, ...

Cho trước số nguyên dương N ( $1 \leq N \leq 100$ ).

**Yêu cầu:** Xác định N có phải là số Fibonacci hay không?

**Dữ liệu vào:** Nhập từ bàn phím số nguyên dương N.

**Kết quả ra:** Nếu N là số Fibonacci in ra chữ YES, nếu không phải in ra chữ NO.

**Ví dụ 1:**

Dữ liệu nhập vào
8

Kết quả xuất ra
YES

**Ví dụ 2:**

Dữ liệu nhập vào
9

Kết quả xuất ra
NO



Để giải bài toán, ta sẽ lưu các số trong dãy Fibonacci vào trong mảng, sau đó kiểm tra giá trị nhập vào có nằm trong mảng hay không. Nếu có in lên màn hình chữ YES, ngược lại in ra NO.

Trước tiên, ta cần import thư viện array vào chương trình, sau đó yêu cầu nhập vào số nguyên N.

```
1 from array import *
2 n = input("Nhập vào số nguyên dương N: ")
3 n = int(n)
```

Ta tạo mảng fibo với kiểu dữ liệu int và có 2 phần tử ban đầu là 1. Sau đó tạo biến để lưu vị trí số cần tính giá trị và biến a để lưu giá trị số hiện tại trong dãy fibo. Hai vị trí bắt đầu trong dãy số là 2 giá trị 1, vậy ta cần tính giá trị từ vị trí số 3. Tuy nhiên trong mảng, vị trí đầu tiên sẽ có chỉ số là 0. Vậy ta đặt giá trị i ban đầu là 2, tương ứng vị trí có chỉ mục 2 trong mảng. Giá trị ban đầu trong dãy fibo là 1, vậy ta đặt biến a nhận giá trị 1.

```
4 fibo = array("i", [1,1])
5 i = 2
6 a = 1
```

Ta thực hiện việc tính các giá trị trong dãy fibonacci bằng cách đặt a thành tổng hai giá trị có chỉ số – 1 và – 2 trong mảng, thêm giá trị tính được vào mảng và tăng giá trị i thêm 1. Ta thực hiện việc tính như vậy khi giá trị a nhỏ hơn n.

```
8 while a < n:
9     a = fibo[i-1] + fibo[i-2]
10    fibo.append(a)
11    i += 1
```

Cuối cùng, ta thực hiện so sánh phần tử cuối cùng trong mảng fibo có phải giá trị n hay không và in ra màn hình kết quả phù hợp. Tuy nhiên giá trị i cuối cùng khi thoát vòng lặp ta đã cộng thêm 1 giá trị, chính vì vậy khi so sánh ta cần giảm i đi 1 giá trị.

```
12 if fibo[i-1] == n:
13     print("YES")
14 else:
15     print("NO")
```

### Bài 3. Đếm số nguyên tố (An Giang, 2020)

Tìm số Nguyên tố có số lần xuất hiện nhiều nhất trong dãy số. Viết chương trình theo yêu cầu sau:

**Input:** Nhập vào từ tập tin NGUYENTO.INP gồm 2 dòng:

+ Dòng 1 là số tự nhiên  $N$  (Với  $10 \leq N \leq 40$ ).

+ Dòng 2 gồm  $N$  số tự nhiên, mỗi số cách nhau ít nhất một ký tự trắng.

**Output:** Ghi vào tập tin NGUYENTO.OUT gồm 02 (hai) số tự nhiên là số nguyên tố có số lần xuất hiện nhiều nhất và số lần xuất hiện tương ứng, cách nhau ít nhất một ký tự trắng.

Lưu ý: Nếu không tìm được kết quả theo yêu cầu đề bài thì thông báo “KHONG TIM DUOC”.

NGUYENTO.INP	NGUYENTO.OUT
14 9 52 11 11 52 11 9 6 9 11 52 52 6 5	11 4

**Phân tích:** Để giải bài toán, ta tạo một mảng (ví dụ: **dem**) để lưu số lần xuất hiện của các số từ 0 tới số lớn nhất trong dãy số. Ban đầu các phần tử trong **dem** ta đặt là 0, sau đó duyệt từng phần tử trong dãy số. Nếu số đó là số nguyên tố thì ta tăng thêm 1 giá trị có vị trí tương ứng trong mảng **dem**. Cuối cùng, ta kiểm tra giá trị lớn nhất trong mảng **dem**. Nếu giá trị lớn nhất bằng 0 nghĩa là trong dãy số không có số nguyên tố, ta xuất ra kết quả là “KHONG TIM DUOC”, trường hợp còn lại ta xuất ra kết quả theo yêu cầu của đề bài.

Trước tiên, ta import thư viện math và array vào chương trình, sau đó thực hiện mở file dữ liệu vào và dữ liệu ra của bài toán.

```
1 from array import *
2 from math import *
3 file = open("NGUYENTO.INP")
4 file2 = open("NGUYENTO.OUT", "w")
```

Tiếp đó, ta tạo hàm kiểm tra tính nguyên tố của một số truyền vào.

```
6 def nguyento(num):
7     if num == 1:
8         return False
9     else:
10        for i in range(2, int(sqrt(num)+1)):
11            if num % i == 0:
12                return False
13        return True
```



Sau khi tạo hàm, ta thực hiện việc quét dòng dữ liệu đầu tiên và lưu vào biến **n**. Với dòng dữ liệu thứ 2, ta lưu vào biến **s**, sau đó sử dụng phương thức `split()` để tách chuỗi **s**. Kết quả sau khi tách ta lưu vào danh sách **data**. Tiếp đó ta khởi tạo mảng **dem** với kiểu dữ liệu là số nguyên (int).

```
15 n = int(file.readline())
16 s = file.readline()
17 data = s.split()
18 dem = array("i")
```



### Bạn có biết: Phương thức `split()`

Ta có thể tách một **chuỗi** thành một **danh sách** với phương thức `split()`. Ta có thể chỉ định dấu phân cách khi tách chuỗi, dấu phân cách mặc định là ký tự khoảng cách.

Cú pháp:

```
string.split(separator, maxsplit)
```

Trong đó:

- + string: Chuỗi cần phân tách
- + separator: Chỉ định dấu phân tách, nếu không điền gì sẽ mặc định là ký tự khoảng trắng.
- + maxsplit: Chỉ định số lượng phần tử cần phân tách, giá trị mặc định là -1 tương ứng tách tất cả chuỗi.

Để tìm giá trị lớn nhất trong danh sách, ta không thể sử dụng hàm `max()` vì danh sách không quy ước kiểu dữ liệu số hay chữ. Vậy nên khi sử dụng hàm `max()` trong danh sách, chương trình sẽ trả về cho chúng ta kết quả so sánh theo kiểu ký tự. Ví dụ danh sách gồm các giá trị `[9, '12', '6', '23']`, khi sử dụng hàm `max()` sẽ trả về cho chúng ta kết quả là `'9'`.

Vậy để tìm giá trị lớn nhất, ta tạo biến **m** lưu giá trị đầu tiên trong danh sách **data**. Tiếp đó, ta thực hiện so sánh **m** với từng phần tử trong danh sách. Khi so sánh, ta cần ép kiểu int cho phần tử lấy ra so sánh.

```
20 m = int(data[0])
21 for i in range(1, n):
22     if m < int(data[i]):
23         m = int(data[i])
```

Khi đã tìm được giá trị lớn nhất, ta thực hiện thêm các giá trị 0 vào mảng **dem**.

```
25 for i in range(m+1):
26     dem.append(0)
```

Sau khi thêm các giá trị 0 vào mảng **dem**, ta duyệt từng phần tử trong **data** và thực hiện đếm các số nguyên tố trong **data**.

```
27 for i in range(n):
28     a = int(data[i])
29     if nguyento(a):
30         dem[a] += 1
```

Tiếp theo, ta đặt **m** thành giá trị lớn nhất của mảng **dem**. Vì mảng **dem** ta khai báo với kiểu dữ liệu số nên ta có thể sử dụng hàm **max()** để biết giá trị lớn nhất của mảng.

```
32 m = max(dem)
```

Cuối cùng, ta kiểm tra nếu **m** bằng 0 thì ghi vào file “KHONG TIM DUOC”, còn không thì ghi vào vị trí có giá trị lớn nhất trong mảng **dem**. Để tìm vị trí của một phần tử trong mảng, ta sử dụng phương thức **index()**. Sau khi ghi dữ liệu, ta thực hiện việc đóng 2 file dữ liệu.

```
34 if m == 0:
35     file2.write("KHONG TIM DUOC")
36 else:
37     file2.write(str(dem.index(m)) + " " + str(m))
38
39 file.close()
40 file2.close()
```

#### Bài 4. Cấp số cộng (Sóc Trăng, 2019)

Cho một dãy A gồm N số nguyên  $a_1, a_2, \dots, a_N$ . Một dãy B được tạo ra từ dãy A bằng cách chọn 1 hay một số phần tử của dãy A và giữ nguyên thứ tự như ban đầu. Hãy tìm trong dãy A một dãy con B dài nhất lập thành một cấp số cộng.

**Dữ liệu vào gồm:**

- + Dòng đầu ghi 2 số nguyên dương  $N$  ( $N \leq 1000$ ) và công sai  $D$  ( $D < 100$ ).
- + Dòng tiếp theo lần lượt ghi  $N$  số  $a_1, a_2, \dots, a_N$ . ( $a_i \leq 30000$  với  $i = 1, 2, \dots, n$ ).

**Kết quả xuất ra:**

- + Dòng đầu ghi số  $M$  là số phần tử của cấp số cộng tìm được.
- + Các dòng tiếp theo lần lượt ghi  $M$  số là chỉ số của các số hạng của dãy A thuộc cấp số cộng, mỗi dòng ghi 10 số, trừ hàng cuối cùng có thể ghi ít hơn. Các số trên cùng một dòng cách nhau bởi dấu cách.



Ví dụ:

INPUT	OUTPUT
10 2 1 2 3 -6 3 8 5 6 7 -4	4 1 3 7 9

Trong bài toán, chúng ta cần tìm dãy cấp số cộng dài nhất với công sai cho trước. Để tìm được dãy cấp số cộng dài nhất, ta tiến hành tìm dãy cấp số cộng từ số thứ trong dãy số. Với i chạy từ vị trí đầu tới vị trí cuối dãy số.

Trước tiên, chúng ta import thư viện array, thư viện math, mở 2 file dữ liệu của bài toán và lập trình đọc dữ liệu dòng đầu tiên gồm số nguyên dương N và công sai D. Vì 2 giá trị này được viết trên cùng một dòng và cách nhau bởi dấu cách. Vậy sau khi đọc dữ liệu, ta tách chuỗi dữ liệu đọc được thành danh sách và gán giá trị biến **n** là phần tử đầu tiên, gán giá trị biến **d** là phần tử thứ 2.

```

1 from array import *
2 from math import *
3
4 file = open("CAPSOCONG.INP")
5 file2 = open("CAPSOCONG.OUT", "w")
6
7 data = file.readline()
8 s = data.split()
9 n = int(s[0])
10 d = int(s[1])

```

Tiếp đó, ta tiến hành đọc dòng dữ liệu thứ 2 là dãy **a**. Sau khi đọc dữ liệu, ta cũng tách chuỗi đọc được thành danh sách **a** bằng phương thức split(). Sau khi có được dãy **a**, ta tạo mảng **b** để lưu vị trí các giá trị thỏa mãn là một dãy cấp số cộng với công sai **d** cho trước. Tạo biến **max\_leng** để lưu độ dài của dãy cấp số cộng dài nhất tìm được, ban đầu biến **max\_leng** ta sẽ đặt là 0.

```

12 data = file.readline()
13 a = data.split()
14 b = []
15 max_leng = 0

```

Để xét từng vị trí có thể bắt đầu của dãy cấp số cộng, ta sử dụng vòng lặp hữu hạn **for i in range(n)**. Với mỗi vị trí **i**, ta tạo biến **dem** để lưu số lượng giá trị trong dãy cấp số cộng từ vị trí . Biến **dem** ban đầu ta đặt là 1, tương ứng có 1 giá trị trong dãy cấp số cộng (chính là giá trị tại vị trí ).

```

17 for i in range(n):
18     dem = 1

```

Vì trong dãy cấp số cộng, giá trị đầu tiên trong dãy số là giá trị tại vị trí  $i$ . Chính vì vậy ta thêm vị trí của giá trị đầu tiên vào mảng  $b$ . Tuy nhiên do vị trí bắt đầu của mảng là vị trí 0, mà trong dãy số vị trí các số lại bắt đầu từ 1. Bởi vậy ta cần thêm giá trị  $i + 1$  vào mảng  $b$ . Ngoài ra, trước khi thêm vào mảng  $b$ , ta thực hiện xóa tất cả các phần tử hiện tại của  $b$  đi bằng phương thức `clear()`.

```
19     b.clear()
20     b.append(i+1)
```

Tiếp theo, ta tạo biến  $temp$  để lưu giá trị số hiện tại trong dãy cấp số cộng tìm được, số đầu tiên là số tại vị trí  $i$  trong mảng  $a$ , vậy ban đầu ta đặt  $temp$  thành  $a[i]$ . Sau đó ta duyệt từng giá trị phía sau từ vị trí  $i$  trở đi. Nếu số đó bằng  $temp + d$  vây giá trị đó nằm trong dãy cấp số cộng. Khi đó ta thêm vị trí đó vào mảng  $b$ , tăng biến  $dem$  thêm 1 và thay đổi giá trị  $temp$  thành giá trị tại vị trí đó.

```
21     temp = int(a[i])
22     for j in range(i+1, n):
23         if temp + d == int(a[j]):
24             b.append(j+1)
25             dem += 1
26             temp += d
```

Sau khi duyệt hết dãy  $a$  từ vị trí  $i$ , ta kiểm tra dãy cấp số cộng tìm được có độ dài lớn hơn  $max\_leng$  hay không. Nếu lớn hơn, ta đặt kết quả là dãy số được lưu trong mảng  $b$  và đặt lại giá trị  $max\_leng$  thành  $dem$ .

```
27     if max_leng < dem:
28         kq = b.copy()
29         max_leng = dem
```

Khi đã duyệt hết các vị trí  $i$  của dãy số, ta tạo biến  $s$  để lưu chuỗi vị trí các số thuộc cấp số cộng dài nhất tìm được. Để nối các phần tử trong mảng lại với nhau, ta sử dụng phương thức `join()`.

```
30 s = ' '.join([str(i) for i in kq])
```

Cuối cùng, ta tiến hành việc ghi dữ liệu ra của bài toán. Dòng đầu tiên là số phần tử của cấp số cộng tìm được chính là giá trị  $max\_leng$ . Để xuống dòng khi ghi dữ liệu vào trong file, ta thực hiện ghi với ký tự '\n', ký tự này Python sẽ ngầm hiểu đây là ký hiệu ngắt dòng (new line).

Tại dòng thứ 2 của tệp dữ liệu ra là vị trí các số thuộc cấp số cộng tìm được, chuỗi này chính là chuỗi  $s$  chúng ta đã nối lại bằng phương thức `join()` bên trên. Vậy ở dòng thứ 2 ta chỉ việc ghi vào file chuỗi  $s$ .

```
32 file2.write(str(max_leng) + '\n')
33 file2.write(s)
```



Cuối cùng, ta thực hiện đóng 2 file dữ liệu sau khi đã hoàn thiện.

```
34 file.close()
35 file2.close()
```

### Bài 5. Phát khẩu trang (Bình Dương, 2020)

Kể từ khi được công bố là “Đại dịch toàn cầu” bởi WHO vào cuối tháng 1/2020, virus corona (COVID-19) đã có tác động không nhỏ đến tâm lý của một bộ phận người dân trong xã hội. Mỗi một nước đều có một cách phòng chống dịch bệnh khác nhau. Nước Thông Thái cũng có chính sách phòng chống dịch bệnh của riêng họ. Để đơn giản hóa, nhà của người dân được đặt trên một đường thẳng, đánh số từ 0 đến  $+\infty$ . Nước Thông Thái có rất nhiều nhà máy sản xuất khẩu trang. Với mỗi nhà máy tại vị trí  $x_i$  có thể cung cấp khẩu trang y tế cho một khu vực với bán kính  $r_i$ . Nhà nước muốn biết còn bao nhiêu điểm chưa được cung cấp khẩu trang y tế và bạn là một người được giao nhiệm vụ tính toán số địa điểm chưa được cung cấp khẩu trang y tế.

#### Dữ liệu vào:

- + Dòng đầu tiên chứa số nguyên dương  $n$  ( $n \leq 1000$ ) là số địa điểm sản xuất.
- +  $N$  dòng tiếp theo chứa hai cặp số  $x_i$  và  $r_i$  ( $1 \leq x_i, r_i \leq 10^6$ ).

#### Dữ liệu ra:

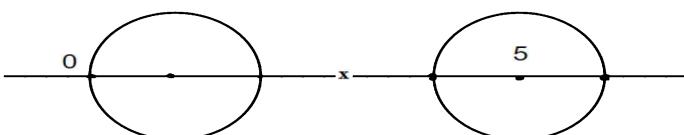
In một số nguyên dương – Tổng các địa điểm chưa được cung cấp khẩu trang từ vị trí 0 đến vị trí cung cấp khẩu trang cuối cùng.

#### Ví dụ

Dữ liệu vào	Dữ liệu ra
2	1
1 1	
5 1	
1	1
2 1	

#### Giải thích:

Ở ví dụ này, chỉ có địa điểm 3 chưa được cung cấp khẩu trang từ điểm 0 đến điểm 5.



Hướng đê giải bài toán chính là ta sẽ tạo một mảng để lưu trạng thái của các địa điểm. Ban đầu các phần tử trong mảng đều có giá trị 0 – chưa được cung cấp khâu trang. Với mỗi nhà máy, ta tiến hành thay đổi giá trị trong mảng của các khu vực nằm trong bán kính thành 1 – đã được cấp phát khâu trang. Cuối cùng ta thực hiện đêm xem trong mảng có bao nhiêu giá trị 0 tương ứng là số khu vực chưa được cấp phát khâu trang và cũng chính là kết quả của bài toán.

Trước tiên, ta import thư viện array và mở 2 file dữ liệu của bài toán.

```
1 from array import *
2 file = open("PHATKHAUTRANG.INP")
3 file2 = open("PHATKHAUTRANG.OUT", "w")
```

Ta tạo các mảng **x** và **r** để lưu vị trí nhà máy cũng như bán kính có thể cấp phát khâu trang của từng nhà máy. Tạo mảng **a** để lưu trạng thái cấp phát khâu trang của các khu vực. Tạo biến **m** lưu vị trí của nhà máy lớn nhất được cung cấp trong bài toán.

```
5 x = []
6 r = []
7 a = []
8 m = 0
```

Sau khi tạo mảng và biến, ta đọc giá trị **n** từ file và lưu vào biến **n**. Tiếp đó đọc **n** dòng dữ liệu tiếp theo. Với mỗi dòng dữ liệu, sau khi đọc ta cần tách thành danh sách bằng phương thức `split()`. Sau đó thêm giá trị tương ứng vào mảng **x**, **r** tương ứng vị trí nhà máy và bán kính nhà máy.

```
10 n = int(file.readline())
11 for i in range(n):
12     temp = file.readline()
13     temp = temp.split()
14     x.append(int(temp[0]))
15     r.append(int(temp[1]))
```

Với mỗi vị trí nhà máy quét được, ta so sánh với giá trị **m** – vị trí lớn nhất trong danh sách các nhà máy. Nếu vị trí nhà máy vừa đọc từ file dữ liệu lớn hơn **m**, ta đặt lại giá trị **m** thành vị trí nhà máy đó.

```
16     if m < int(temp[0]):
17         m = int(temp[0])
```

Khi đã tìm được giá trị cũng như vị trí lớn nhất trong các nhà máy, ta tiến hành thêm các giá trị 0 vào mảng **a**, bao gồm cả vị trí lớn nhất của nhà máy **m**.

```
19 for i in range(m+1):
20     a.append(0)
```



Tiếp đó ta xét vị trí và bán kính của từng nhà máy và chuyển đổi trạng thái các vị trí nằm trong bán kính thành 1 tương ứng trạng thái có thể cung cấp khâu trang. Tuy nhiên khi chuyển đổi giá trị trong mảng **a**, ta cần lưu ý vị trí sau khi tăng hoặc giảm có vượt quá kích thước của mảng hay không. Ví dụ khi nhà máy ở vị trí 1, bán kính là 2, nếu giảm đi với bán kính 2 sẽ có thể là 0 và -1. Tuy nhiên trong mảng không tồn tại vị trí có chỉ số -1. Vậy trước khi thay đổi giá trị phần tử trong mảng **a**, ta cần kiểm tra giá trị được trừ đi phải lớn hơn hoặc bằng 0. Trường hợp vị trí nhà máy là vị trí lớn nhất trong danh sách, khi xét các khu vực phía bên phải (khi tăng lên) thì trong mảng **a** ta chỉ thêm các phần tử 0 tới vị trí **m** cũng chính là vị trí của nhà máy đó, như vậy sẽ không tồn tại các giá trị phía sau **m**. Do vậy, trước khi thay đổi giá trị phần tử trong mảng **a**, ta cần kiểm tra vị trí đó phải nhỏ hơn hoặc bằng **m**, ta mới thực hiện thay đổi giá trị.

```
22 for i in range(n):
23     temp = x[i]
24     for j in range(r[i]+1):
25         if (temp - j) >= 0:
26             a[temp-j] = 1
27         if (temp + j) <= m:
28             a[temp+j] = 1
```

Sau khi chuyển đổi các vị trí từ chưa được cấp phát khâu trang thành được cấp phát khâu trang, ta thực hiện đếm số lượng phần tử 0 trong mảng **a** tương ứng số lượng khu vực chưa được cấp phát khâu trang, sau đó ghi kết quả ra file dữ liệu ra của bài toán.

Cuối cùng đóng 2 file dữ liệu sau khi đã hoàn thành đọc và ghi dữ liệu.

```
30 kq = str(a.count(0))
31 file2.write(kq)
32 file.close()
33 file2.close()
```

## Bài tập thực hành

### Bài tập 1. Cấp số nhân (Hậu Giang, 2020)

Cấp số nhân là một dãy số (hữu hạn hay vô hạn) mà trong đó, kể từ số hạng thứ hai, mỗi số hạng đều bằng tích của số hạng đứng ngay trước nó và một số q không đổi.

Viết chương trình cho phép nhập vào mảng số nguyên dương A gồm n phần tử ( $n > 2$ ). Kiểm tra xem các phần tử vừa nhập có phải là dãy số cấp số nhân hay không? Nếu là cấp số nhân thì in ra màn hình số q, sau đó cho phép tính giá trị số hạng thứ m, với m nhập vào từ bàn phím (m là số tự nhiên  $10 < m < 100$ ).

**Ví dụ:**

2, 4, 8, 16 là một cấp số nhân có  $q = 2$ ;

Nhập số  $m = 11$

Số hạng thứ 11 là: 2048

### Bài tập 2. Mảng một chiều (TP. Đà Lạt, 2019)

Em hãy tạo một danh sách mList và nhập trực tiếp thành 1 dãy số, sau đó:

- Thông báo dãy số vừa tạo lên màn hình.
- Tìm giá trị lớn nhất của dãy số và in lên màn hình.
- Sắp xếp dãy số theo chiều tăng dần và thông báo lên màn hình.

### Bài tập 3. Dãy con độ dài D (Bắc Giang, 2020)

Cho dãy số A gồm n số nguyên  $A_i$  ( $i = 1, 2, \dots, n$ ) và số nguyên dương D. Một dãy con là tập hợp các phần tử liên tiếp từ vị trí l đến vị trí thứ r nào đó trong dãy số A và có độ dài là  $r - l + 1$ .

**Yêu cầu:** Hãy tìm dãy con độ dài D có tổng các phần tử lớn nhất.

**Dữ liệu vào:** đọc từ file văn bản gồm:

+ Dòng 1: ghi 2 số nguyên dương n và D ( $D \leq n \leq 10^6$ );

+ Dòng 2: ghi n số nguyên  $A_i$  ( $|A_i| \leq 10^9$ ).

**Kết quả:** ghi ra một số duy nhất là giá trị tổng lớn nhất có thể của dãy con độ dài D trong dãy số A.

INPUT	OUTPUT
5 2 4 2 -8 9 1	10



### Bài tập 4. Đếm cặp (Ninh Bình 2019)

Cho số nguyên dương  $N$  và mảng  $A$  gồm  $N$  số nguyên dương  $a_1, a_2, \dots, a_n$  ( $2 \leq N \leq 10^5$ ,  $1 \leq a_i \leq 10^5$ ).

**Yêu cầu:** Đếm số cặp  $(a_i, a_j)$  thỏa mãn  $a_i = a_j$  ( $1 \leq i, j \leq N$ ,  $i \neq j$ ,  $(a_i, a_j)$  và  $(a_j, a_i)$  chỉ được tính là một cặp).

**Dữ liệu vào:** Đọc từ file văn bản DC.INP gồm:

- + Dòng thứ nhất chứa số nguyên dương  $N$ ;
- + Dòng thứ hai chứa  $N$  số nguyên dương  $a_1, a_2, \dots, a_N$ .

**Kết quả:** Ghi ra tệp văn bản DC.OUT số lượng cặp thỏa mãn.

Ví dụ:

INPUT	OUTPUT	Giải thích
7 6 2 4 2 4 3 4	4	Có 4 cặp số bằng nhau là: $a_2 = a_4 = 2$ $a_3 = a_5 = 4$ $a_3 = a_7 = 4$ $a_5 = a_7 = 4$

### Bài tập 5. Số âm trong dãy (Tây Ninh, 2019)

Cho một dãy số gồm  $N$  số nguyên  $a_1, a_2, \dots, a_N$ , mỗi số có giá trị tuyệt đối không vượt quá  $10^3$ .

**Yêu cầu:** Hãy tìm số âm và vị trí của nó trong dãy.

**Dữ liệu:**

- + Dòng đầu tiên chứa số nguyên dương  $N$  ( $1 \leq N \leq 10^3$ ).
- +  $N$  dòng tiếp theo, dòng thứ  $i$  chứa số  $a_i$ .

**Kết quả:** Dòng 1 ghi các số âm tìm được, dòng 2 ghi vị trí của nó trong dãy. Trong trường hợp không có lời giải, ghi ra số 0.

Ví dụ:

INPUT	OUTPUT
4	-5 -7
-5	1 4
3	
5	
-7	

### Bài tập 6. Dãy đơn nhất (Vũ Quang, 2019)

Một dãy số được gọi là *dãy số đơn nhất* nếu trong dãy số không có một phần tử nào giống nhau.

Cho một dãy số A gồm N số tự nhiên:  $A_1, A_2, \dots, A_N$  ( $0 < N \leq 10^3, A_i < 10^9$ ).

Hãy kiểm tra dãy số A có phải là *dãy đơn nhất* hay không?

**Dữ liệu vào:** Tệp văn bản DONNHAT.INP theo cấu trúc:

- + Dòng đầu ghi số N.
- + Dòng thứ hai ghi vào N số của dãy A, các số ghi cách nhau một dấu cách.

**Dữ liệu ra:** Ghi vào tệp văn bản DONNHAT.OUT theo cấu trúc:

- + Nếu dãy số A là *dãy đơn nhất* thì ghi 1.
- + Ngược lại ghi 0.

INPUT	OUTPUT
6 20 4 5 6 2 1	1
6 9 4 5 4 6 2	0

### Bài tập 7. Tính tổng (Trà Vinh, 2019)

An là một bạn học sinh rất thích Tin học. Nhân dịp xuân về, lớp tổ chức trò chơi “Ai làm toán nhanh”. Cách chơi như sau: Có n gói kẹo được đánh số từ 1 đến n, gói thứ 1 có ai chiếc kẹo; nhiệm vụ của người chơi là chọn hai gói kẹo trong n gói kẹo đã cho, sao cho tổng của số kẹo trong hai gói kẹo được chọn là k cho trước. Người thắng cuộc sẽ được nhận toàn bộ số kẹo đó.

**Yêu cầu:** Hãy lập trình giúp bạn An là người thắng cuộc trong trò chơi.

**Dữ liệu vào:** Đọc từ tệp văn bản SUM.INP có cấu trúc như sau:

- + Dòng đầu tiên chứa 2 số nguyên dương n và k.
- + Dòng thứ hai chứa n số nguyên  $a_1, a_2, \dots, a_N$ .

**Dữ liệu ra:** Ghi vào tệp văn bản SUM.OUT là vị trí của 2 gói kẹo thỏa mãn yêu cầu trò chơi, nếu không tồn tại cách chọn thì ghi số 0.



Ví dụ:

<b>INPUT</b>	<b>OUTPUT</b>
5 7 2 3 1 5 4	Vi tri cua 2 goi co tong bang 7 1 va 4 2 va 5

### Bài tập 7. Đếm số (Dcows0)

Cho một chuỗi sau “123456789101112”

Chuỗi trên là ghép của các số tự nhiên từ 1 đến N (ở chuỗi trên N=12).

**Yêu cầu:** Hãy đếm số chữ số 0,1,2,..9 của một chuỗi với số N cho trước.

**Input:** “Dcows0.inp”

Số đầu tiên là 1 số nguyên dương NTEST, tiếp theo NTEST dòng mỗi dòng gồm 1 số nguyên dương N ( $10000 \geq N > 0$ ).

**Output:** “Dcows0.out”

Hiện ra theo mẫu sau:

<b>DCOWS0.INP</b>	<b>DCOWS0.OUT</b>
2	N=2
2	(0) 0 (1) 1 (2) 1 (3) 0 (4) 0
12	(5) 0 (6) 0 (7) 0 (8) 0 (9) 0
	N=12
	(0) 1 (1) 5 (2) 2 (3) 1 (4) 1
	(5) 1 (6) 1 (7) 1 (8) 1 (9) 1

### Bài tập 8. Số bé nhất LEXDIV (TP Bắc Giang, 2019)

Khi làm việc với các số tự nhiên, Tom và Jerry sắp xếp chúng theo thứ tự từ điển giống như so sánh xâu ký tự chẵng hạn dãy số (1, 8, 9, 10, 11, 100) sẽ được sắp xếp thành (1, 10, 100, 11, 8, 9). Bài toán Tom lập ra cho Jerry là cho 3 số A, B, K thì số trong đoạn [A; B] chia hết cho K có thứ tự từ điển nhỏ nhất là số nào. Hãy giúp Jerry giải bài toán trên.

**Dữ liệu vào:** Một dòng duy nhất ghi 3 số nguyên A, B, K.

( $1 \leq A \leq B \leq 10^9$ ,  $1 \leq K \leq 10^9$ ). Dữ liệu vào đảm bảo luôn có nghiệm.

**Dữ liệu ra:** Một dòng duy nhất ghi một số nguyên tìm được.

Ví dụ:

<i>Input</i>	<i>Output</i>	<i>Giải thích</i>
96 105 3	102	Đoạn [96; 105] có 4 số chia hết cho 3 là 96, 99, 102, 105. Dãy số trên được sắp xếp theo thứ tự từ điển ở dạng xâu là 102, 105, 96, 99. Đáp án cần tìm là 102.

### Bài tập 9. Tìm số

Viết chương trình đọc n số nguyên, tìm số xuất hiện ít lần nhất, nếu có nhiều số như vậy chọn số bé nhất.

#### Dữ liệu vào:

Từ file “TIMSO.INP” có định dạng như sau:

- + Dòng đầu tiên gồm 1 số nguyên dương NTEST ( $NTEST \leq 100$ ) là số lượng test của bài.
  - + Mỗi test gồm 2 dòng, dòng thứ nhất gồm 1 số nguyên dương n ( $n \leq 100$ ), dòng thứ 2 là n số nguyên.

#### Dữ liệu ra:

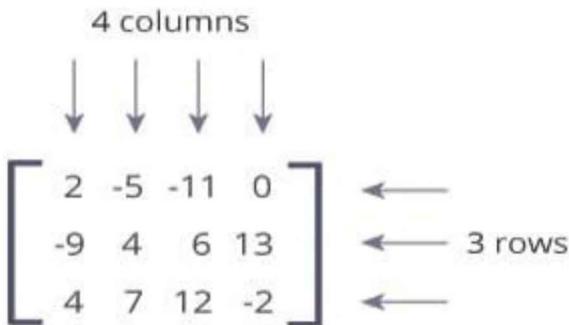
- + Ghi vào file TIMSO.OUT
- + Mỗi test hiện ra số cần tìm và số lần xuất hiện của nó.

<b>TIMSO.INP</b>	<b>TIMSO.OUT</b>
2	1 1
4	2 2
1 2 3 3	
8	
1 1 1 2 2 4 4 4	



## DẠNG BÀI MA TRẬN

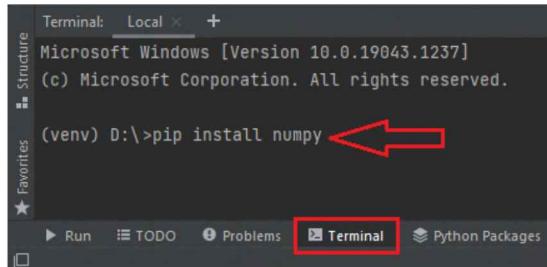
**Ma trận** là cấu trúc dữ liệu hai chiều, trong đó các số được sắp xếp thành các hàng và cột. Ví dụ:



Hình trên là ma trận  $3 \times 4$  vì nó có 3 hàng và 4 cột.

**NumPy** là thư viện được viết bằng Python nhằm phục vụ cho việc tính toán, Numpy hỗ trợ nhiều kiểu dữ liệu đa chiều giúp cho việc tính toán, lập trình, làm việc với các hệ cơ sở dữ liệu cực kì thuận tiện.

Tuy nhiên, để sử dụng được thư viện NumPy, trước tiên ta cần cài đặt thư viện bằng cách mở cửa sổ Terminal trong PyCharm, sau đó gõ `pip install numpy` và đợi thư viện được cài đặt.



## Bài 1. Cách ly (Bình Dương, 2020)

Để ngăn chặn dịch Covid-19 đang diễn ra trên địa bàn, chính quyền cần phải cách ly những đối tượng mang bệnh và có nguy cơ nhiễm bệnh cao tại các khu cách ly tập trung. Các đối tượng có nguy cơ nhiễm bệnh cao được chia ra làm 2 nhóm. Nhóm F1 là các đối tượng tiếp xúc trực tiếp với người mang bệnh, F2 là đối tượng tiếp xúc trực tiếp với người thuộc nhóm F1.

### Dữ liệu đầu vào:

+ Dòng đầu tiên gồm  $N, M$  là số hàng và số cột. ( $1 \leq N, M \leq 100$ ).

+  $N$  dòng tiếp theo mỗi dòng có chính xác  $M$  ký tự biểu thị cho các đối tượng. Ký tự '.' biểu thị cho người có nguy cơ nhiễm bệnh và '\*' biểu thị cho người đang mang bệnh.

### Dữ liệu ra:

Một dòng duy nhất gồm: số lượng các đối tượng thuộc nhóm F1, số lượng các đối tượng thuộc nhóm F2 và số lượng người cần cách li tại khu tập trung. (Cách nhau một khoảng trắng).

Ghi chú: 2 người được gọi là tiếp xúc nếu có cạnh chung trên ma trận.

Dữ liệu vào:	Dữ liệu ra:
2 2 * ..	2 1 4

Giải thích: 2 F1 và 1 F2 và 1 người bệnh.

**Phân tích:** Trường hợp tại vị trí hàng  $i$ , cột  $j$  ( $i, j$ ) là vị trí người nhiễm bệnh, thì khi đó các vị trí xung quanh là các vị trí  $(i, j-1), (i, j+1), (i-1, j), (i+1, j)$ , khi đó tại các vị trí này là vị trí các đối tượng F1. Tương tự như vậy, các vị trí xung quanh các F1 sẽ là F2 tương ứng các vị trí  $(i-2, j), (i-1, j-1), (i-1, j+1), (i, j-2), (i, j+2), (i+1, j-1), (i+1, j+1), (i+2, j)$ . Đề bài yêu cầu ta đếm số lượng F1, F2 và tổng số người cần cách ly (là tổng số F1, F2 và người nhiễm bệnh).

	j-2	j-1	j	j+1	j+2
i-2			F2		
i-1		F2	F1	F2	
i	F2	F1	*	F1	F2
i+1		F2	F1	F2	
i+2			F2		



Để giải bài toán, ta thực hiện tạo một ma trận có kích thước  $n \times m$  và quy ước:

- + Giá trị 0: Người không mang bệnh.
- + Giá trị 1: F2 – Người có nguy cơ mang bệnh.
- + Giá trị 2: F1 – Người có nguy cơ mang bệnh.
- + Giá trị 3: F0 – Người mang bệnh.

Ban đầu các giá trị trong ma trận đều có giá trị 0. Khi đọc dữ liệu tới ký tự “\*” (người mang bệnh), ta tiến hành kiểm tra và chuyển đổi giá trị theo hàng và cột thành các giá trị tương ứng. Sau chuyển đổi xong, trong ma trận, số lượng giá trị 1 là số lượng F2, số lượng giá trị 2 là số F1 và tổng số người cần cách ly là tổng số lượng giá trị 1, 2 và 3.

Trước tiên, chúng ta import thư viện numpy vào dự án và tiến hành mở 2 file dữ liệu vào và dữ liệu ra của bài toán.

```
1 from numpy import *
2 file = open("INPUT.INP")
3 file2 = open("OUTPUT.OUT", "w")
```

Tiếp đó, ta đọc 2 giá trị **n** và **m** trên dòng đầu tiên từ file dữ liệu vào.

```
5 data = file.readline()
6 data = data.split()
7 n = int(data[0])
8 m = int(data[1])
```

Sau khi đọc được **n** và **m**, ta tạo danh sách **a** để lưu dữ liệu đầu vào của bài toán và ma trận **b** có kích thước **n** x **m** và các giá trị trong ma trận đều có giá trị 0.

```
10 a = []
11 b = zeros((n,m))
```

Trong dữ liệu vào có **n** dòng dữ liệu, vậy ta thực hiện lặp lại **n** lần việc đọc dòng dữ liệu và lưu vào **a**.

```
13 for i in range(n):
14     data = file.readline()
15     a.append(data)
```

Sau khi đọc dữ liệu, chúng ta có thể dùng lệnh `print(a)` để quan sát dữ liệu hiện tại đã quét được đúng hay chưa.

Tiếp theo ta tiến hành kiểm tra từng ký hiệu đã quét được trong **a** bằng cách sử dụng hai vòng lặp for lồng nhau. Vòng lặp thứ nhất để duyệt từng hàng và vòng lặp thứ hai để duyệt từng cột trong hàng.

## Chương II. Giải đê theo cách dạng bài

```
17 for i in range(n):  
18     for j in range(m):
```

Với mỗi vị trí (*i*, *j*) ta kiểm tra tại vị trí đó nếu là ký tự "\*" (người nhiễm bệnh), ta đặt **b[i][j]** thành 3 và kiểm tra vị trí các F1 sau đó chuyển giá trị tương ứng thành 2. Tuy nhiên ta chỉ đổi giá trị tại vị trí F1 thành 2 khi giá trị hiện tại trong ma trận nhỏ hơn 2 (tương ứng là người không mang bệnh hoặc đang là F2 sẽ chuyển thành F1, nếu hiện tại người đó là F0 ta sẽ không đổi thành F1 nữa).

```
19     if a[i][j] == '*':  
20         b[i][j] = 3  
21         if i+1 < n:  
22             b[i+1][j] = 2  
23             if j+1 < m:  
24                 b[i][j+1] = 2  
25                 if (j-1 >= 0) and (int(b[i][j-1]) < 2):  
26                     b[i][j-1] = 2  
27                     if i-1 >= 0 and int(b[i-1][j]) < 2:  
28                         b[i-1][j] = 2
```

Tiếp đó ta tiếp hành chuyển đổi các F2 liên quan tới F1. Khi chuyển chúng ta cũng cần kiểm tra vị trí cũng như giá trị hiện tại của phần tử đó trong ma trận.

```
30     if i-2 >= 0 and b[i-2][j] < 1:  
31         b[i-2][j] = 1  
32     if i-1 >= 0:  
33         if j-1 >= 0 and b[i-1][j-1] < 1:  
34             b[i-1][j-1] = 1  
35             if j+1 < m and b[i-1][j+1] < 1:  
36                 b[i-1][j+1] = 1  
37             if j-2 >= 0 and b[i][j-2] < 1:  
38                 b[i][j-2] = 1  
39             if j+2 >= 0 and b[i][j+2] < 1:  
40                 b[i][j+2] = 1  
41  
42             if i+1 < n:  
43                 if j-1 >= 0 and b[i+1][j-1] < 1:  
44                     b[i+1][j-1] = 1  
45                     if j+1 < m and b[i+1][j+1] < 1:  
46                         b[i+1][j+1] = 1  
47             if i+2 < n and b[i+2][j] < 1:  
48                 b[i+2][j] = 1
```

Tiếp đó, ta thực hiện đếm số lượng người nhiễm bệnh (giá trị 3), F1 (giá trị 2) và F2 (giá trị 1) trong ma trận **b**.



```

50 f0 = f1 = f2 = 0
51 for i in range(n):
52     for j in range(m):
53         if int(b[i][j]) == 3:
54             f0 += 1
55         if int(b[i][j]) == 2:
56             f1 += 1
57         if int(b[i][j]) == 1:
58             f2 += 1

```

Cuối cùng, chúng ta lập trình ghi kết quả của bài toán và đóng 2 file dữ liệu.

```

60 file2.write(str(f1) + ' ' + str(f2) + ' ' + str(f1+f2+f0))
61 file.close()
62 file2.close()

```

### Bài 2. Xây dựng bệnh viện dã chiến (Hải Phòng, 2020)

Trước tình hình bùng phát dịch bệnh Covid-19 với tốc độ lây lan rất nhanh, hội đồng thành phố họp bàn cần xây dựng một bệnh viện dã chiến để chữa bệnh cho người dân. Diện tích của thành phố được chia thành  $N \times M$  ( $N, M < 1000$ ) ô vuông đã có đầy đủ các tòa nhà và vẫn còn một số khu vực trống, mỗi tòa nhà nằm trọn trong một số ô vuông. Nếu hội đồng thành phố muốn xây dựng bệnh viện dã chiến thì phải tìm khu vực có diện tích ít nhất là 4 ô vuông thẳng hàng. Hãy viết chương trình có thể đọc bản đồ thành phố từ tệp tin MAP.INP sau đó đưa ra tọa độ đầu và tọa độ cuối của những khu vực có thể xây được bệnh viện vào tệp tin MAP.OUT, ô đất nào xây được bệnh viện đánh dấu "B". Bản đồ thành phố được minh họa như Hình 1.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1				B	B	B	B							
2														
3														
4														
5				B	B	B	B					B		
6												B		
7					B	B	B	B	B			B		
8												B		

Hình 1. Minh họa bản đồ thành phố

### **Định dạng tệp tin MAP.INP**

N: 8

M: 14

+++0000++00++0

00+00+00+++++0

000+++0++++000

++++0+000+++++

+0+0000+0+00+0

+++00+++++00++

0+++00000+00++

00++0++0+++0+0

### **Định dạng tệp MAP.OUT**

So khu vực: 4

(1,4) (1,7)

(5,4) (5,7)

(7,5) (7,9)

(5,12) (8,12)

**Phân tích:** Để giải bài toán, ta tạo một ma trận n x m và quy ước:

- + Giá trị 0: khu vực còn trống.
- + Giá trị 1: khu vực có tòa nhà hoặc đã được xây dựng bệnh viện dã chiến.

Ta khởi tạo mảng với toàn bộ các giá trị là 1. Khi đọc dữ liệu vào, ta xét các vị trí còn trống trong dữ liệu vào (vị trí trống là vị trí ký tự 0) và chuyển đổi từ giá trị 1 thành giá trị 0 tại vị trí tương ứng trong ma trận.

Tiếp đó ta duyệt từng hàng và đếm số lượng bệnh viện dã chiến có thể xây dựng được. Để kiểm tra trong hàng có vị trí nào xây dựng được bệnh viện dã chiến hay không, ta tiến hành duyệt các phần tử trong hàng. Nếu tại vị trí đó là giá trị 0, ta tiến hành đếm số lượng giá trị 0 phía sau. Nếu số giá trị 0 phía sau lớn hơn 4, ta tăng số lượng khu vực thêm 1 và lưu vị trí đầu và cuối của bệnh viện dã chiến, cũng như chuyển đổi các giá trị 0 thành 1 tại khu vực đã xây dựng bệnh viện dã chiến.

Khi đã duyệt hết các hàng, ta tiến hành duyệt các cột, đếm và lưu tọa độ đầu và cuối của các bệnh viện dã chiến có thể xây dựng được.



Trước tiên, ta import thư viện numpy và mở 2 file dữ liệu của bài toán.

```
1 from numpy import *
2 file = open("MAP.INP")
3 file2 = open("MAP.OUT", "w")
```

Tiếp đó, ta khởi tạo danh sách **a** để lưu dữ liệu vào của bài toán, danh sách **hang** và **cot** để lưu chỉ số hàng và chỉ số cột của các vị trí có thể xây dựng bệnh viện dã chiến. Với mỗi khu vực có thể xây dựng, ta cần lưu lại 2 cặp chỉ số hàng và cột tại vị trí bắt đầu và kết thúc. Tạo biến **dem** để lưu số lượng các ô có giá trị 0 liền nhau và biến **khuvuc** để lưu số lượng khu vực ta có thể xây dựng bệnh viện dã chiến.

```
5 a = []
6 hang = []
7 cot = []
8 dem = 0
9 khuvuc = 0
```

Sau khi khởi tạo danh sách và các biến, ta đọc dữ liệu từ file dữ liệu vào và lưu vào biến cũng như danh sách tương ứng.

```
11 n = int(file.readline())
12 m = int(file.readline())
13 for i in range(n):
14     a.append(file.readline())
```

Để khởi tạo ma trận ban đầu gồm tất cả các giá trị đều bằng 1, ta sử dụng phương thức `ones((n, m))` với **n** là số hàng và **m** là số cột của ma trận.

```
16 b = ones((n, m))
```

Sau khi tạo được ma trận **b** gồm các giá trị 1, ta tiến hành kiểm tra các vị trí là ký tự “0” trong dữ liệu vào và đổi giá trị tại vị trí đó trong ma trận **b** thành 1.

```
17 for i in range(n):
18     for j in range(m):
19         if a[i][j] == '0':
20             b[i][j] = 0
```

Để tiến hành duyệt các vị trí có thể xây dựng bệnh viện dã chiến theo chiều ngang, ta xét từng hàng trong ma trận **b**. Ta xét từ vị trí đầu tới vị trí cuối trong hàng, vậy ta tạo biến **j** và đặt giá trị ban đầu cho biến bằng 0 và tăng dần tới vị trí cuối cùng trong hàng. Với mỗi vị trí, ta kiểm tra nếu giá trị tại vị trí hiện tại là 0, ta tiến hành đếm các giá trị 0 phía sau vị trí đó.

```

22   for i in range(n):
23     j = 0
24     while j < m:
25       if int(b[i][j]) == 0:
26         temp = j
27         dem = 1
28         j += 1
29         while ((j < m) and (int(b[i][j]) == 0)):
30           dem += 1
31           j += 1

```

Khi đã đếm được các số 0 liền sau, ta kiểm tra giá trị biến **dem**, nếu **dem** lớn hơn 3 thì khu vực này ta có thể xây dựng bệnh viện dã chiến. Khi đó ta tăng số khu vực thêm 1, đặt lại biến **dem** về 0 trước khi duyệt tiếp các vị trí tiếp theo, thêm vị trí hàng hiện bắt đầu, cột bắt đầu, hàng kết thúc và cột kết thúc của khu vực có thể xây dựng bệnh viện dã chiến vào hai danh sách **hang** và **cot**. Sau đó ta thay đổi các giá trị 0 trong khu vực thành 1 để đánh dấu tại các vị trí này đã xây dựng bệnh viện dã chiến.

```

32   if dem > 3:
33     khuvuc += 1
34     dem = 0
35     hang.append(i+1)
36     hang.append(i+1)
37     cot.append(temp+1)
38     cot.append(j)
39     for k in range(temp, j):
40       b[i][k] = 1
41     j += 1

```

Sau khi hoàn thiện duyệt từng hàng, ta tiến hành duyệt từng cột theo chiều dọc và thay đổi các giá trị nếu khu vực có thể xây dựng bệnh viện dã chiến tương tự như trên.

```

42 for i in range(m):
43   j = 0
44   while j < n:
45
46     if int(b[j][i]) == 0:
47       temp = j
48       dem = 1
49       j += 1
50       while ((j < n) and (int(b[j][i]) == 0)):
51         dem += 1
52         j += 1
53     if dem > 3:
54       khuvuc += 1
55       dem = 0
56       hang.append(temp+1)
57       hang.append(j)
58       cot.append(i+1)

```



```

59             cot.append(i+1)
60             for k in range(temp, j):
61                 b[k][i] = 1
62             j += 1

```

Khi đã duyệt được tất cả các vị trí có thể xây dựng bệnh viện dã chiến cả theo chiều ngang và chiều dọc. Ta tiến hành xuất kết quả ra tệp tin dữ liệu ra của bài toán. Dòng đầu tiên của dữ liệu ra là số lượng khu vực ta có thể xây dựng bệnh viện dã chiến. Vì dữ liệu ra có nhiều dòng, để xuống dòng ta ghi vào file ký tự '\n'.

```
64 file2.write('Số khu vực: ' + str(khuvuc) + '\n')
```

Các dòng tiếp theo cần ghi vào file dữ liệu ra là vị trí đầu và vị trí cuối của các khu vực có thể xây dựng bệnh viện dã chiến. Các vị trí này ta đã lưu vào danh sách **hang** và **cot**.

```

65 for i in range(0, khuvuc*2, 2):
66     s = '(' + str(hang[i]) + ',' + str(cot[i]) + ')' + (
67         + str(hang[i+1]) + ',' + str(cot[i+1]) + ')'
68     file2.write(s + '\n')

```

**Lưu ý:** Giữa dòng 66 và 67 không được đánh số. Ta ngầm hiểu dòng đó sẽ được viết liền với dòng trên (dòng số 66) trong giao diện lập trình.

Cuối cùng, sau khi hoàn thiện ghi dữ liệu ra của bài toán, ta đóng 2 file dữ liệu của bài toán.

```

68 file.close()
69 file2.close()

```

### Bài 3. Tứ giác đồng hồ cạnh K (Đông Triều, 2019)

Cho mảng vuông  $A[i, j]$  ( $i, j = 1, 2, \dots, n$ ), chỉ số hàng của  $A$  được tính từ trên xuống dưới, chỉ số cột được tính từ trái sang phải, các phần tử của nó là các số nguyên. Bốn ô  $A[i, j]; A[i, j+k]; A[i+k, j+k]; A[i+k, j]$  thuộc mảng được gọi là 4 đỉnh của một “**Tứ giác đồng hồ cạnh k**” nếu các số ở đỉnh của nó xếp theo thứ tự tăng dần theo chiều kim đồng hồ từ một đỉnh nào đó.

**Yêu cầu:** Tính số lượng các Tứ giác đồng hồ cạnh k của bảng A.

**Dữ liệu vào:**

Dòng 1: Hai số  $n, k$  ( $2 \leq n \leq 100; 1 \leq k \leq n-1$ ).

$n$  dòng tiếp theo mỗi dòng có  $n$  số nguyên mỗi số có trị tuyệt đối không vượt quá  $10^6$  là giá trị của mảng  $A[i, j]$ . Hai số liên tiếp cách nhau ít nhất một ký tự trắng.

**Dữ liệu ra:**

Số S là số các tứ giác đồng hồ cạnh k.

<b>INPUT</b>	<b>OUTPUT</b>
4 3	1
4 3 2 5	
4 4 5 5	
2 3 5 3	
3 5 6 7	

**Phân tích:** Để giải bài toán, ta tiến hành xét từng vị trí có thể là đỉnh trên bên trái của tứ giác cạnh k. Với mỗi vị trí, ta kiểm tra lần lượt 4 đỉnh có được sắp xếp theo thứ tự tăng dần theo chiều kim đồng hồ hay không. Nếu có, ta tăng số lượng tứ giác thỏa mãn yêu cầu là tứ giác đồng hồ cạnh k thêm 1 và khi đã duyệt tất cả các vị trí có thể là đỉnh trên bên trái của tứ giác. Ta xuất kết quả chính là số lượng tứ giác thỏa mãn yêu cầu vào tệp tin OUTPUT.

Trước tiên, ta import thư viện numpy và mở 2 file dữ liệu của bài toán.

```
1 from numpy import *
2 file = open("INPUT.INP")
3 file2 = open("OUTPUT.OUT", "w")
```

Tiếp đó, ta quét dữ liệu dòng đầu tiên, tách dữ liệu thành danh sách và gán vào biến **n** và **k** tương ứng giá trị **n** và **k** của bài toán.

```
5 temp = file.readline()
6 temp = temp.split()
7 n = int(temp[0])
8 k = int(temp[1])
```

Ta khởi tạo biến **dem** lưu số lượng tứ giác đồng hồ cạnh **k** thỏa mãn yêu cầu đề bài, tạo ma trận **a** ban đầu gồm các phần tử 0.

```
10 dem = 0
11 a = zeros((n, n))
```

Sau khi khởi tạo ma trận **a**, ta tiến hành đọc **n** dòng dữ liệu và lưu vào ma trận **a**.

```
13 for i in range (n):
14     temp = file.readline()
15     temp = temp.split()
16     a[i] = temp
```

Ta tạo hàm **check(i, j)** với **i** và **j** tương ứng vị trí hàng và cột tại đỉnh trên bên trái của tứ giác cạnh **k** cần kiểm tra. Ta tạo 4 biến **num1**, **num2**, **num3**, **num4** lưu giá trị tại 4 đỉnh của tứ giác cạnh **k**.



```

18 def check(i, j):
19     num1 = int(a[i][j])
20     num2 = int(a[i][j+k-1])
21     num3 = int(a[i+k-1][j+k-1])
22     num4 = int(a[i+k-1][j])

```

Vì tứ giác đồng hồ cạnh **k** có thể tăng dần từ bất kỳ vị trí góc nào. Chính vì vậy ta so sánh và kiểm tra lần lượt thứ tự tại 4 đỉnh. Nếu tứ giác thỏa mãn điều kiện tăng dần ta lập trình hàm trả về kết quả True, còn không thì trả về kết quả False.

```

24     if num1 < num2 and num2 < num3 and num3 < num4:
25         return True
26     elif num2 < num3 and num3 < num4 and num4 < num1:
27         return True
28     elif num3 < num4 and num4 < num1 and num1 < num2:
29         return True
30     elif num4 < num1 and num1 < num2 and num2 < num3:
31         return True
32     else:
33         return False

```

Trong đoạn chương trình chính, ta duyệt từng vị trí có thể là đỉnh trên bên trái của tứ giác cạnh **k**, và gọi hàm kiểm tra tứ giác đó có thỏa mãn là tứ giác đồng hồ cạnh **k** hay không bằng cách gọi hàm hàm **check (i, j)**. Nếu tứ giác đó thỏa mãn, ta tăng biến **dem** thêm 1.

```

35 i = 0
36 while i+k-1 < n:
37     j = 0
38     while j+k-1 < n:
39         if check(i, j):
40             dem += 1
41         j += 1
42     i += 1

```

Cuối cùng ta ghi kết quả vào file dữ liệu ra của bài toán và đóng 2 file dữ liệu.

```

44 file2.write(str(dem))
45 file.close()
46 file2.close()

```

#### Bài 4. Ma trận hợp lệ

Cho 1 ma trận  $A (n \times n)$ , ma trận  $a$  được gọi là “**hợp lệ**” nếu tất cả các phần tử trên đường chéo chính đều bằng 0; các phần tử phía trên đường chéo chính đều dương và các phần tử còn lại đều âm. Hãy kiểm tra xem ma trận vừa nhập có hợp lệ không? Nếu hợp lệ ghi một số 1 và 0 nếu ngược lại.

Đường chéo chính xuất phát từ tọa độ  $(0, 0)$  đến  $(n, n)$ .

Bài tập có Ntest.

**Dữ liệu vào:** Từ file “**MatranHL.inp**” có format như sau:

- + Dòng đầu tiên gồm 1 số nguyên dương  $NTEST (NTEST \leq 100)$  là số lượng test của bài.
- + Mỗi test là một ma trận, bắt đầu là một dòng chứa số  $n$  là kích cỡ, tiếp theo là  $n \times n$  số nguyên.

Ví dụ:

<b>MatranHL.inp</b>	<b>MatranHL.out</b>
2	0
2	1
1 -1	
3 3	
3	
0 3 1	
-1 0 1	
-1 -7 0	

Trước tiên, ta thực hiện mở 2 file dữ liệu vào và dữ liệu ra của bài toán.

```
1  file = open("MatranHL.inp")
2  file2 = open("MatranHL.out", "w")
```

Tiếp theo, ta thực hiện đọc dữ liệu dòng đầu tiên trong file dữ liệu vào chính là số test của bài toán sau đó lưu vào biến **ntest** tương ứng.

```
4  ntest = int(file.readline())
```

Ta sử dụng vòng lặp for để chạy số **test** từ 0 tới **ntest**. Với mỗi bộ **test**, trước tiên ta đọc giá trị **n** chính là kích thước của ma trận.

```
5  for test in range(ntest):
6      n = int(file.readline())
```



Ta tạo biến **kq** lưu kết quả của bài toán, ban đầu ta đặt **kq** bằng 1 sau đó đọc các dòng dữ liệu tương ứng các hàng trong ma trận. Với hàng thứ **i**, ta thực hiện kiểm tra giá trị thứ **0** trong hàng nếu khác “0” ta chuyển giá trị biến **kq** thành 0 – Ma trận không hợp lệ.

```
7     kq = 1
8     for i in range(n):
9         data = file.readline()
10        data = data.split()
11        if data[i] != '0':
12            kq = 0
```

Sau khi đã duyệt và đọc hết dữ liệu trong ma trận, ta ghi kết quả vào file dữ liệu ra của bài toán. Tuy nhiên, ta cần lưu ý ghi ký tự ngắt dòng “\n” vào file nếu **test** hiện tại không phải **test** đầu tiên trong bài toán.

```
13     print(kq)
14     if test > 0:
15         file2.write("\n")
16         file2.write(str(kq))
```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán sau khi đã ghi xong kết quả.

```
18 file.close()
19 file2.close()
```

## Bài tập thực hành

### Bài tập 1. Tổng ma trận

Cho 1 ma trận hình vuông, hãy tính tổng các phần tử của nó.

Bài tập có N test.

**Dữ liệu vào:**

Từ file “TONGMATRAN.INP” có định dạng như sau:

- + Dòng đầu tiên gồm 1 số nguyên dương NTEST ( $NTEST \leq 100$ ) là số lượng test của bài.
- + Mỗi test là một ma trận, bắt đầu là một dòng chứa số n là kích cỡ, tiếp theo là  $n \times n$  số nguyên.

**Dữ liệu ra:**

Ghi vào file “TONGMATRAN.OUT”.

Mỗi dòng là tổng của từng ma trận.

Ví dụ:

TONGMATRAN.INP	TONGMATRAN.OUT
2	8
2	9
1 1	
3 3	
3	
1 1 1	
1 1 1	
1 1 1	

### Bài tập 2. Số ô liên tục trên bàn cờ (Đông Triều, 2019)

Cho một bàn cờ vuông  $N \times N$ , trên đó cho trước một số quân cờ. Ví dụ hình vẽ bên là một bàn cờ ( $8 \times 8$ ) như vậy:

**Dữ liệu vào:** BANCO.INP

- + Dòng 1: Số N ( $N \leq 20$ ) kích thước của bàn cờ.
- + N dòng tiếp theo, mỗi dòng gồm N số 0 hoặc 1, vị trí các quân cờ ứng với số 1, các ô trống ứng với số 0.



Hãy viết chương trình tính số quân cờ liên tục lớn nhất nằm trên một đường thẳng trên bàn cờ. Đường thẳng ở đây có thể là đường thẳng đứng, đường nằm ngang hoặc đường chéo.

	x		x		x		
x			x				x
x		x				x	x
			x		x		
		x					
	x		x				x
x			x	x			
	x				x	x	

Dữ liệu ra: BANCO.OUT

Một số duy nhất là số quân cờ nhiều nhất nằm trên một đường thẳng.

Ví dụ:

BANCO.INP	BANCO.OUT
8 0 1 0 1 0 1 0 0 1 0 0 1 1 0 0 1 1 0 1 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 0 1 1 0	4

### Bài tập 3. Hình vuông đồng nhất (Gia Lai, 2019)

Cho một lưới ô vuông kích thước  $M \times N$ . Ô nằm trên giao của dòng  $i$  và cột  $j$  của lưới sẽ được gọi là ô  $(i, j)$  của lưới người ta viết số nguyên không âm  $a_{ij}$ . Ta gọi hình vuông đồng nhất bậc 2 của lưới là tập gồm 4 ô nằm trên giao của hai dòng liên tiếp và 2 cột liên tiếp của lưới với các số viết trên chúng là như nhau.

**Yêu cầu:** Tính số lớn nhất các hình vuông đồng nhất bậc 2 chứa cùng một số.

**Dữ liệu vào:** Được đặt trong file văn bản HINHVUONG.INP:

+ Dòng đầu tiên chứa các số nguyên dương  $M, N$  ( $M, N \leq 1000$ );

+ Dòng thứ  $i$  trong số  $M$  dòng tiếp theo chứa các số  $a_{i1}, a_{i2}, \dots, a_{iN}$ ,  $i = 1, 2, \dots, M$ , hai số liên tiếp trên dòng được viết cách nhau một dấu cách.  $0 \leq a_{ij} \leq 255$ .

**Kết quả ra:** đặt trong file văn bản HINHVUONG.OUT: số lớn nhất các hình vuông đồng nhất bậc 2 chứa cùng một số.

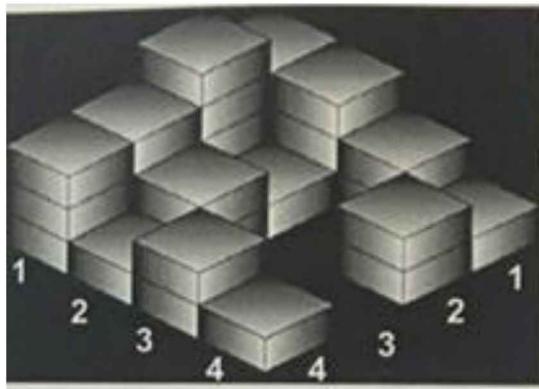
Ví dụ:

HINHVUONG.INP	HINHVUONG.OUT
5 10	3
0 1 1 0 2 2 0 5 5 0	
0 1 1 0 2 2 0 5 5 0	
0 0 0 1 1 0 0 0 0 0	
0 8 0 1 1 1 1 0 9 0	
0 0 0 0 0 1 1 0 0 0	

#### Bài tập 4. Xếp sách (Mê Linh, 2019)

Thư viện trường vừa được bổ sung một khối lượng lớn sách. Người thủ thư phân loại sách và xếp chúng thành từng chồng trên một bàn cao. Mặt bàn được chia thành lưới  $n \times n$  ô ( $1 \leq n \leq 1000$ ). Mỗi chồng sách chiếm vừa khít một ô vuông. Có thể có các ô trống trên bàn. Các cuốn sách được đánh dấu ký hiệu phân loại ở gáy sách và các phía xung quanh. Nam được giao nhiệm vụ ghi các sách mới vào phiếu tra cứu của thư viện. Nam đi 1 vòng quanh bàn, nhìn đồng sách theo các hướng song song với cạnh bàn, đọc các phiếu phân loại của từng chồng sách nhìn thấy được.

Một chồng sách có thể được nhìn thấy nếu giữa Nam và chồng sách không có chồng nào cao hơn hoặc bằng theo hướng nhìn. Theo tình huống ở hình bên, người Nam không thấy được chồng sách ở ô (2, 2). Các ô (2, 3), (3, 3) và (3, 4) – ô trống.



**Yêu cầu:** Hãy xác định số chòng sách mà Nam nhìn thấy được.

**Dữ liệu:** Vào từ file văn bản BOOK.INP:

+ Dòng đầu tiên chứa số nguyên  $n$ .

+ Dòng thứ  $i$  trong  $n$  dòng sau chứa  $n$  số nguyên xác định độ cao các chòng sách trong hàng, mỗi độ cao có giá trị không vượt quá 1000.

**Kết quả:** Đưa ra file văn bản BOOK.OUT một số nguyên – số chòng sách nhìn thấy được.

Ví dụ:

INPUT	OUTPUT
4	12
3 3 2 1	
4 1 0 2	
3 2 0 0	
3 1 2 1	

### Bài tập 5. Di chuyển cây (Lâm Đồng, 2019)

Bờm có một vườn cây được mô tả dưới dạng hình chữ nhật gồm  $m$  dòng và  $n$  cột. Trong vườn có  $k$  loại cây khác nhau và được đánh số từ 1 đến  $k$  ( $0 < k < 10$ ). Mỗi ô của khu vườn có chứa duy nhất một số nguyên dương  $i$  ( $i \leq k$ ) nếu tại ô này có trồng một cây loại  $i$  hoặc số 0 nếu tại ô này không có cây. Bờm muốn chỉnh trang khu vườn cho đẹp hơn bằng cách giữ lại những hàng cây có ít nhất  $t$  cây liền nhau, thuộc cùng một loại cây, nằm trên cùng một dòng hoặc cùng một cột. Những cây không thuộc những hàng cây được giữ lại sẽ bị di chuyển đến vị trí khác phù hợp hơn.

**Yêu cầu:** Cho trước vườn cây được mô tả dưới dạng hình chữ nhật gồm  $m$  dòng và  $n$  cột. Hãy đếm số lượng cây cần di chuyển.

**Dữ liệu vào:**

Từ file DICHUYEN.INP

+ Dòng đầu chứa ba số nguyên dương  $m$ ,  $n$  và  $t$  ( $1 < m, n, t, \leq 100$ ), mỗi số cách nhau một dấu cách.

+ Trong  $m$  dòng tiếp theo, mỗi dòng chứa  $n$  số nguyên dương mô tả vườn cây (mỗi số cách nhau một dấu cách).

**Kết quả:**

Ghi ra file DICHUYEN.OUT

+ Ghi số lượng cây cần phải di chuyển.

Ví dụ:

DICHUYEN.INP	DICHUYEN.OUT	Giải thích
5 6 3	10	Những số gạch chân dưới đây biểu thị những cây cần phải di chuyển:
1 3 3 3 3 4		<u>1</u> 3 3 3 3 4
1 2 3 2 0 4		<u>1</u> 2 <u>3</u> 2 0 4
3 2 2 2 4 4		<u>3</u> 2 2 2 4 4
1 0 0 2 4 0		<u>1</u> 0 0 2 4 0
1 2 3 0 4 4		<u>1</u> 2 <u>3</u> 0 4 4

### Bài tập 6. Ma trận thưa (TP. Vinh, 2019)

Một ma trận gọi là *thưa* nếu số phần tử 0 chiếm quá một nửa tổng số phần tử. Em hãy nhập vào 1 ma trận và kiểm tra tính chất *thưa* của ma trận đó.

**Dữ liệu nhập:** Cho trong file KTR.INP gồm:

- + Dòng 1: Số nguyên  $n$  là số dòng và  $m$  là số cột của ma trận.
- + Các dòng còn lại là giá trị từng phần tử của ma trận.

**Dữ liệu xuất:** Xuất ra file KTR.OUT gồm 1 dòng duy nhất trả lời đúng hay sai.

Ví dụ:

KTR.INP	KTR.OUT	KTR.INP	KTR.OUT
3 3	Sai, khong la ma tran thua	3 3	Dung la ma tran thua
1 0 0		1 0 0	
0 4 1		0 0 1	
6 8 0		0 8 0	

### Bài tập 7. Bảng Ma phương (An Giang, 2020)

Bảng Ma phương là bảng vuông  $N \times N$  ( $N$  lẻ) có tổng các hàng ngang, hàng dọc và 2 đường chéo đều bằng nhau với giá trị các ô là các số tự nhiên từ 1 đến  $N^2$  (không có giá trị trùng nhau).

Ví dụ: đây là bảng Ma phương cấp 3:

2	7	6
9	5	1
4	3	8



Viết chương trình tạo bảng Ma phương theo yêu cầu sau:

**Input:**

Cho trong tập tin MAPHUONG.INP chứa số tự nhiên N (N lẻ).

**Output:**

Ghi vào tập tin MAPHUONG.OUT chứa bảng Ma phương gồm N dòng, mỗi dòng gồm N giá trị, mỗi giá trị cách nhau ít nhất một ký tự trắng.

Ví dụ:

MAPHUONG.INP	MAPHUONG.OUT
3	2 7 6 9 5 1 4 3 8

### Bài tập 8. Tìm dãy số (Hải Phòng, 2020)

Cho một tập bộ ba các số nguyên dương lưu trữ trong một tập tin INPUT.INP. Quy định mỗi bộ số đặt trên một dòng, mỗi số trong một bộ số cách nhau bằng một dấu cách và được minh họa như sau:

7 8 2  
2 5 6  
1 7 4  
4 5 1  
2 1 1 1 1  
1 4 3 4 2 4  
1 1 6 8  
1 0 7 1 1  
8 1 1 1

Hãy tìm các dãy có độ dài từ hai bộ số trở lên. Trong đó, số cuối cùng của bộ số đứng trước trùng với số đầu tiên của bộ số tiếp theo. Kết quả được ghi ra tập tin OUTPUT.OUT với mỗi dãy tìm được trên một dòng. Tập kết quả có định dạng như sau:

So dãy: 2

7 8 2, 2 5 6  
1 7 4, 4 5 1

### Bài tập 9. Đếm số nguyên tố

Cho 1 ma trận  $m \times n$  ( $m$  hàng,  $n$  cột), hãy tìm số nguyên tố lớn nhất và đếm số lần xuất hiện của số nguyên tố lớn nhất đó trong ma trận.

Số nguyên tố là số nguyên dương lớn hơn 1 chỉ chia hết 1 và chính nó.

Nếu không có số nguyên tố nào, ghi ra một số không “0”.

**Dữ liệu vào:** Từ file “DEMNTO.INP” có định dạng như sau:

+ Dòng đầu tiên gồm 1 số nguyên dương NTEST ( $NTEST \leq 100$ ) là số lượng test của bài.

+ Mỗi test là một ma trận, bắt đầu là một dòng chứa số nguyên dương  $m$  và  $n$  ( $0 < m, n \leq 100$ ) là kích cỡ, tiếp theo là  $m \times n$  số nguyên.

DEMNTO.INP	DEMNTO.OUT
2	3 2
3 2	7 1
1 -1	
3 3	
1 1	
3 3	
1 3 1	
1 5 1	
1 7 1	

### Bài tập 10. Square

Cho một ma trận nhị phân (các phần tử trong ma trận có giá trị 0 hoặc 1).

Tìm số lượng các hình vuông thuộc ma trận trên sao cho các phần tử trong hình vuông đều có giá trị khác 0.

**Dữ liệu đầu vào:** Từ file văn bản “SQUARE.INP” có định dạng như sau:

Dòng đầu tiên ghi một số NTEST là số lượng ma trận ( $0 < NTEST \leq 1000$ ).

Tiếp theo là NTEST bộ dữ liệu trong đó:

+ Dòng đầu gồm 3 số: số hàng, số cột và kích cỡ của hình vuông cần tìm (giá trị không quá 100).

+ Các giá trị của ma trận tương ứng.

**Đầu ra:** Ghi vào file “SQUARE.OUT” gồm NTEST dòng, mỗi dòng hiển thị số lượng hình vuông thỏa mãn yêu cầu ứng với từng ma trận và kích cỡ hình vuông đã cho.



SQUARE.INP	SQUARE.OUT
2	3
2 2 1	4
0 1	
1 1	
4 5 3	
1 1 1 1 0	
1 1 1 1 0	
1 1 1 1 0	
1 1 1 1 0	

### Bài tập 11. Ma trận đối gương

Cho 1 ma trận A ( $n \times n$ ), ma trận a được gọi là “đối gương” nếu tất cả các phần tử bằng các phần tử đối xứng với nó qua trực dọc trung tâm. Hãy kiểm tra xem ma trận có đối gương hay không? Nếu đối gương ghi một số 1 và 0 nếu ngược lại.

**Dữ liệu vào:** Từ file “MTDOIGUONG.INP” có format như sau:

- + Dòng đầu tiên gồm 1 số nguyên dương NTEST là số lượng test của bài.
- + Mỗi test là một ma trận, bắt đầu là một dòng chứa số nguyên dương n là kích cỡ ( $1 < n \leq 100$ ), tiếp theo là  $n \times n$  số nguyên.

MTDOIGUONG.INP	MTDOIGUONG.OUT
3	1
2	1
1 1	0
3 3	
3	
1 3 1	
1 0 1	
0 1 0	
3	
0 3 1	
3 0 1	
1 2 0	

## DẠNG BÀI SẮP XẾP

Sắp xếp là quá trình bố trí lại các phần tử của một tập đối tượng nào đó theo một thứ tự nhất định. Chẳng hạn như thứ tự tăng dần (hay giảm dần) đối với một dãy số, thứ tự từ điển đối với các từ,... Yêu cầu sắp xếp thường xuyên xuất hiện trong các ứng dụng Tin học với các mục đích khác nhau: Sắp xếp dữ liệu trong máy tính để tìm kiếm cho thuận lợi, sắp xếp các kết quả xử lý để in ra trên bảng biểu, sắp xếp danh sách học sinh....

Trong các ngôn ngữ lập trình bậc cao khác như C/C++, Java,... để sắp xếp ta cần sử dụng các kỹ thuật khác nhau tùy thuộc vào độ phức tạp của bài toán. Các kỹ thuật này được gọi chung là thuật toán sắp xếp. Một số thuật toán sắp xếp cơ bản như sắp xếp kiểu chọn (Selection Sort), sắp xếp nổi bọt (Bubble Sort), ...

Với Python, ta có thể thực hiện việc sắp xếp đơn giản hơn nhờ phương thức sort() hoặc sorted(). Các bạn cũng có thể tìm hiểu thêm về một số thuật toán sắp xếp bằng cách tìm kiếm với một số từ khóa: thuật toán sắp xếp (Sort algorithm), sắp xếp kiểu chọn (Selection Sort), sắp xếp nổi bọt (Bubble Sort), Quick Sort, ...

### Lưu ý:

- + Hàm sort() và sorted() là một phương thức của danh sách (list) trong python.
- + Hàm sort() không trả về một list mới được sắp xếp mà thay đổi ngay trên danh sách đó.
- + Hàm sorted() không thay đổi danh sách ban đầu mà tạo ra một danh sách mới được sắp xếp lại từ danh sách ban đầu.

### Cú pháp:

```
list.sort(reverse=True|False, key=myFunc)
```

```
List.sorted(reverse=True|False, key=myFunc)
```

### Trong đó:

- list: danh sách cần sắp xếp
- reverse: reverse = True – sắp xếp theo chiều giảm dần. Mặc định reverse là False – sắp xếp theo chiều tăng dần.
- key: một hàm để chỉ định tiêu chí khi sắp xếp. Ví dụ sắp xếp theo độ dài chuỗi tăng dần: `list.sort(key=len)`



### Bài 1. Dãy số (Bắc Giang, 2019)

Sử dụng hàm Randomize để khởi tạo dãy số ngẫu nhiên từ 0 đến 9 gồm N phần tử ( $0 < N \leq 100$ ), kết quả ghi ra tệp RANDOM.OUT, mỗi phần tử cách nhau 01 dấu cách.

Viết chương trình đọc dữ liệu từ tệp RANDOM.INP, sau đó sắp xếp lại các phần tử theo chiều tăng dần, đồng thời cho biết số lần xuất hiện của mỗi phần tử trong dãy số đã được khởi tạo.

**Kết quả:**

Ghi ra tệp RANDOM.OUT gồm 11 dòng:

- + Dòng thứ nhất là dãy các phần tử đã được sắp xếp.
- + Dòng thứ 2 đến dòng thứ 11 tương ứng chữ số ghi tổng số lần xuất hiện của 0, 1, ..., 9.

Ví dụ:

INPUT	OUTPUT
0 5 2 0 1 6 7 8 7 3 1	0 0 1 1 2 3 5 6 7 7 8
	2
	2
	1
	1
	0
	1
	1
	2
	1
	0

Để bài yêu cầu khởi tạo một dãy số ngẫu nhiên gồm N chữ số. Vậy trước tiên ta cần import thư viện random vào dự án và mở 2 file dữ liệu của bài toán. Tuy nhiên với file RANDOM.INP chúng ta cần mở và ghi dãy số được lấy ngẫu nhiên. Vậy file input chúng ta cần mở để cả ghi và đọc.

```
1 from random import *
2 file = open('RANDOM.INP', 'w+')
3 file2 = open('RANDOM.OUT', 'w')
```

Tiếp đó, ta yêu cầu nhập vào giá trị N và lập trình tạo dãy gồm n số ngẫu nhiên có giá trị từ 0 tới 9. Để lấy giá trị ngẫu nhiên, ta sử dụng phương thức ranint(a, b) với a và b là giới hạn của giá trị cần lấy ngẫu nhiên.

```

5 n = int(input('Nhập giá trị N: '))
6 s = ''
7 for i in range(n):
8     temp = randint(0, 9)
9     s += str(temp) + ' '

```



### Bạn có biết: Một số hàm trong thư viện random

1. `randrange(a)`: tạo ra một số nguyên ngẫu nhiên nhỏ hơn a.
2. `randrange(a, b, step)`: tạo một số ngẫu nhiên trong khoảng từ a tới b - 1 với bước nhảy là step.
3. `randint(a, b)`: tạo một số nguyên ngẫu nhiên trong khoảng từ a tới b bao gồm cả a và b.
4. `uniform(a, b)`: tạo một số thực ngẫu nhiên trong khoảng từ a tới b.

Sau khi tạo được dãy số ngẫu nhiên, ta ghi dãy số vào file input. Sau khi ghi dữ liệu con trỏ sẽ nằm tại vị trí cuối dãy số. Chính vì vậy khi muốn đọc dữ liệu, ta cần chuyển con trỏ về vị trí đầu file bằng phương thức `seek()`.

```

11 file.write(s)
12 file.seek(0)

```

Tiếp đó ta khởi tạo danh sách `c` để lưu số lượng các số từ 0 đến 9 trong dãy số. Ta quy ước như sau:

Vị trí	0	1	2	3	...
	Số lượng số 0	Số lượng số 1	Số lượng số 2	Số lượng số 3	...

Ban đầu ta khởi tạo các giá trị trong `c` đều là 0, sau đó đọc dữ liệu từ file và tách dãy số bằng phương thức `split()`.

```

14 c = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
15 data = file.readline()
16 data = data.split()

```



Tiếp đó, ta sắp xếp lại dãy số theo yêu cầu của đề bài. Để sắp xếp, trong bài toán này ta có thể sử dụng phương thức sort(). Sau khi sắp xếp, ta có thể in dãy số sau khi sắp xếp lên cửa sổ Run để kiểm tra kết quả.

```
17 data.sort()
18 print(data)
```

Sau khi sắp xếp, ta lập trình đếm số lượng các số trong dãy số.

```
20 for x in list(data):
21     i = int(x)
22     c[i] += 1
```

Cuối cùng, ta ghi dữ liệu ra của bài toán là dãy số sau khi sắp xếp và số lượng các số trong dãy số đó. Đồng thời đóng 2 file dữ liệu của bài toán sau khi ghi xong dữ liệu ra.

```
24 file2.write(' '.join(data))
25 for x in list(c):
26     file2.write(('\n' + str(x)))
27
28 file.close()
29 file2.close()
```

## Bài 2. Dãy không giảm (Vị Thanh – Hậu Giang, 2019)

Nhập từ bàn phím 3 số nguyên dương  $a_1, a_2, a_3$  ( $100 < a_1, a_2, a_3 < 10^5$ ).

Dãy số  $b$  được sinh ra bằng cách ghép từng số nguyên dương đã nhập lần lượt với 2 số còn lại, ví dụ  $a_1 = 234, a_2 = 123, a_3 = 345$  ta tìm được  $b_1 = 234123, b_2 = 234345, b_3 = 123234, b_4 = 123345, b_5 = 345234$ .

Sắp xếp các số trong dãy số  $b$  thành dãy không giảm và xuất ra màn hình, các số cách nhau một khoảng trắng.

Ví dụ:

INPUT	OUTPUT
Nhập $a_1: 234$ Nhập $a_2: 123$ Nhập $a_3: 345$	123234 123345 234123 234345 345123 345234

Để giải bài toán, trước tiên ta yêu cầu nhập và 3 giá trị  $a_1, a_2, a_3$  từ bàn phím và lưu vào 3 biến tương ứng.

```
1 a1 = input('Nhập a1: ')
2 a2 = input('Nhập a2: ')
3 a3 = input('Nhập a3: ')
```

Tiếp đó ta tạo danh sách **b** để lưu các số được nối bởi 2 trong 3 giá trị **a1**, **a2**, **a3** đã nhập. Sau đó ta lập trình nối chuỗi và thêm vào danh sách **b**.

```
5 b = []
6 temp = str(a1) + str(a2)
7 b.append(int(temp))
8 temp = str(a1) + str(a3)
9 b.append(int(temp))
10 temp = str(a2) + str(a1)
11 b.append(int(temp))
12 temp = str(a2) + str(a3)
13 b.append(int(temp))
14 temp = str(a3) + str(a1)
15 b.append(int(temp))
16 temp = str(a3) + str(a2)
17 b.append(int(temp))
```

Cuối cùng, ta sắp xếp lại danh sách **b** và xuất kết quả lên màn hình.

```
19 b.sort()
20 print(*b)
```



## Bài tập thực hành

### Bài tập 1. SORTING1

Cho vào  $M$  dãy số nguyên  $A_i$  ( $|A_i| < 32000$ ).

Hãy sắp xếp từng dãy số trên theo thứ tự tăng dần.

**Dữ liệu:** Trong  $M$  dòng, mỗi dòng là dãy số, bắt đầu là một số nguyên  $n$  là số lượng các phần tử của dãy số ( $100 \geq n \geq 1$ ),  $n$  số nguyên tiếp theo là giá trị các phần tử của dãy.

**Kết quả:** Ghi ra  $M$  dòng là  $M$  dãy số đã được sắp xếp theo thứ tự tăng dần.

INPUT	OUTPUT
2 2 1	1 2
3 4 3 1	1 3 4
4 1 4 5 2	1 2 4 5

### Bài tập 2. 2Dsort1

Cho vào  $M$  ma trận số nguyên dương  $A[i,j]$  ( $A[i,j] < 32000$ ). Hãy sắp xếp từng ma trận trên theo thứ tự tăng dần, từ trái qua phải, từ trên xuống dưới.

**Dữ liệu:** vào từ file văn bản 2DSORT1.TXT có dạng sau:

+ Dòng thứ nhất ghi một số tự nhiên  $NTEST$  là số ma trận cần sắp xếp.

( $100 \geq NTEST \geq 1$ ). Tiếp theo dữ liệu về  $NTEST$  ma trận như sau:

+ Dòng đầu tiên gồm 2 số là kích thước số dòng  $NDONG$  và số cột  $NCOT$  của ma trận ( $1 \leq NDONG, COT \leq 100$ ).

+  $NDONG$  dòng tiếp theo, mỗi dòng gồm  $NCOT$  số cách nhau ít nhất một khoảng trống là giá trị các phần tử của ma trận.

**Kết quả:** gồm  $NTEST$  bộ dữ liệu ra:

+ Dòng đầu tiên là Số thứ tự  $ITEST$  trong bộ test.

+ Các dòng tiếp theo là các ma trận đã được sắp xếp lại.

INPUT	OUTPUT
1	1
3 3	1 2 3
5 2 1	4 5 6
3 4 6	7 8 9
9 7 8	

### Bài tập 3. SORTING2

Cho vào  $M$  dãy số nguyên  $A_i$  ( $|A_i| < 32000$ )

Hãy sắp xếp từng dãy số trên theo thứ tự:

+ Số chẵn trước, số lẻ sau.

+ Số chẵn theo thứ tự tăng dần, số lẻ theo thứ tự giảm dần.

**Dữ liệu vào:**

Dòng thứ nhất ghi một số tự nhiên  $M$  là số dãy số cần sắp xếp ( $100 \geq M \geq 1$ )

Trong  $M$  dòng tiếp theo mỗi dòng là dãy số, bắt đầu là một số nguyên  $n$  là số lượng các phần tử của dãy số ( $100 \geq n \geq 1$ ),  $n$  số nguyên tiếp theo là giá trị các phần tử của dãy.

**Kết quả:** Ghi ra  $M$  dòng là  $M$  dãy số đã được sắp xếp lại.

INPUT	OUTPUT
2	4 3 1
3 4 3 1	2 4 5 1
4 1 4 5 2	

### Bài tập 4. SORTING3

Cho vào  $M$  dãy số nguyên  $A_i$  ( $|A_i| < 32000$ )

Hãy sắp xếp từng dãy số trên theo thứ tự:

+ Số chẵn xen kẽ số lẻ, số chẵn trước (nếu có).

+ Số chẵn theo thứ tự tăng dần, số lẻ theo thứ tự tăng dần.

**Dữ liệu:** vào từ file văn bản SORTING3.INP có dạng sau:

Dòng thứ nhất ghi một số tự nhiên  $M$  là số dãy số cần sắp xếp ( $100 \geq M \geq 1$ )

Trong  $M$  dòng tiếp theo mỗi dòng là dãy số, bắt đầu là một số nguyên  $n$  là số lượng các phần tử của dãy số,  $n$  số nguyên tiếp theo là giá trị các phần tử của dãy.

**Kết quả:** ghi ra  $M$  dòng là  $M$  dãy số đã được sắp xếp lại.

SORTING3.INP	SORTING3.OUT
3	2 5 6
3 6 2 5	4 1 3
3 3 4 1	2 1 4 5
4 1 4 5 2	



## Bài tập 5. CƠ LOA THÀNH

Truyền thuyết kể rằng Vua An Dương Vương xây thành mãi không xong, thành cứ đắp lên lại đổ. Về sau được Thần Kim Quy mách cho bí quyết xây dựng thành, tuy nhiên công trình thì lớn, bí quyết của Thần vua nghe lại có vẻ phức tạp Vua sợ rằng mình lại thất bại nên nghĩ ra một cách là cáo thị dân chúng, ai quản lý xây dựng thành công được thành theo bí quyết của Thần Kim Quy sẽ được thưởng 1000 con bò, 1000 con ngựa, 1000 con dê..

**Yêu cầu:** Cho vào NTEST ma trận số nguyên dương  $A_{ij}$  ( $A_{ij} < 32000$ ).

Hãy sắp xếp từng ma trận trên theo thứ tự giảm dần theo hình xoắn ốc, từ trái qua phải, từ ngoài vào trong.

**Dữ liệu:**

Dòng thứ nhất ghi một số tự nhiên NTEST ( $100 \geq NTEST \geq 1$ ) là số ma trận cần sắp xếp.

Tiếp theo dữ liệu về NTEST ma trận như sau:

+ Dòng đầu tiên gồm 1 số NDONG là kích thước ma trận (ma trận vuông số dòng bằng số cột) ( $1 \leq NDONG \leq 100$ ).

+ NDONG dòng tiếp theo, mỗi dòng gồm NDONG số cách nhau ít nhất một khoảng trắng là giá trị các phần tử của ma trận.

**Kết quả:** gồm NTEST bộ dữ liệu ra:

- + Dòng đầu tiên là Số thứ tự ITEST trong bộ test.
- + Các dòng tiếp theo là mô hình thành cần xây dựng. Trong đó mỗi số cách nhau một khoảng trắng.

INPUT	OUTPUT
1	1
3	9 8 7
5 2 1	2 1 6
3 4 6	3 4 5
9 7 8	

### Bài tập 6. SORTING4e

Cho vào một dãy các chuỗi, hãy sắp xếp lại các theo thứ tự alphabet.

File dữ liệu đầu vào “Sorting4e.txt”:

+ Số đầu tiên  $N$  là số lượng bộ test.

$N$  bộ test được tổ chức như sau:

+ Dòng đầu tiên là 1 số  $M$  ( $M < 100$ ).

+  $M$  dòng tiếp theo là các chuỗi cần sắp xếp.

Biết rằng các chuỗi đều có độ dài bằng nhau và không quá 100 ký tự.

Dữ liệu ra:

Gồm  $N$  bộ dữ liệu ra:

+ Dòng đầu tiên là Số thứ tự của test trong bộ test.

+ Các dòng tiếp theo là các chuỗi đã được sắp xếp lại.

INPUT	OUTPUT
2	1
3	ab
ac	ac
ba	ba
ab	2
4	abc
baa	abd
abd	aca
abc	baa
aca	



## DẠNG BÀI SỬ DỤNG THUẬT TOÁN

### Bài 1. Thuật toán đệ quy - RECURSION

Đệ quy (tiếng Anh: *RECURSION*) là phương pháp dùng trong các chương trình máy tính trong đó có một hàm tự gọi chính nó.

#### Ví dụ: Tính giai thừa

*Tìm giai thừa của n.*

**Input:** Dòng đầu là số lượng test.

Mỗi dòng tiếp theo gồm 1 số nguyên  $n$  ( $0 < n < 13$ ).

**Output:** Với mỗi test, hãy in ra số  $n!$  theo mẫu sau:

INPUT	OUTPUT
2	$3! = 6$
3	$4! = 24$
4	

Giai thừa của  $n$  chính là tích các số từ 1 tới  $n$ .

Vậy:

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times (n-1) \times n$$

$$(n-1)! = 1 \times 2 \times 3 \times 4 \times \dots \times (n-2) \times (n-1)$$

$$(n-2)! = 1 \times 2 \times 3 \times 4 \times \dots \times (n-3) \times (n-2)$$

Từ 3 biểu thức trên, ta có thể thấy,

$$n! = (n-1)! \times n$$

$$(n-1)! = (n-2)! \times (n-1)$$

...

Qua đó, ta sẽ tạo hàm **giaithua(a)** với  $a$  là giá trị truyền vào để tính giai thừa. Khi tính giai thừa, ta kiểm tra nếu  $a$  bằng 1 sẽ trả về giá trị 1, còn không thì trả về kết quả là  $a$  nhân với **(a-1)!**.

Trước tiên, ta mở 2 file dữ liệu của bài toán để đọc và ghi dữ liệu.

```
1  file = open('INPUT.INP')
2  file2 = open('OUTPUT.OUT', 'w')
```

Tiếp đó, ta định nghĩa hàm giao thừa với giá trị truyền vào hàm là số cần tính giao thừa.

```
4 def giaithua(x):
5     if x == 1:
6         return 1
7     else:
8         return x * giaithua(x-1)
```

Để lấy giá trị số test , ta đọc dữ liệu dòng đầu tiên trong file dữ liệu vào và thực hiện lặp lại tương ứng số lần đọc được việc đọc giá trị **n**, tính giao thừa của **n** và ghi kết quả vào file dữ liệu ra của bài toán.

```
9 n = int(file.readline())
10 for i in range(n):
11     a = int(file.readline())
12     kq = giaithua(a)
13     if i > 0:
14         file2.write('\n')
15     file2.write(str(a) + '! = ' + str(kq))
```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```
17 file.close()
18 file2.close()
```

## Bài tập thực hành

### Bài tập 1. F91

McCarthy là một nhà khoa học máy tính nổi tiếng, ông ta đã định nghĩa hàm đệ quy f91 như sau:

Nếu  $N \leq 100$ , thì  $f91(N) = f91(f91(N+11))$ ;

Nếu  $N \geq 101$ , thì  $f91(N) = N - 10$ .

Hãy viết một chương trình tính toán hàm McCarthy's f91.

**Input:** File input chứa một dãy các số nguyên dương, mỗi số không quá 1,000. Mỗi số trên một dòng.

**Output:** Hiện ra theo định dạng trong ví dụ sau:

INPUT	OUTPUT
500	$f91(500) = 490$
91	$f91(91) = 91$



### Bài tập 2. Chính hợp

Tìm tất cả các chính hợp chập k của n phần tử từ 1 đến n ( $0 < k \leq n < 10$ ).

**Input:** 1 dòng gồm 2 số nguyên n và k.

**Output:** In ra số chính hợp tìm được và liệt kê các chính hợp, giữa các test là 1 dòng trống.

INPUT	OUTPUT	INPUT	OUTPUT
1 3	3 1 2 3	2 3	6 1 2 1 3 2 1 2 3 3 1 3 2

### Bài tập 3. Hoán vị (Nghệ An, 2019)

Viết chương trình in ra tất cả các hoán vị của N. N nguyên dương ( $0 < N \leq 10$ ), N được nhập từ bàn phím.

Ví dụ:

INPUT	OUTPUT
2	1 2 2 1
3	1 2 3 1 3 2 2 1 3 2 3 1 3 2 1 3 1 2

### Bài tập 4. Trò chơi số học

Một trò chơi phổ biến của trẻ em với bảng NxN ô ( $2 \leq N \leq 5$ ). Trong mỗi ô chứa 1 số có 1 chữ số từ 1 đến 9. Với một số số cho trước, hãy điền các số còn lại vào ô sao cho tổng các hàng ngang bằng nhau và bằng tổng các hàng dọc.

**Input:** “Ngame.inp”

Số đầu tiên là 1 số nguyên Ntest là số lượng test của bài, mỗi test gồm có:

- + Dòng đầu tiên là số N.
- + Tiếp theo là ma trận N x N trong đó số 0 là ô trắng cần điền.

**Output:** “Ngame.out”

Với mỗi test, nếu có thể tìm ra đáp án thỏa mãn quy luật của bảng thì in ra ma trận N x N một cách điền bất kỳ. Nếu không có kết quả nào in ra 1 chuỗi: “KHONG TIM DUOC.”

Giữa các test cách nhau một dòng trắng.

Ngame1.inp	Ngame1.out
1	1 4 5
3	6 3 1
1 0 5	3 3 4
0 3 0	
3 0 4	

### Bài tập 5. Tổ hợp

Tìm tất cả các tổ hợp chập k của n phần tử từ 1 đến n ( $0 < k \leq n < 10$ ).

**Input:** Dòng đầu gồm một số tự nhiên NTEST là số lượng test của file input. Mỗi test 1 dòng gồm 2 số nguyên n và k.

**Output:** Với mỗi test, hãy in ra số tổ hợp tìm được và liệt kê các tổ hợp, giữa các test là 1 dòng trắng.

INPUT	OUTPUT
2	3
1 3	1
2 3	2
	3
	3
	1 2
	1 3
	2 3



## Bài 2. Thuật toán tìm kiếm – SEARCH ALGORITHMS

Cùng với sắp xếp, tìm kiếm cũng là hoạt động hằng ngày chúng ta thường xuyên sử dụng trong các ứng dụng tin học. Trong cuốn sách sẽ giới thiệu 2 kỹ thuật tìm kiếm cơ bản nhất.

### a. Tìm kiếm tuần tự - SEQUENTIAL SEARCH

Tìm kiếm tuần tự là kỹ thuật tìm kiếm đơn giản nhất. Chương trình hoạt động bằng cách so sánh từ khóa tìm kiếm với từng phần tử trong danh sách cho tới khi tìm thấy hoặc đã tìm hết danh sách mà chưa thấy kết quả.

Tuy nhiên khi danh sách quá dài, kỹ thuật tìm kiếm này mất khá nhiều thời gian thực thi. Do vậy kỹ thuật này chỉ được sử dụng trong các bài toán nhỏ.

### b. Tìm kiếm nhị phân – BINARY SEARCH

Tìm kiếm nhị phân là một thuật toán dùng để tìm kiếm phần tử trong một danh sách đã được sắp xếp. Thuật toán hoạt động như sau: Trong mỗi bước, so sánh phần tử cần tìm với phần tử nằm ở chính giữa danh sách. Nếu hai phần tử bằng nhau thì phép tìm kiếm thành công và thuật toán kết thúc. Nếu chúng không bằng nhau thì tùy vào phần tử nào lớn hơn, thuật toán lặp lại bước so sánh trên với nửa đầu hoặc nửa sau của danh sách. Vì số lượng phần tử trong danh sách cần xem xét giảm đi một nửa sau mỗi bước, nên thời gian thực thi của thuật toán nhanh hơn rất nhiều so với thuật toán tìm kiếm tuần tự nhưng cũng có một số nhược điểm. Nếu nội dung danh sách bị thay đổi thì danh sách phải được sắp xếp lại trước khi sử dụng tìm kiếm nhị phân. Thao tác này thường tốn nhiều thời gian.

#### Ví dụ: Bí ẩn xà phòng

*Một chàng trai đi mua xà phòng từ một cửa hàng. Cửa hàng có N bánh xà phòng. Giá của xà phòng được đưa ra dưới dạng một mảng A. Giá của xà phòng thứ i là A[i]. Hãy giúp chàng trai biết số lượng xà phòng có giá thấp hơn tiền hiện có của anh ta.*

#### Đầu vào:

- + Dòng đầu tiên chứa số nguyên N là tổng số bánh xà phòng có sẵn trong cửa hàng.
- + Dòng thứ hai chứa N số nguyên, mỗi số cách nhau 1 dấu cách.
- + Dòng thứ ba chứa ntest là số lượng test của bài toán.
- + Ntest dòng tiếp theo chứa số nguyên M là số tiền hiện có của chàng trai.

#### Đầu ra:

- + Với mỗi test đưa ra số lượng xà phòng có giá thấp hơn M.

<b>INPUT.INP</b>	<b>OUTPUT.OUT</b>
5	1
1 4 10 5 6	1
4	2
2	5
3	
5	
11	

Thuật toán tìm kiếm tuần tự rất đơn giản, các bạn có thể tự nghiên cứu và thực hành. Trong bài sẽ hướng dẫn cách áp dụng thuật toán tìm kiếm nhị phân. Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1 file = open("INPUT.INP")
2 file2 = open("OUTPUT.OUT", "w")
```

Ta đọc các dữ liệu vào của bài toán và lưu vào biến tương ứng. Để sử dụng thuật toán tìm kiếm nhị phân, dãy số cần được sắp xếp trước khi tìm kiếm. Chính vì vậy, ta thực hiện chuyển đổi kiểu dữ liệu đầu vào thành danh sách các số, sau đó sử dụng phương thức sort() để sắp xếp danh sách các số đã đọc từ file dữ liệu đầu vào của bài toán.

```
4 a = []
5 n = int(file.readline())
6 data = file.readline()
7 data = data.split()
8 for i in range(n):
9     a.append(int(data[i]))
10 a.sort()
11 print(a)
```

Để tìm kiếm phần tử trong dãy, ta tạo hàm **search(i)** với **i** là tham số được truyền vào chính là vị trí giữa của dãy số. Bên trong hàm, ta xét 6 trường hợp:

+ Trường hợp 1: Tất cả các số trong dãy số đều nhỏ hơn giá trị **m**:

```
13 def search(i):
14     global dau
15     global cuoi
16     if a[n-1] < m:
17         return n
```

+ Trường hợp 2: Giá trị tại vị trí chính giữa bằng giá trị **m**:



```
19     if a[i] == m:
20         return i
```

+ Trường hợp 3: Giá trị tại vị trí **i** lớn hơn **m** và giá trị tại vị trí **i - 1** nhỏ hơn **m**:

```
21     if a[i] > m and a[i-1] < m:
22         return i
```

+ Trường hợp 4: Giá trị tại vị trí **i** nhỏ hơn **m** và giá trị tại vị trí **i + 1** lớn hơn **m**:

```
23     if a[i] < m and a[i+1] > m:
24         return i+1
```

+ Trường hợp 5: Giá trị tại vị trí **i** lớn hơn **m**, ta thu ngắn đoạn dãy số cần duyệt:

```
25     if a[i] > m:
26         cuoi = i
```

+ Trường hợp 6: Giá trị tại vị trí **i** nhỏ hơn **m**, ta thu ngắn đoạn dãy số cần duyệt:

```
27     elif a[i] < m:
28         dau = i
```

Sau khi đã thu ngắn được đoạn dãy số, ta thực hiện tìm kiếm trong đoạn dãy số đã thu gọn trong hai trường hợp 5 và 6.

```
29     i = int((dau+cuoi)/2)
30     return search(i)
```

Trong đoạn chương trình chính, ta đọc dữ liệu **ntest**, với mỗi **test** ta đọc giá trị **m** và tạo biến **dau**, **cuoi** lưu vị trí đầu và vị trí cuối của đoạn con trong dãy số cần tìm kiếm.

```
32 ntest = int(file.readline())
33 for test in range(ntest):
34     m = int(file.readline())
35     dau = 0
36     cuoi = n - 1
```

Tiếp đó, ta thực hiện tính vị trí giữa của dãy số và tìm kiếm bằng cách gọi hàm **search(i)**. Ta lưu kết quả bài toán vào biến **kq** sau đó ghi **kq** vào file dữ liệu ra.

```
38     i = int((dau + cuoi) / 2)
39     kq = search(i)
40     print(kq)
41     if test > 0:
42         file2.write("\n")
43         file2.write(str(kq))
44 file.close()
45 file2.close()
```

## Bài tập thực hành

### Bài tập 1: Tìm kiếm cơ bản

Cho một dãy số A gồm N số: A<sub>1</sub>, A<sub>2</sub>, ..., A<sub>N</sub>.

Tìm vị trí giá trị K trong danh sách đã cho.

**Dữ liệu vào:**

+ Dòng đầu tiên chứa giá trị N

+ Dòng thứ 2 gồm N số, mỗi số cách nhau 1 dấu cách

+ Dòng thứ 3 chứa giá trị K

**Dữ liệu ra:**

Vị trí của K trong dãy số đã cho.

INPUT	OUTPUT
5	3
1 2 3 4 5	
4	

### Bài tập 2: Tìm xâu (Quảng Ngãi, 2019)

Xâu P được gọi là xuất hiện tại vị trí i của xâu Q nếu P[1]=Q[i], P[2]=Q[i+1]..., P[n]=Q[i+n-1] với n là độ dài của xâu P.

Cho hai xâu P và Q chỉ gồm các ký tự chữ cái thường. Hãy viết chương trình tìm tất cả các vị trí mà P xuất hiện trong Q.

**Dữ liệu vào:** Nhập từ bàn phím:

- Dòng đầu nhập xâu P.

- Dòng thứ 2 nhập xâu Q.

Độ dài hai xâu P và Q không quá 1000 ký tự.

**Kết quả:** Xuất ra màn hình các vị trí tìm được theo thứ tự tăng dần. Nếu P không xuất hiện trong Q thì xuất ra màn hình số 0.

Dữ liệu vào	Kết quả
ab	3 7
cbabbcab	
cba	0
abcdcbe	



### Bài tập 3. HÀNG CÂY

Trong khu vườn, người ta trồng một hàng cây chạy dài gồm có  $N$  cây, mỗi cây có độ cao là  $a_1, a_2, \dots, a_N$ .

Người ta cần lấy  $M$  mét gỗ bằng cách đặt cưa máy sao cho lưỡi cưa ở độ cao  $H$  (mét) để cưa tất cả các cây có độ cao lớn hơn  $H$  (dĩ nhiên những cây có độ cao không lớn hơn  $H$  thì không bị cưa).

Ví dụ: Nếu hàng cây có các cây với độ cao tương ứng là 20; 15; 10 và 18 mét, cần lấy 7 mét gỗ. Lưỡi cưa đặt tại độ cao hợp lý là 15 mét thì độ cao của các cây còn lại sau khi bị cưa tương ứng là 15; 15; 10 và 15 mét. Tổng số mét gỗ lấy được là 8 mét (dư 1 mét).

**Yêu cầu:** Hãy tìm vị trí đặt lưỡi cưa hợp lý (số nguyên  $H$  lớn nhất) sao cho lấy được  $M$  mét gỗ và số mét gỗ dư ra là ít nhất.

**Dữ liệu:**

- Dòng thứ nhất chứa 2 số nguyên dương  $N$  và  $M$  cách nhau một dấu cách.
- Dòng thứ hai chứa  $N$  số nguyên dương ai là độ cao của mỗi cây trong hàng ( $1 \leq a_i \leq 10^9$ ;  $i = 1, \dots, N$ ), mỗi số cách nhau ít nhất một dấu cách.

**Kết quả:** Đưa ra màn hình một số nguyên cho biết giá trị cần tìm.

INPUT	OUTPUT
47	15
20 15 10 18	



## Giải đề thi chính thức

Chương này sẽ giải trọng bộ đề thi Tin học trẻ Quốc gia, đề thi sơ khảo Quốc gia và đề thi của các tỉnh/thành phố lớn như Hà Nội, Hồ Chí Minh, Đà Nẵng và Nghệ An. Các đề thi trong sách có thể không sử dụng hình ảnh gốc của đề mà dùng những hình ảnh tương đương. Theo định hướng của Hội thi Tin học trẻ, những năm trước đề thi gồm nhiều nội dung khác nhau như lập trình Free Pascal, C/C++, Scratch, soạn thảo văn bản MSWord, slide trình chiếu PowerPoint.... Từ năm 2020 ngôn ngữ lập trình Python sẽ được bổ sung thêm trong các đề thi, vì vậy với một số đề thi sử dụng Pascal hoặc C, trong sách sẽ được thực hiện với ngôn ngữ lập trình Python. Những nội dung khác ngoài lập trình như soạn thảo văn bản, vẽ paint, soạn thảo powerpoint, Scratch... vẫn được nêu lên trong sách nhưng sẽ không hướng dẫn thực hiện, bạn đọc chủ động ôn luyện các dạng bài nêu trên (có thể tìm hiểu thêm trên Internet).



## ĐỀ THI QUỐC GIA NĂM 2019

### Bài 1. Trung bình cộng

Cho một số thực  $R$ . Tìm số lượng số nguyên ít nhất để trung bình cộng của chúng bằng  $R$ .

Dữ liệu nhập vào sẽ có 10 test, mỗi test chứa một số thực  $R$ .

Em cần tạo file văn bản TBC.TXT gồm 10 dòng, mỗi dòng chứa duy nhất một nguyên dương là số lượng nhỏ nhất các số nguyên thỏa mãn đề bài. Mỗi dòng ghi kết quả của một test tương ứng. Nếu test nào không tìm thấy kết quả, để trống dòng của test đó.

<b>Test</b>	<b>Input</b>	<b>TBC.TXT</b>
1	2.5	2
2	8.125	
3	1.0625	
4	5.0475	
5	11.5936	
6	2019.0873984	
7	1707.740873728	
8	2.237663674368	
9	4.47532757942272	
10	1.8036983803445248	

**Giải thích test 1:** 2.5 là trung bình cộng của ít nhất 2 số nguyên, ví dụ: trung bình cộng của 1 và 4 là 2.5.

**Phân tích:** Để giải bài toán, ta thực hiện chuyển đổi số nhập vào thành dạng phân số rút gọn nhờ thư viện fractions. Sau khi chuyển đổi, đáp án cần tìm chính là mẫu số trong phân số đó.

Trước tiên, ta thực hiện import thư viện Fraction, sau đó mở file dữ liệu vào của bài toán.

```
from fractions import Fraction
file = open("TBC.TXT")
```

Trong bài có 10 test, vậy ta tiến hành lặp lại 10 lần việc đọc dữ liệu từ file dữ liệu vào.

```
for test in range(10):
    num = float(file.readline())
```

Tiếp đó ta chuyển đổi số thực đã đọc được thành phân số. Để lấy được tử số và mẫu số, ta chuyển đổi phân số thành kiểu dữ liệu chuỗi và tách chuỗi bởi dấu “/”. Kết quả chính là phần tử có chỉ số trong danh sách sau khi tách.

```
s = str(Fraction(num))
s = s.split("/")
print(s[1])
```

Cuối cùng, ta đóng file dữ liệu sau khi đã hoàn thành và tìm kết quả các test của bài toán.

```
file.close()
```

## Bài 2. Phân số

*Cho  $N$  phân số được xếp cạnh nhau theo một thứ tự cho trước. Hãy điền  $N - 1$  dấu phép tính: cộng, trừ hoặc nhân (+, -,  $\times$ ) vào để được biểu thức có kết quả là một phân số  $R$  cho trước.*

Dữ liệu nhập vào sẽ có 10 test, mỗi test sẽ có định dạng như sau: dòng đầu tiên chứa các phân số, các phân số cách nhau bởi dấu cách. Dòng thứ 2 chứa một phân số là kết quả của biểu thức. Tử số và phân số phân cách bởi dấu chia “/”.

*Em cần tạo file văn bản PHANSO.TXT gồm 10 dòng, mỗi dòng gồm lần lượt các ký tự dấu phép tính của biểu thức từ trái qua phải tương ứng với dữ liệu cho trong bảng dưới đây. Dữ liệu cho đảm bảo luôn có kết quả. Mỗi dòng ghi kết quả của test tương ứng. Nếu test nào không tìm thấy kết quả, để trống dòng của test đó.*

Test	Input	PHANSO.TXT
1	1/2 1/3 3/4 1/8	-x
2	5/2 4/6 3/8 -2/3 19/12	
3	3/4 -2/7 -3/14 1/2 -2/1 -10/7	



4	$5/6 \ 1/9 \ 3/2 \ -2/2 \ 2/3 \ 1/6$ $7/6$	
5	$2/9 \ 3/4 \ -5/8 \ -2/3 \ 7/3 \ 1/9$ $29/72$	
6	$1/4 \ 2/3 \ 3/2 \ 4/1 \ -1/4 \ -2/3 \ -3/2 \ -4/1$ $-13/12$	
7	$-1/16 \ 12/24 \ 3/12 \ 10/8 \ 2/6 \ -14/18 \ 5/3 \ -5/2$ $137/36$	
8	$1/2 \ 2/3 \ 3/4 \ 4/5 \ 5/6 \ 6/7 \ 7/8 \ 8/9 \ 9/10 \ 10/11 \ 11/12 \ 12/13$ $-363401/90090$	
9	$1/2 \ 1/3 \ 1/4 \ 1/6 \ 1/8 \ 2/9 \ 2/5 \ 2/1 \ 2/2 \ 2/3 \ 1/4 \ 1/6 \ 1/8 \ 1/9 \ 1/5$ $-43/80$	
10	$1/12 \ 19/15 \ 8/13 \ 17/4 \ 14/13 \ 3/12 \ 3/4 \ 17/4 \ 7/19 \ 12/15 \ 20/6$ $7/9 \ 8/8 \ 5/6 \ 6/13 \ 19/15 \ 18/18 \ 10/7 \ 7/3 \ 17/2$ $337643/35568$	

*Giải thích test 1:  $1/2 - 1/3 \times 3/4 = 1/4$*

*Chú ý: ký tự dấu phép nhân là chữ X viết hoa.*

Để giải bài toán, ta sử dụng thuật toán đệ quy để tìm các phép toán.

Trước tiên, ta import thư viện fractions và mở dữ liệu vào của bài toán.

```
1 from fractions import *
2
3 file = open("INPUT.INP")
4 file2 = open("OUTPUT.OUT", "w")
```

Ta tạo danh sách a lưu các phân số của bài toán, tiếp đó đọc dòng đầu tiên trong file dữ liệu vào là các phân số, sau đó tách và lưu vào danh sách a.

```
6 a = []
7 data = file.readline()
8 data = data.split()
9 n = len(data)
10
11 for i in range(n):
12     s = data[i]
13     s = s.split("/")
14     tu = int(s[0])
15     mau = int(s[1])
16     a.append(Fraction(tu, mau))
```

Sau khi đọc và tách dữ liệu, ta đọc dòng dữ liệu thứ 2 tương ứng kết quả của các phân số bên trên.

```
18 data = file.readline()
19 data = data.split("/")
20 kq = Fraction(int(data[0]), int(data[1]))
```

Ta tạo hàm **tinh(s)** để tính giá trị các phân số với lần lượt các phép tính được lưu trong tham số **s** được truyền vào. Ta tạo biến **m** lưu kết quả của phép tính, trong toán học, ta cần thực hiện phép nhân trước và cộng trừ sau. Do vậy ta thực hiện nhân các số tại vị trí có ký tự “X” trong chuỗi **s**. Tuy nhiên ta cần lưu ý khi có các phép nhân liền nhau, ta cần nhân tới khi phép tính không phải phép nhân nữa.

```
22 def tinh(s):
23     global kq
24     global n
25     global a
26     m = 0
27     i = 0
28     while i < n-1:
29         if s[i] == "X":
30             temp = a[i] * a[i+1]
31             if i > 0 and s[i-1] == "-":
32                 temp = -temp
33             i += 1
34             while i < n-2 and s[i] == "X":
35                 temp *= a[i+1]
36                 i += 1
37             m += temp
38         else:
39             i += 1
```

Sau khi tính được các phép toán nhân, ta tính các phép cộng và trừ theo thứ tự từ trái sang phải. Tuy nhiên ta cần lưu ý khi phép toán đầu tiên là phép nhân, ta không cần cộng phân số đầu tiên vào **m**, trường hợp phép tính đầu tiên là cộng hoặc trừ, ta cần cộng **m** với phân số đầu tiên.

```
41     if s[0] != "X":
42         m += a[0]
43
44     i = 0
45     while i < n - 1:
46         if i == n-2:
47             if s[i] == "+":
48                 m += a[i+1]
49             elif s[i] == "-":
50                 m -= a[i+1]
51             elif s[i] == "+" and s[i+1] != "X":
52                 m += a[i+1]
53             elif s[i] == "-" and s[i+1] != "X":
54                 m -= a[i+1]
55         i += 1
```



Khi đã tính được hết các phép cộng, trừ và nhân, ta kiểm tra nếu **m** bằng **kq**, hàm trả về giá trị True, còn không thì trả về giá trị False.

```
57     if m == kq:
58         return True
59     else:
60         return False
```

Để tạo chuỗi **s** gồm các phép toán, ta tạo hàm **pheptoan()** với các tham số truyền vào lần lượt là **stt**, **pt**, **s** tương ứng là vị trí phép toán, phép toán thêm vào và chuỗi **s**. Trong đó **pt** sẽ nhận giá trị từ 0 tới 2 với quy ước như sau:

- + **pt** = 0: Phép toán cộng.
- + **pt** = 1: Phép toán trừ.
- + **pt** = 2: Phép toán nhân.

Trong hàm ta kiểm tra giá trị **pt** và nối vào chuỗi **s** ký tự tương ứng với phép toán. Sau khi nối, ta kiểm tra **stt** có phải là phép toán cuối cùng hay không, nếu là phép toán cuối cùng, ta thực hiện tính và kiểm tra kết quả, còn không thì lặp lại việc thêm phép toán và thay đổi phép toán bằng vòng lặp for.

```
62 def pheptoan(stt, pt, s):
63     global n
64
65     if pt == 0:
66         s += "+"
67     elif pt == 1:
68         s += "-"
69     else:
70         s += "X"
71
72     if stt == n-1:
73         if tinh(s):
74             print(s)
75     else:
76         for j in range(3):
77             pheptoan(stt+1, j, s)
```

Trong đoạn chương trình chính, ta thực hiện gọi hàm tạo phép toán tại vị trí đầu tiên với vòng lặp for.

```
79 for i in range(3):
80     pheptoan(1, i, "")
```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán sau khi đã hoàn tất việc đọc và ghi dữ liệu.

```
82 file.close()
83 file2.close()
```

## ĐỀ THI SƠ KHẢO QUỐC GIA NĂM 2020

### Bài 1. Phần thưởng

An là người thắng cuộc trong cuộc thi “Tim hiểu Đoàn Thanh niên Cộng sản Hồ Chí Minh” và được nhận phần thưởng của Ban tổ chức. Ban tổ chức chuẩn bị một bảng kích thước  $m \times n$ . Các dòng của bảng được đánh số từ 1 đến  $m$ , từ trên xuống dưới, dòng  $i$  ( $1 \leq i \leq m$ ) có trọng số là  $a_i$ . Các cột của hàng được đánh số từ 1 đến  $n$ , từ trái qua phải, cột  $j$  ( $1 \leq j \leq n$ ) có trọng số là  $b_j$ . Ô nằm trên giao của dòng  $i$  và cột  $j$  được gọi là ô  $(i, j)$  và trên ô đó ghi một số nguyên có giá trị  $a_i + b_j$ .

Để nhận phần thưởng, An được phép chọn một bảng con kích thước  $w \times h$  chiếm trọn  $w \times h$  ô của bảng và phần thưởng mà An nhận được sẽ có giá trị bằng tổng giá trị các ô nằm trong bảng con đó.

**Yêu cầu:** Hãy xác định tổng giá trị lớn nhất mà An có thể nhận được.

**Dữ liệu:** Vào từ file văn bản BONUS.INP

- + Dòng thứ nhất chứa bốn số nguyên dương  $m, n, w, h$  ( $w \leq m; h \leq n$ );
- + Dòng thứ hai chứa  $m$  số nguyên  $a_1, a_2, \dots, a_m$  ( $|a_i| \leq 10^6$ ,  $i = 1, 2, \dots, m$ );
- + Dòng thứ ba chứa  $n$  số nguyên  $b_1, b_2, \dots, b_n$  ( $|b_i| \leq 10^6$ ,  $i = 1, 2, \dots, n$ ).

**Kết quả:** Ghi ra file văn bản BONUS.OUT một số nguyên duy nhất là tổng giá trị lớn nhất mà An có thể nhận được.

Ràng buộc:

- + Có 20% số test ứng với  $m, n \leq 10$  và  $w = h = 1$ ;
- + 30% số test khác ứng với  $m, n \leq 10$ ;
- + 20% số test khác ứng với  $m, n \leq 10^3$ ;
- + 30% số test còn lại có  $m, n \leq 10^5$ .

Ví dụ:

Dữ liệu vào	Kết quả ra	Giải thích				
3 4 2 2	6					
1 -1 2			Cột	1	2	3
1 1 1 1			Dòng	(1)	(1)	(1)
			1	2	2	2
			2	0	0	0
			3	3	3	3
			(-1)			
			(2)			



Bảng kích thước  $3 \times 4$ , trọng số của các hàng và các cột được ghi trong ngoặc ở hàng và cột tương ứng. Một cách chọn bảng con kích thước  $2 \times 2$  là hình được tô màu có tổng giá trị bằng 6.

**Phân tích:** Để giải bài toán, ta tạo danh sách  $a$  và  $b$  để lưu các số nguyên, tạo ma trận  $c$  gồm  $m$  hàng  $n$  cột với các giá trị mặc định ban đầu là 0. Tiếp đó sử dụng hai vòng lặp for lồng nhau. Vòng lặp thứ nhất để lấy chỉ số hàng ( $i$ ), vòng lặp thứ hai để lấy chỉ số cột ( $j$ ). Với mỗi cặp hàng, cột ta thực hiện tính  $c[i][j]$  bằng tổng  $a[i] + b[j]$ .

Sau khi tính được ma trận  $c$ , ta tiến hành duyệt các bảng con và tính tổng của các bảng còn, từ đó tìm ra tổng lớn nhất của bảng con.

Trước tiên, ta import thư viện numpy và mở hai file dữ liệu của bài toán.

```
1 from numpy import *
2 file = open("BONUS.INP")
3 file2 = open("BONUS.OUT", "w")
```

Ta đọc dữ liệu từ file dữ liệu vào và lưu vào các biến tương ứng theo yêu cầu của đề bài.

```
5 data = file.readline()
6 data = data.split()
7 m = int(data[0])
8 n = int(data[1])
9 w = int(data[2])
10 h = int(data[3])
```

Trước khi đọc dãy  $a$  và dãy  $b$ , ta tiến hành tạo danh sách **a**, **b** và ma trận **c** gồm **m** hàng, **n** cột và các giá trị trong **c** đều bằng 0.

```
12 a = []
13 b = []
14 c = zeros((m, n))
```

Khi đã tạo danh sách **a** và **b**, ta tiến hành quét dữ liệu và thêm các giá trị và danh sách tương ứng.

```
16 data = file.readline()
17 a = data.split()
18 data = file.readline()
19 b = data.split()
```

Trước khi tính các giá trị trong ma trận **c**, ta tạo hàm tính tổng các giá trị trong bảng con với hai tham số **x** và **y** được truyền vào tương ứng vị trí hàng và cột bắt đầu của bảng con. Hàm này sẽ trả về tổng các giá trị trong bảng con bắt đầu từ hàng **x**, cột **y** và có chiều rộng **w**, chiều cao **h**.

```

21 def tinh(x, y):
22     global w
23     global h
24     tong = 0
25     for i in range(h):
26         for j in range(w):
27             tong += c[x+i][y+j]
28     return tong

```

Sau khi khởi tạo hàm, ta tiến hành tính các giá trị trong ma trận **c** theo yêu cầu của đề bài.

```

30 for i in range(m):
31     for j in range(n):
32         c[i][j] = int(a[i]) + int(b[j])

```

Khi đã tính được các giá trị trong ma trận **c**, ta tiến hành duyệt các vị trí có thể là vị trí bắt đầu của bảng con, sau đó tính giá trị bảng con từ vị trí bắt đầu đó. Ta tạo biến **max** để lưu giá trị lớn nhất của bảng con. Với vị trí đầu tiên trong ma trận, ta đặt **max** bằng giá trị tổng bảng con tính được. Các vị trí phía sau ta cần so sánh và kiểm tra, nếu **max** nhỏ hơn tổng bảng con phía sau, ta sẽ đổi giá trị **max** thành tổng của bảng con đó.

```

34 for i in range (m-h+1):
35     for j in range (n-w+1):
36         if i == 0 and j == 0:
37             max = tinh(i, j)
38         else:
39             temp = tinh(i, j)
40             #     print(temp)
41             if temp > max:
42                 max = temp

```

Cuối cùng, ta ghi kết quả của bài toán vào file dữ liệu ra của bài toán và đóng hai file dữ liệu lại. Tuy nhiên, chúng ta cần chuyển đổi kết quả thành số nguyên trước khi ghi vào file.

```

44 max = int(max)
45 file2.write(str(max))
46 file.close()
47 file2.close()

```

## Bài 2. Đề thi

*Hội thi Tin học trẻ được tổ chức hàng năm và đã thu hút được sự quan tâm của cả nước. Đề thi ngày càng phong phú và đa dạng là do sự đóng góp ý tưởng từ rất nhiều nhà khoa học và các tổ chức công nghệ. Đến nay, ngân hàng đề thi có tổng cộng  $n$  bài, các bài được đánh số từ 1 đến  $n$ , bài thứ  $i$  có độ khó là  $i$ . Để xây dựng đề*



thi năm nay, Ban giám khảo muốn chọn k bài khác nhau từ ngân hàng đề thi mà tổng độ khó của k bài đúng bằng n. Để khảo sát tính đa dạng của đề thi, Ban giám khảo muốn tính số cách xây dựng đề thi khác nhau (hai đề thi được gọi là khác nhau nếu có một bài được chọn trong đề thứ nhất nhưng không được chọn trong đề thứ hai).

**Yêu cầu:** Cho n và k, hãy giúp Ban giám khảo tính số cách xây dựng đề thi khác nhau. Vì kết quả có thể rất lớn nên chỉ cần đưa ra số dư của phép chia kết quả tìm được cho ( $10^9 + 7$ ).

**Dữ liệu:** Vào từ thiết bị nhập chuẩn hai số nguyên dương n, k.

**Kết quả:** Ghi ra thiết bị xuất chuẩn một số nguyên duy nhất là số dư của phép chia kết quả tìm được cho ( $10^9 + 7$ ).

Ràng buộc:

- + Có 20% số test ứng với  $n \leq 100$  và  $k \leq 5$ ;
- + 20% số test khác ứng với  $n \leq 10^6$  và  $k \leq 5$ ;
- + 20% số test khác ứng với  $n \leq 10^9$  và  $k = 2$ ;
- + 20% số test khác ứng với  $n \leq 10^9$  và  $k = 3$ ;
- + 20% số test còn lại có  $m$ ,  $n \leq 10^9$  và  $k \leq 5$ .

Dữ liệu vào	Dữ liệu ra	Giải thích
10 3	4	Có 4 cách tạo một đề thi gồm 4 bài mà tổng độ khó bằng 10 được liệt kê dưới đây: $1 + 2 + 7 = 10$ $1 + 3 + 6 = 10$ $1 + 4 + 5 = 10$ $2 + 3 + 5 = 10$

Để giải bài toán, ta sử dụng thuật toán đệ quy. Ta tạo hàm để tìm đề với 3 chỉ số được truyền vào lần lượt là thứ tự bài trong đề, thứ tự bài trong danh sách và tổng độ khó hiện tại của đề.

Trước tiên, ta mở 2 file dữ liệu của bài toán, đọc dữ liệu và khởi tạo các biến.

```

1  file = open('INPUT.INP')
2  file2 = open('OUTPUT.OUT', 'w')
3
4  data = file.readline()
5  data = data.split()
6
7  n = int(data[0])
8  k = int(data[1])
9  count = 0

```

Tiếp đó ta tạo hàm **timde()** với 3 chỉ số cần truyền vào lần lượt là **i**, **a**, **tong** tương ứng thứ tự bài trong đề, thứ tự bài trong danh sách và tổng độ khó hiện tại của đề. Bên trong hàm, ta kiểm tra giá trị **i**, nếu **i = k** thì kiểm tra giá trị **tong** truyền vào có đúng độ khó **n** theo yêu cầu hay không, nếu bằng ta tăng biến **count** thêm 1.

```
11 def Timde(i, a, tong):
12     global count
13     if i == k:
14         if tong == n:
15             count += 1
```

Trường hợp còn lại khi **i** nhỏ hơn **k**, ta tiến hành tìm bài tiếp theo trong đề với chỉ số bài bắt đầu tăng dần từ **a + 1**. Tuy nhiên ta chỉ tiến hành tiếp tục tìm các bài tiếp theo trong đề khi tổng độ khó của đề chưa vượt quá độ khó theo yêu cầu của đề bài.

```
16     elif i < k:
17         for j in range (a+1, n):
18             if tong <= n:
19                 Timde(i+1, j, tong+j)
20             else:
21                 break
```

Trong đoạn chương trình chính, ta sử dụng vòng lặp **for** để thiết lập bài đầu tiên trong đề thi và gọi hàm **dethi()** với 3 chỉ số truyền vào.

```
23 for i in range (1,n-k):
24     Timde(1, i, i)
```

Đề bài yêu cầu chúng ta tìm phép chia lấy dư của số lượng đề tìm được cho  $10^9 + 7$ . Vậy ta thực hiện tính kết quả và ghi kết quả vào file dữ liệu ra của bài toán.

```
26 kq = count % (10**9 + 7)
27 print(kq)
28
29 file2.write(str(kq))
30 file.close()
31 file2.close()
```



## ĐỀ THI THÀNH PHỐ HÀ NỘI NĂM 2020

### Bài 1. SQ

Cho ba số nguyên dương  $a$ ,  $b$ ,  $N$ . Hãy tính tổng  $N$  chữ số sau dấu phẩy của phép chia  $a$  cho  $b$ . Có thể thêm vô hạn số 0 vào cuối phần thập phân.

<b>INPUT</b>	<b>OUTPUT</b>	<b>Giải thích</b>
20 13 5	26	$20 : 13 = 1,5384615$ <i>Tổng 5 chữ số sau dấu phẩy là: <math>5 + 3 + 8 + 4 + 6 = 26</math></i>
25 3 4	12	$25 : 3 = 8,3333333$ <i>Tổng 4 chữ số sau dấu phẩy là: 12</i>
4 2 6	0	$4 : 2 = 2,000000000$ <i>Tổng 6 chữ số sau dấu phẩy là: 0</i>

Để tính được tổng  $N$  chữ số phần thập phân, với kiểu dữ liệu float, chương trình trả về cho chúng ta khoảng 15 chữ số phần thập phân, khi muốn lấy lớn hơn 15 chữ số phần thập phân, ta cần sử dụng thư viện decimal. Với decimal, ta có thể lựa chọn số lượng chữ số phần thập phân. Để sử dụng thư viện decimal, trước tiên ta cần import thư viện decimal vào dự án. Sau đó, ta lập trình mở 2 file dữ liệu của bài toán.

```
1 from decimal import *
2 file = open("INPUT.INP")
3 file2 = open("OUTPUT.OUT", "w")
```

Tiếp đó, ta đọc dữ liệu từ file dữ liệu vào, tách chuỗi thành danh sách và gán các phần tử trong danh sách vào các giá trị biến  $a$ ,  $b$ ,  $n$  tương ứng.

```
5 data = file.readline()
6 data = data.split()
7 a = int(data[0])
8 b = int(data[1])
9 n = int(data[2])
```

Trường hợp  $a$  chia hết cho  $b$  sẽ không có các chữ số phần thập phân. Chính vì vậy, ta lập trình khi  $a$  chia hết cho  $b$ , sẽ xuất luôn kết quả là 0.

```
11 if a % b == 0:
12     print(0)
13     file2.write("0")
```

Trường hợp khi **a** không chia hết cho **b**, ta tạo biến **sum** lưu tổng các chữ số phần thập phân, biến **sum** ban đầu có giá trị bằng 0. Tiếp đó ta thiết lập số lượng chữ số phần thập phân cần lấy và thực hiện phép tính chia **a** cho **b**. Tuy nhiên khi tính, ta cần ép kiểu Decimal cho hai biến **a** và **b**.

```
14 else:
15     getcontext().prec = n
16     sum = 0
17     temp = Decimal(a)/Decimal(b)
```

Để tách các chữ số phần thập phân với các chữ số phần nguyên, ta chuyển **temp** thành chuỗi và thực hiện tách bằng phương thức **split()** với ký tự phân tách và dấu chấm “.”. Sau khi phân tách, các chữ số phần thập phân cần tính tổng chính là phần tử thứ 2 trong danh sách sau khi phân tách.

```
18     temp = str(temp)
19     temp = temp.split('.')
20     s = temp[1]
```

Tiếp đó, ta sử dụng vòng lặp **for** để tính tổng các chữ số trong **s**. Lưu ý, khi thực hiện phép chia với decimal, các ký tự 0 sẽ không tự động được thêm vào sau nếu số lượng chữ số phần thập phân không đủ theo hàm **getcontext()**. Ví dụ  $5/2 = 2.5$  mà khi ta đặt **getcontext()** là 5 thì kết quả phép chia vẫn chỉ là 2.5. Chính vì vậy khi chạy vòng lặp, ta sẽ lặp tới giá trị độ dài chuỗi **s** mà không lặp tới giá trị **n** của đề bài. Sau khi đã tính được tổng các chữ số, ta ghi kết quả vào tệp tin chứa dữ liệu ra của bài toán.

```
22     for i in range(len(s)):
23         sum += int(s[i])
24     print(sum)
```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```
26 file.close()
27 file2.close()
```

## Bài 2. SX10

*Cho hai số nguyên dương N và K. Ta có X10 của một số là nhân số đó với 10, ví dụ số X10 của 24 là 240; X10 tiếp được số 2400... Số S là tổng của số N và K số X10 liên tiếp của N.*

*Ví dụ  $N = 123$ ,  $k = 3$  ta có  $S = 123 + 1230 + 12300 + 123000 = 136653$ . Hãy in ra tổng các chữ số của S.*

**Dữ liệu:**

*Nhập vào từ file gồm hai số nguyên dương N và K cách nhau 1 dấu cách.*



**Kết quả:**

Gồm một số nguyên duy nhất là kết quả của bài toán.

<b>INPUT</b>	<b>OUTPUT</b>
123 3	24

Trước tiên, ta mở 2 file dữ liệu của bài toán, sau đó tiến hành đọc, tách và gán dữ liệu vào các biến tương ứng. Ta khởi tạo biến **s** lưu giá trị số **s** và biến **temp** lưu giá trị X10 liên tiếp của số **n**. Ban đầu 2 biến **s** và **temp** đều nhận giá trị là **n**.

```

1  file = open("INPUT.INP")
2  file2 = open("OUTPUT.OUT", "w")
3
4  data = file.readline()
5  data = data.split()
6  n = int(data[0])
7  k = int(data[1])
8  s = n
9  temp = n

```

Tiếp đó, ta tính tổng **s** sau **k** lần X10.

```

11 for i in range(k):
12     temp = temp * 10
13     s += temp

```

Tương tự bài toán tính tổng các chữ số phần thập phân, ta tạo biến **sum** lưu tổng các chữ số của **s**, ban đầu **sum** nhận giá trị 0. Sau đó ta sử dụng vòng lặp for tới **len(s)** và cộng các chữ số của **s** vào biến **sum**. Khi đã tính được giá trị **sum**, ta ghi kết quả vào file dữ liệu ra của bài toán.

```

15 s = str(s)
16 sum = 0
17 for i in range(len(s)):
18     sum += int(s[i])
19 print(sum)
20 file2.write(str(sum))

```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```

22 file.close()
23 file2.close()

```

### Bài 3. HCN

Cho  $N$  điểm phân biệt trên hệ trục tọa độ Oxy. Hãy đếm xem có bao nhiêu hình chữ nhật có các cạnh song song với các trục tọa độ mà bốn đỉnh là bốn điểm trong  $N$  điểm đã cho.

**Dữ liệu:**

+ Dòng đầu gồm một số nguyên dương  $N$  là số lượng các điểm.

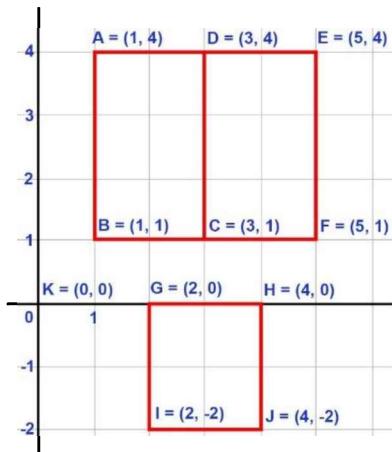
+  $N$  dòng sau, mỗi dòng gồm 2 số  $x, y$  là tọa độ của một điểm.

**Kết quả:**

Gồm một số nguyên dương duy nhất là số lượng hình chữ nhật thỏa mãn đề bài.

Ví dụ

INPUT	OUTPUT	Giải thích
11		Như hình vẽ, có 4 hình chữ nhật là: ABCD, AEFB, CDEF, GHJI.
1 4		
1 1		
3 1		
3 4		
5 4		
5 1		
2 0		
4 0		
2 -2		
4 -2		
0 0		



Trước tiên, ta mở 2 file dữ liệu của bài toán, đọc từ file giá trị  $n$  sau đó tạo danh sách  $x$ , danh sách  $y$  lưu tọa độ x và tọa độ y của các điểm. Tạo biến  $kq$  để lưu kết quả của bài toán.

```

1  file = open("INPUT.INP")
2  file2 = open("OUTPUT.OUT", "w")
3
4  n = int(file.readline())
5  x = []
6  y = []
7  kq = 0

```



Việc kiểm tra tọa độ các đỉnh có thỏa mãn điều kiện là 4 đỉnh của hình chữ nhật hay không, ta gói gọn vào hàm **check(a, b, c, d)**. Hàm **check** trả về giá trị True nếu 4 điểm cần kiểm tra là 4 đỉnh của hình chữ nhật và trả về giá trị False nếu 4 điểm cần kiểm tra không phải 4 đỉnh của hình chữ nhật.

Trong hàm **check**, ta tạo các biến **x1, y1, x2, y2, x3, y3, x4, y4** lưu lần lượt tọa độ 4 đỉnh. Đồng thời đặt giá trị các biến tương ứng tọa độ 4 đỉnh cần kiểm tra.

```
9  def check (a, b, c, d) :
10     x1 = x[a]
11     x2 = x[b]
12     x3 = x[c]
13     x4 = x[d]
14     y1 = y[a]
15     y2 = y[b]
16     y3 = y[c]
17     y4 = y[d]
```

Tiếp theo ta lập trình so sánh tọa độ các đỉnh có thỏa mãn điều kiện là 4 đỉnh của hình chữ nhật hay không. Tuy nhiên ta cũng cần loại trừ trường hợp các đỉnh nhập vào là các điểm có tọa độ giống nhau. Ta cũng cần xét trường hợp các đỉnh có thể bị đảo lộn thứ tự. Tổng cộng có 6 trường hợp có thể xét đến theo thứ tự: 1234, 1243, 1324, 1342, 1423, 1432.

```
19      #Trường hợp 1 và 2
20      if x1 == x2 and y1 != y2:
21          if x3 == x4 and y3 != y4:
22              if (y1 == y3 and y2 == y4) or (y2 == y3 and y1 == y4):
23                  return True
24      #Trường hợp 3 và 4
25      elif x1 == x3 and y1 != y3:
26          if x2 == x4 and y2 != y4:
27              if (y1 == y2 and y3 == y4) or (y2 == y3 and y1 == y4):
28                  return True
29      #Trường hợp 5 và 6
30      elif x1 == x4 and y1 != y4:
31          if x2 == x3 and y2 != y3:
32              if (y1 == y2 and y3 == y4) or (y1 == y3 and y2 == y4):
33                  return True
34      else:
35          return False
```

Trong đoạn chương trình chính, ta lập trình đọc các dữ liệu đầu vào của bài toán là tọa độ của các điểm. Mỗi khi đọc tọa độ một điểm, ta tách tọa độ x và tọa độ y bằng phương thức `split()`, sau đó thêm tọa độ đó vào danh sách **x, y** tương ứng.

```
37 for i in range(n):
38     data = file.readline()
39     data = data.split()
40     x.append(int(data[0]))
41     y.append(int(data[1]))
```

Một hình chữ nhật gồm 4 đỉnh. Trong đó 2 cặp đỉnh thẳng nhau sẽ có cùng tọa độ x hoặc tọa độ y. Do vậy ta thực hiện kiểm tra tọa độ 4 đỉnh cùng lúc. Ta tạo biến **i1**, **i2**, **i3**, **i4** lưu vị trí 4 đỉnh trong danh sách tọa độ. Để tránh kiểm tra lặp lại các đỉnh, ta lập trình **i1** tăng dần từ giá trị 1, **i2** tăng dần từ giá trị **i1 + 1**, **i3** tăng dần từ **i2 + 1** và **i4** tăng dần từ **i3 + 1**. Sau khi lấy được 4 giá trị **i1**, **i2**, **i3**, **i4**, ta kiểm tra tạo độ 4 đỉnh có thỏa mãn là 4 đỉnh của hình chữ nhật hay không bằng cách gọi hàm **check(i1, i2, i3, i4)**. Nếu thỏa mãn là 4 đỉnh của hình chữ nhật, ta tăng **kq** thêm 1.

```
43 for i1 in range(n-3):
44     for i2 in range(i1+1, n-2):
45         for i3 in range(i2+1, n - 1):
46             for i4 in range(i3 + 1, n):
47                 if check(i1, i2, i3, i4):
48                     kq += 1
```

Khi đã tìm được kết quả bài toán, ta ghi kết quả vào file dữ liệu ra của bài toán và đóng 2 file dữ liệu.

```
50 print(kq)
51 file2.write(str(kq))
52
53 file.close()
54 file2.close()
```



## ĐỀ THI THÀNH PHỐ HỒ CHÍ MINH NĂM 2020

### Bài 1. Mẫu số chung nhỏ nhất

Khi thực hiện các phép cộng, trừ dạng phân số, thường học sinh phải thực hiện tìm mẫu số chung nhỏ nhất (MSCNN) rồi mới thực hiện lấy tử số nhân với tích của (MSCNN/Mẫu số).

*Bài toán đặt ra cho chúng ta là xác định MSCNN đó.*

Ví dụ:  $\frac{1}{2} - \frac{2}{3} = \frac{3}{6} - \frac{4}{6} = -\frac{1}{6}$ . Trong bài toán ví dụ này, 6 là MSCNN.

Các phạm vi:

- + Tử và mẫu của các phân số có phạm vi  $10^6$  và khác 0;
- + Số phân số cần tính không quá 100.

**Dữ liệu vào:**

Cho ở tập MSCNN.INP có dạng như sau:

+ Gồm 10 test, mỗi test có 2 dòng, dòng đầu là số nguyên  $n$ , là số phân số. Dòng kế có  $2 \times n$  số nguyên viết lần lượt từng phần của phân số, tử số trước, mẫu số sau, các toán tử không sử dụng. Ví dụ  $\frac{1}{2} - \frac{2}{3}$  được viết là 1 2 2 3.

**Kết quả:**

Ghi vào tập MSCNN.TXT cũng có 10 dòng, mỗi dòng là MSCNN tương ứng.

**Bộ test chuẩn:**

<b>MSCNN.INP</b>	<b>MSCNN.TXT</b>
2	6
1 2 2 3	
3	
2 3 3 5 4 6	
4	
2 4 3 4 5 6 7 8	
5	
1 2 3 4 5 6 7 8 9 10	
6	
2 4 4 5 5 6 10 12 15 19 21 23	
7	

1 3 2 5 3 7 4 9 5 11 6 13 5 17	
8	
2 3 3 13 3 17 4 19 5 23 6 29 7 37 8 41	
9	
1 2 2 4 3 6 4 8 5 10 6 12 7 14 8 16 9 18	
10	
2 3 3 6 4 9 5 12 6 15 7 18 8 21 8 24 10 27 8 30	
15	
17 3 21 5 33 7 11 2 123 17 1 37 2 41 5 53 2 67 1 23 5 3 7 8 101 103 12 37 9 103	

Để giải bài toán, ta áp dụng công thức tính bội chung nhỏ nhất (BCNN) dựa vào ước chung lớn nhất (UCLN) của hai số.

$$\text{BCNN}(a, b) = \frac{a * b}{\text{UCLN}(a, b)}$$

Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1  file = open("MSCNN.INP")
2  file2 = open("MSCNN.TXT", "w")
```

Tiếp đó, ta định nghĩa hàm **UCLN(a, b)** là hàm tìm ước chung lớn nhất của hai số **a** và **b**, với **a**, **b** là 2 tham số được truyền vào.

Để tìm ước chung lớn nhất của 2 số, ta đặt biến **i** là số có giá trị nhỏ hơn trong 2 số **a** và **b**. Tiếp đó giảm **i** tới khi cả **a** và **b** đều chia hết cho **i**, khi đó **i** chính là ước chung lớn nhất của hai số.

```
4  def UCLN (a, b):
5      if a > b:
6          temp = a
7          a = b
8          b = temp
9      i = a
10     while True:
11         if a % i == 0 and b % i == 0:
12             return i
13         i -= 1
```

Sau khi định nghĩa hàm tìm ước chung lớn nhất, ta đọc dữ liệu vào của bài toán, tổng có 10 test, vậy ta sử dụng vòng lặp for để đọc dữ liệu từng test và tìm BCNN của từng test.



```

15 for test in range (10):
16     n = int(file.readline())
17     temp = file.readline()
18     data = temp.split()

```

Ban đầu, ta đặt mẫu số chung nhỏ nhất (MSCNN) chính là mẫu số của phân số đầu tiên. Sau đó tìm MSCNN của MSCNN hiện tại với mẫu số của phân số tiếp theo trong danh sách.

```

19     MSCNN = int(data[1])
20     i = 1
21     while i < n*2:
22         temp = int(data[i])
23         MSCNN = int(MSCNN * temp/UCLN(MSCNN, temp))
24         i += 2

```

Sau khi tìm được MSCNN, ta in kết quả lên màn hình để kiểm tra kết quả và ghi kết quả vào file dữ liệu ra của bài toán.

```

25     print(MSCNN)
26     file2.write(str(MSCNN))
27     if test < 9:
28         file2.write('\n')

```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```

30 file.close()
31 file2.close()

```

## Bài 2. Chữ số

Tìm  $K$  chữ số cuối cùng của  $M^N$  ( $0 < K \leq 9$ ;  $0 \leq M, N \leq 10^6$ ).

**Dữ liệu vào:**

Từ tập tin CHUSO.INP gồm 10 dòng, mỗi dòng ghi 3 số  $M, N, K$ .

**Kết quả:**

Ghi ra tập tin CHUSO.TXT gồm 10 dòng, mỗi dòng ghi một số nguyên là  $K$  chữ số cuối cùng của  $M^N$  tương ứng với mỗi dòng dữ liệu vào.

**Bộ test chuẩn:**

CHUSO.INP	CHUSO.TXT
12 3 1	8
23 4 2	...
345 6 2	
567 89 3	

6789 101 4 78912 3456 5 891234 5678 6 123456 78912 7 234567 891234 8 345678 991122 9	
---	--

Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1  file = open("CHUSO.INP")
2  file2 = open("CHUSO.TXT", "w")
```

Trong bài có 10 test, vậy ta sử dụng vòng lặp for và đọc dữ liệu từ file CHUSO.INP. Sau khi đọc 1 dòng dữ liệu, ta tách các số thành danh sách và gán giá trị vào các biến **m**, **n**, **k** tương ứng ba giá trị M, N và K của bài toán.

```
4  for test in range(10):
5      data = file.readline()
6      data = data.split()
7      m = int(data[0])
8      n = int(data[1])
9      k = int(data[2])
```

Ta tạo biến **kq** lưu kết quả của bài toán. Ban đầu, ta đặt **kq** bằng **m** ban đầu và lặp lại **n** lần việc lũy thừa. Tuy nhiên ta chỉ cần lấy **k** ký tự trong kết quả, vậy sau khi nhân, ta cắt lấy **k** ký tự cuối của biến **kq**. Để cắt **k** ký tự ta cần chuyển **kq** thành chuỗi, sau đó đổi lại kiểu dữ liệu biến **kq** thành int sau khi cắt **k** ký tự cuối cùng.

```
11     kq = m
12     for i in range(n-1):
13         kq *= m
14         temp = str(kq)
15         if len(temp) > k:
16             kq = temp[len(temp)-k:]
17             kq = int(kq)
```

Khi đã tính xong kết quả, ta ghi dữ liệu vào file dữ liệu ra của bài toán.

```
18     print(kq)
19     if test < 9:
20         file2.write(str(kq) + '\n')
21     else:
22         file2.write(str(kq))
```



Cuối cùng, ta đóng 2 file dữ liệu của bài toán sau khi đã ghi xong kết quả.

```
24 file.close()
25 file2.close()
```

### Bài 3. Đổi hướng

Trên sân trường, các bạn học sinh đang đứng với hướng nhìn khác nhau so với bạn Nam. Các bạn nhìn về phía Nam ta gọi là nhìn về trước; các bạn quay lưng lại với Nam ta gọi là nhìn về sau; các bạn nhìn về phía bên trái Nam ta gọi là nhìn sang trái; các bạn nhìn về phía bên phải Nam ta gọi là nhìn sang phải. Nam muốn hô để điều động các bạn xoay về cùng một hướng nhưng Nam không muốn các bạn phải xoay nhiều (mỗi lần xoay các bạn chỉ xoay một góc  $90^\circ$  nên nếu bạn đang quay lưng lại với Nam muốn nhìn về phía Nam, bạn cần phải xoay 2 lần).

**Yêu cầu:** Em hãy giúp Nam tìm hướng nào nên chọn để các bạn trên sân chỉ phải xoay với số lần ít nhất là có thể cùng nhìn về hướng ấy.

**Dữ liệu vào:** Trong tệp tin văn bản DOIHUONG.INP ghi 10 test trên 10 dòng, mỗi dòng ghi một dãy các chữ số viết liền với nhau, dãy số chỉ gồm các chữ số 1, 2, 3, 4 mô tả hướng nhìn của từng bạn trên sân theo quy tắc: 1 – nhìn về trước; 2 – nhìn về sau; 3 – nhìn sang trái; 4 – nhìn sang phải.

**Kết quả:** Ghi ra tệp văn bản DOIHUONG.TXT với kết quả của mỗi test ghi trên 2 dòng theo mô tả sau:

- + Dòng thứ nhất ghi số tương ứng với hướng mà Nam cần chọn để hô điều động. Nếu có nhiều hơn một hướng có thể chọn thì ghi lần lượt từng số tương ứng với các hướng theo thứ tự từ nhỏ đến lớn (trước – sau – trái – phải), mỗi số cách nhau ít nhất một khoảng trắng.

- + Dòng thứ hai ghi số lần cần xoay ít nhất để các bạn trên sân cùng nhìn về hướng cần điều động.

**Ví dụ:**

DOIHUONG.INP	DOIHUONG.TXT	Giải thích
1143322	3 6	Có 7 bạn trên sân với hướng nhìn lần lượt là: 2 bạn nhìn trước, một bạn nhìn phải, 2 bạn nhìn trái và 2 bạn nhìn sau. <b>→</b> Nam nên chọn hướng điều động là nhìn về bên trái (3) thì các bạn cần xoay chỉ phải xoay với số lần ít nhất là 6.
21213414	1 4 7	

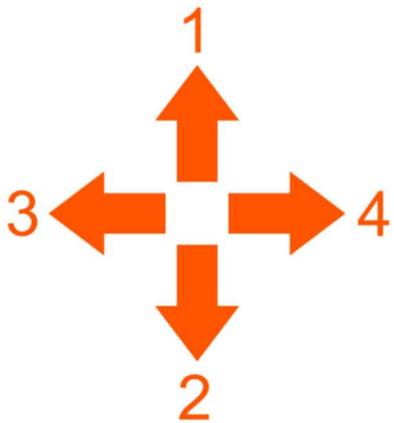
**Bộ test chuẩn:**

DOIHUONG.INP	DOIHUONG.TXT
2311124334	
1334232142141423	
214314341433141431431414143	
433233413232422142321234221113	
32332142142414232141421342321421414133	
314322133432213221441122322441444423142	
113322422344122441241324111123311223312144	
2144431411334224323141433344344433241222242244142	
133412313342321334232142141234214142321421414213342	
41231334212332133423214214123421414232142141421334	

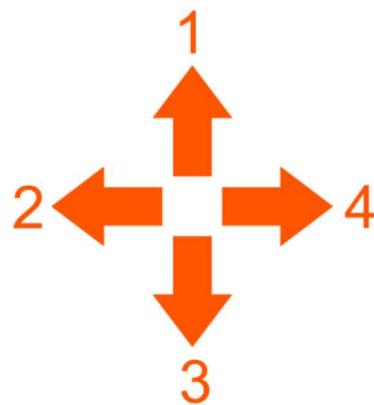
Trước khi lập trình, ta quan sát các hướng theo đề bài mô tả trong Hình 1 bên dưới. Giả sử hướng lên trên là hướng về phía trước. Khi muốn tính số lần xoay với các hướng như vậy, ta cần kiểm tra từng hướng các bạn học sinh đang đứng và từng hướng mà bạn Nam cần hô.

Ví dụ nếu Nam muốn hô xoay về hướng 1, ta cần kiểm tra nếu hướng hiện tại của bạn học sinh là 3 thì số lần xoay là 1, nếu hướng là 2, số lần xoay là 2, nếu hướng là 4 số lần xoay là 1. Để tính được số lần xoay ta cần kiểm tra từng hướng.

Để thuận tiện hơn khi tính toán, ta có thể đổi lại hướng số 2 và số 3 như Hình 2, khi đó số lần xoay chính là |hướng Nam cần hô – hướng hiện tại| và nếu số lần xoay tính được = 3, ta đặt lại số lần xoay thành 1. Vì khi ở hướng 4 cần xoay về hướng 1, thực tế ta chỉ cần xoay 1 lần.



Hình 1



Hình 2



Đầu tiên, lập trình mở 2 file dữ liệu của bài toán.

```
1  file = open("DOIHUONG.INP")
2  file2 = open("DOIHUONG.TXT", "w")
```

Ta tạo danh sách **kq** lưu danh sách các hướng có số lần hô ít nhất. Sau đó thực hiện lặp lại 10 lần việc đọc dữ liệu từ file DOIHUONG.INP.

```
4  kq = []
5  for test in range(10):
6      s = file.readline()
```

Với mỗi dòng dữ liệu vào, trước khi tính toán số lần đổi hướng, ta tạo danh sách **s1** gồm các ký tự của **s** sau đó chuyển đổi các ký tự '2' trong **s1** thành '3' và ngược lại, đổi các ký tự '3' thành '2'.

```
7  s1 = list(s)
8  kq.clear()
9  for i in range(len(s)):
10     if s1[i] == '2':
11         s1[i] = '3'
12     elif s1[i] == '3':
13         s1[i] = '2'
```

**Lưu ý:** Chuỗi trong Python là bất biến (không thay đổi được), vì vậy khi muốn thay đổi ký tự trong chuỗi, ta nên tạo một bản sao dưới dạng danh sách, sau đó thay đổi giá trị trong danh sách bản sao đó.

Tiếp đó, ta tính số lần xoay với từng hướng từ 1 tới 4 xem hướng nào có số lần xoay ít nhất. Ta sử dụng vòng lặp for với hướng bắt đầu từ 1. Ta tạo biến **dem** lưu số lần cần xoay của tất cả các bạn trong hàng. Ban đầu ta đặt **dem** là 0, sau đó duyệt từng phần tử trong danh sách **s1** tính số lần xoay và cộng vào biến **dem**.

```
15  for huong in range(1, 5):
16      dem = 0
17      for i in range(len(s)-1):
18          temp = abs(huong - int(s1[i]))
19          if temp > 2:
20              temp = 1
21          dem += temp
```

Ta bắt đầu tính hướng từ 1, vậy sau khi tính được tổng số lần xoay của các bạn trong hàng, ta kiểm tra nếu hướng bằng 1, ta đặt kết quả là hướng và số lần xoay là **dem**. Trường hợp còn lại, khi hướng bằng 2, 3 hoặc 4, ta so sánh kết quả đã lưu trước đó nếu lớn hơn số lần xoay hiện tại, ta đặt lại kết quả của bài toán. Nếu kết quả đã lưu bằng với số lần xoay hiện tại, ta thêm hướng hiện tại vào danh sách các hướng đã lưu.

```

33     if huong == 1:
34         tong = dem
35         kq.append(huong)
36     elif tong > dem:
37         kq.clear()
38         tong = dem
39         kq.append(huong)
40     elif tong == dem:
41         kq.append(huong)

```

Sau khi tìm được tổng số lần xoay ít nhất và hướng xoay phù hợp, ta cần đổi trả lại hướng 2 thành 3 và hướng 3 thành 2 nếu có trong danh sách hướng có số lần xoay ít nhất.

```

33     for i in range(len(kq)):
34         if kq[i] == 2:
35             kq[i] = 3
36         elif kq[i] == 3:
37             kq[i] = 2

```

Tiếp đó, ta ghi kết quả vào file dữ liệu ra của bài toán. Tuy nhiên trong test cuối cùng ta không ghi ký tự xuống dòng. Ngoài ra, các bạn có thể kiểm tra kết quả bằng cách dùng lệnh print() để in kết quả trực tiếp lên cửa sổ Run.

```

39     temp = ' '.join([str(i) for i in kq])
40     if test < 9:
41         file2.write(temp + '\n' + str(tong) + '\n')
42     else:
43         file2.write(temp + '\n' + str(tong))
44
45     print(temp)
46     print(tong)

```

Cuối cùng, ta đóng hai file dữ liệu của bài toán sau khi đã hoàn tất ghi kết quả vào file dữ liệu ra.

```

48 file.close()
49 file2.close()

```



## ĐỀ THI THÀNH PHỐ ĐÀ NẴNG NĂM 2021

### Bài 1. Số giàu có

Trong các số tự nhiên từ 1 đến  $N$ , số tự nhiên được gọi là giàu có nhất nếu nó có tổng các ước lớn nhất trong các số này.

Ví dụ: Số 12 là số giàu có nhất trong các số tự nhiên từ 1 đến 15. Tổng các ước của 12 là  $1 + 2 + 3 + 4 + 6 + 12 = 28$ .

**Yêu cầu:** Hãy xác định số giàu có nhất trong các số tự nhiên từ 1 đến  $N$ .

**Dữ liệu vào:** Nhập từ bàn phím một số tự nhiên  $N$  ( $0 < N < 10^6$ ).

**Dữ liệu ra:** In ra màn hình số giàu có nhất trong các số từ 1 đến  $N$ .

Chú ý: Nếu kết quả có nhiều hơn một số thì in ra số nhỏ nhất trong các số đó.

Ví dụ	Nhập từ bàn phím	In ra màn hình
	15	12

**Phân tích:** Để tính tổng các ước của một số (ví dụ a), trước tiên ta cần tìm các ước của a. Để tìm ước, ta sử dụng vòng lặp for, lấy giá trị i từ 1 tới a. Nếu a chia hết cho i, ta cộng i vào tổng ước của a. Tuy nhiên, ta có thể giảm số lần lặp từ 1 tới  $\sqrt{a}$ . Bởi vì thực tế nếu  $a \times b = n$ , thì khi đó n chia hết cho cả a và b. Vậy khi kiểm tra a chia hết cho i, ngoài việc cộng thêm i vào tổng, ta sẽ cộng thêm  $a/i$  vào tổng các ước.

Trong bài toán, ta cần sử dụng phép căn bậc hai, vậy ta import thư viện math vào dự án, tiếp đó yêu cầu nhập dữ liệu vào của bài toán từ cửa sổ Run và thiết lập giá trị ban đầu cho biến **max** – biến lưu tổng lớn nhất của các ước.

```

1 from math import *
2
3 n = input("Nhập vào số N: ")
4 n = int(n)
5 max = 0

```

Tiếp theo, ta tạo hàm **TongUoc()** với tham số truyền vào a để tính tổng các ước của a. Trong hàm ta kiểm tra nếu  $a = 1$ , hàm sẽ trả về tổng ước là 1, còn không thì tính tổng các ước từ 1 tới  $\sqrt{a}$ , sau đó trả về tổng các ước đã tính được.

Sau khi tính được các ước từ 1 tới  $\sqrt{a}$ , ta kiểm tra  $\sqrt{a}$  có phải số nguyên hay không bằng cách lấy bình phương phần nguyên của  $\sqrt{a}$ , nếu bình phương đó bằng a, ta cộng thêm giá trị  $\sqrt{a}$  và tổng.

```

7 def TongUoc(a):
8     if a == 1:
9         return 1
10    sum = 0
11    temp = int(sqrt(a))
12    for i in range(1, temp):
13        if a % i == 0:
14            sum += i + a/i
15    if temp**2 == a:
16        sum += temp
17    return sum

```

Trong đoạn chương trình chính, ta dùng vòng lặp for chạy từ 1 tới **n** + 1. Với mỗi giá trị **i**, ta tính tổng các ước của , sau đó so sánh với giá trị **max**. Nếu **max** nhỏ hơn tổng vừa tính, ta đổi lại **max** thành tổng và đặt kết quả là **i**. Cuối cùng, ta in kết quả lên màn hình sau khi đã tìm được số giàu có.

```

19 for i in range(1, n+1):
20     temp = TongUoc(i)
21     if max < temp:
22         max = temp
23         kq = i
24
25 print(kq)

```

## Bài 2. Dịch cúm

Như chúng ta đã biết dịch cúm toàn cầu COVID-19 do virus Corona nhân bản và lây lan gây hội chứng suy hô hấp cấp tính nặng ở người. Người bệnh ban đầu không nhận biết được đã nhiễm bệnh do virus còn tiềm ẩn chưa khởi phát. Ở đâu đó những con virus đang ẩn mình, chúng ta cùng tìm chúng nhé!

Cho một xâu ký tự C, O, R, N, A ở vị trí bất kì. Ta có thể thực hiện hoán đổi các ký tự này để tạo thành những cụm từ CORONA liên tiếp, mỗi cụm từ CORONA tương ứng với một con virus.

Ví dụ: Với xâu ký tự S = “COOCROONRANNA”, sau khi thực hiện hoán đổi các ký tự của xâu S ta được xâu “CORONACORONAN” có hai cụm từ CORONA tương ứng với hai con virus.

**Yêu cầu:** Hãy xác định số lượng con virus Corona sau khi thực hiện hoán đổi các ký tự trong xâu S theo yêu cầu như trên.

**Dữ liệu vào:** Đọc từ file văn bản có tên CORONA.INP một xâu S chỉ chứa các ký tự C, O, R, N, A và có độ dài  $L$  ( $0 < L \leq 255$ ).

**Dữ liệu ra:** Ghi ra file văn bản có tên CORONA.OUT một số nguyên là số lượng con virus Corona tạo ra sau khi hoán đổi các ký tự trong xâu S.



Ví dụ

CORONA.INP	CORONA.OUT
COOCROONRANNA	2

**Phân tích:** Trong bài ta cần tìm số lượng các từ CORONA có thể ghép được từ một xâu cho trước. Như ta thấy, trong cụm từ CORONA sẽ cần 1 chữ C, 2 chữ O, 1 chữ R, 1 chữ N và 1 chữ A. Vậy thay vì sắp xếp, đảo vị trí các chữ, ta tiến hành đếm số lượng các chữ cái trong xâu cho trước. Sau đó tính số lượng virus dựa theo số lượng từ đã đếm.

Trước tiên, ta mở 2 file dữ liệu vào, dữ liệu ra và đọc xâu ký tự từ file dữ liệu vào của bài toán.

```
1  file = open("CORONA.INP")
2  file2 = open("CORONA.OUT", "w")
3
4  s = file.readline()
```

Tiếp đó ta lập trình đếm các chữ cái trong xâu **s**. Để đếm chữ cái, ta sử dụng phương thức `count()`.

```
6  c = s.count("C")
7  o = s.count("O")
8  r = s.count("R")
9  n = s.count("N")
10 a = s.count("A")
```

Trong cụm từ CORONA, số lượng chữ C, R, N và A bằng nhau, vậy ta tạo một biến **temp** lưu giá trị nhỏ nhất trong số lượng chữ C, R, N và A.

Số lượng chữ O trong cụm từ là 2, vậy ta kiểm tra nếu **temp** \* 2 nhỏ hơn hoặc bằng số lượng chữ O thì ta đặt kết quả là **temp**, còn không thì đặt kết quả là số lượng chữ O chia cho 2.

```
12 temp = min(c, r, n, a)
13 if 2*temp <= o:
14     kq = temp
15 else:
16     kq = int(o/2)
```

Cuối cùng, ta ghi kết quả vào file dữ liệu ra của bài toán và đóng 2 file dữ liệu. Các bạn có thể thêm lệnh `print(kq)` để kiểm tra trực tiếp kết quả bài toán trên cửa sổ Run của PyCharm.

```
18 print(kq)
19 file2.write(str(kq))
20
21 file.close()
22 file2.close()
```

### Bài 3. Cắt dây

Tý muốn cắt một sợi dây có chiều dài  $N$  (mét) thành 3 đoạn dây có chiều dài mỗi đoạn là số nguyên dương (đơn vị mét) sao cho 3 đoạn dây này là 3 cạnh của một tam giác cân có cạnh đáy lớn hơn cạnh bên.

**Lưu ý:** Tam giác cân là tam giác có hai cạnh bằng nhau, hai cạnh bằng nhau gọi là hai cạnh bên, cạnh còn lại là cạnh đáy.

**Yêu cầu:** Em hãy giúp Tý tính có bao nhiêu cách cắt đoạn dây này.

**Dữ liệu vào:** Đọc từ file văn bản CATDAY.INP số nguyên dương  $N$ .

**Dữ liệu ra:** Ghi ra file văn bản CATDAY.OUT số  $M$  là số cách cắt sợi dây theo yêu cầu.

Ví dụ	CATDAY.INP	CATDAY.OUT
	19	2

*Giải thích:*

Có 2 cách cắt sợi dây thành 3 đoạn thỏa mãn đề bài là:  $(5m, 5m, 9m)$  và  $(6m, 6m, 7m)$ .

**Lưu ý:**

+ Các cách cắt sợi dây thành 3 đoạn  $(x$  mét,  $x$  mét,  $y$  mét) và các hoán vị của bộ 3 số  $(x, x, y)$  chỉ được tính là 1 cách cắt. **Chẳng hạn:** Cách cắt thành các đoạn  $(5m, 5m, 9m)$  và các hoán vị của nó là  $(5m, 9m, 5m)$  hoặc  $(9m, 5m, 5m)$  chỉ được tính là một cách cắt.

Điều kiện để 3 đoạn thẳng có thể ghép thành một hình tam giác là tổng 2 cạnh sẽ lớn hơn cạnh còn lại. Đề bài yêu cầu cắt đoạn dây thành 3 đoạn để ghép thành tam giác cân có 2 cạnh bằng nhau và độ dài cạnh đáy lớn hơn 2 cạnh bên. Vậy ban đầu ta chia 3 sợi dây, sau đó giảm độ dài cạnh bên đi. Ta giảm độ dài tới khi tổng độ dài 2 cạnh bên không lớn hơn cạnh đáy (không thỏa mãn là 3 cạnh của tam giác) thì dừng vòng lặp.

Trước tiên, ta mở 2 file dữ liệu của bài toán và đọc dữ liệu vào. Ta tạo biến **a** lưu độ dài 2 cạnh bên, biến **b** lưu độ dài cạnh đáy, biến **dem** lưu số cách có thể cắt sợi dây thỏa mãn yêu cầu đề bài.

```

1  file = open("CATDAY.INP")
2  file2 = open("CATDAY.OUT", "w")
3
4  n = int(file.readline())
5  a = int(n/3)
6  b = n - 2*a
7  dem = 0

```



Tiếp đó ta lặp lại việc giảm **a** và tính độ dài cạnh đáy **b** là độ dài còn lại của sợi dây sau khi cắt 2 cạnh **a**. Với mỗi cặp **a**, **b**, ta kiểm tra nếu **a** nhỏ hơn **b** và  $2 * a$  lớn hơn **b**, ta tăng **đếm** thêm 1.

```

9  while (2*a > b):
10     if (a < b and 2*a > b):
11         dem += 1
12
13     a -= 1
14     b = n - 2 * a

```

Cuối cùng, ta ghi kết quả vào file dữ liệu ra của bài toán và đóng 2 file dữ liệu.

```

16 print(dem)
17 file2.write(str(dem))
18
19 file.close()
20 file2.close()

```

#### Bài 4. Sắp xếp theo modul K

Từ dãy số tự nhiên 1; 2; 3; ...;  $N$  người ta sắp xếp lại dãy số này theo số dư trong các phép chia các số hạng của dãy số cho một số tự nhiên  $K$  là ước nào đó của  $N$  như sau:

- + Đoạn thứ nhất gồm tất cả các số chia hết cho  $K$ .
- + Đoạn thứ hai gồm tất cả các số chia  $K$  dư 1.
- + Đoạn thứ ba gồm tất cả các số chia  $K$  dư 2.
- + ...
- + Đoạn cuối cùng gồm tất cả các số chia  $K$  dư  $K - 1$ .

Các số hạng trong mỗi đoạn cũng được sắp xếp theo chiều tăng dần.

Ví dụ: Với  $N = 12$  và  $K = 4$  sau khi sắp xếp ta có dãy số sau: 4; 8; 12; 1; 5; 9; 2; 6; 10; 3; 7; 11

Cho trước 3 số nguyên dương  $N$ ;  $K$ ;  $M$  (với  $K$  là ước của  $N$  và  $M \leq N$ ).

**Yêu cầu:** Tìm số hạng thứ  $M$  của dãy đã sắp xếp.

**Dữ liệu vào:** Đọc từ file văn bản MODK.INP 3 số nguyên dương  $N$ ,  $K$ ,  $M$  trên cùng một dòng, mỗi số cách nhau một dấu cách.  $K$  là ước của  $N$ ;  $M \leq N$ .

**Dữ liệu ra:** Ghi ra file văn bản MODK.OUT số hạng thứ  $M$  của dãy số.

Ví dụ	MODK.INP	MODK.OUT
	12 4 6	9

**Phân tích:** Để giải bài toán, ta tạo biến **du** lưu giá trị dư của phép chia. Ban đầu ta đặt **du** bằng 0 và tăng dần lên. Với mỗi giá trị **du**, ta tìm các số trong khoảng từ 1 tới **N** sao cho số đó chia cho K dư **du**. Tuy nhiên ta có thể nhân K lần lượt với các số tăng dần từ 0, 1, 2,... và cộng với giá trị **du**, khi đó ta sẽ tìm được trực tiếp các số chia cho K dư **du**.

Trước tiên, ta mở 2 file dữ liệu của bài toán.

```
1 file = open("MODK.INP")
2 file2 = open("MODK.OUT", "w")
```

Tiếp đó, ta tiến hành đọc dữ liệu vào của bài toán và lưu vào các biến tương ứng theo yêu cầu của đề bài.

```
3 data = file.readline()
4 data = data.split()
5 n = int(data[0])
6 k = int(data[1])
7 m = int(data[2])
```

Ta tạo danh sách **kq** lưu dãy số được sắp xếp theo số dư, tạo biến **du** lưu số dư của phép chia và biến **i** lưu vị trí số tìm được trong dãy số.

```
9 kq = []
10 du = 0
11 i = 0
```

Tiếp theo, ta tiến hành tìm và thêm các số vào danh sách **kq** theo yêu cầu các số sắp xếp tăng dần theo giá trị số dư của phép chia cho **k**.

```
13 while i <= m:
14     num = 0
15     while (num*k + du) <= n:
16         temp = num*k + du
17         kq.append(temp)
18         i += 1
19         num += 1
20     if i > m:
21         break
22     du += 1
```

Khi đã tìm được giá trị tại vị trí thứ **m** trong dãy số, ta ghi kết quả vào tệp dữ liệu ra của bài toán. Kết quả chính là số cuối cùng trong danh sách.

```
24 print(kq[-1])
25 file2.write(str(kq[-1]))
```

Cuối cùng, ta đóng 2 file dữ liệu của bài toán.

```
27 file.close()
28 file2.close()
```



## ĐỀ THI TỈNH NGHỆ AN NĂM 2020

### Bài 1. Số chính phương

An vừa được học về số chính phương, An hiểu rằng một số nguyên là số chính phương nếu số đó là bình phương của một số nguyên khác. Ví dụ các số: 1, 4, 9, 16, ... là số chính phương. Các số 2, 3, 5, 6, 7, 8, 10, ... không là số chính phương. Khi cho một số, An dễ dàng xác định số đó có là số chính phương hay không. Hôm nay, thầy giáo cho An một bài toán khó hơn như sau: "Cho 2 số nguyên dương  $L, R$  ( $L \leq R$ ). Hãy xác định số lượng số chính phương trong đoạn  $[L, R]$ ".

**Yêu cầu:** Hãy lập trình giúp An giải bài toán trên.

**Dữ liệu:** Vào từ file **SQUARE.INP** 2 số nguyên dương  $L, R$ .

**Dữ liệu ra:** Ghi ra file **SQUARE.OUT** một số nguyên duy nhất là số lượng số chính phương trong đoạn  $[L, R]$  tìm được.

Ví dụ	<b>SQUARE.INP</b>	<b>SQUARE.OUT</b>
	2 16	3

**Giải thích:** Có 3 số chính phương thỏa mãn là 4, 9, 16.

Trước tiên, ta import thư viện math, mở 2 file dữ liệu của bài toán và tiến hành đọc dữ liệu đầu vào, sau đó lưu vào các biến tương ứng.

```

1 from math import *
2
3 file = open("SQUARE.INP")
4 file2 = open("SQUARE.OUT", "w")
5
6 data = file.readline()
7 data = data.split()
8 L = int(data[0])
9 R = int(data[1])

```

Ta tạo biến **num**, đặt **num** thành phần nguyên của phép căn bậc hai của **L**. Khi đó để tìm các số chính phương trong đoạn **[L, R]**, ta thực hiện việc tăng **num** thêm 1 và bình phương **num**. Nếu giá trị bình phương đó vẫn nằm trong đoạn **[L, R]** ta tăng số lượng các số chính phương thêm 1 và tăng **num** thêm 1.

Tuy nhiên, trước khi tăng **num**, ta cần kiểm tra giá trị **L** có phải số chính phương hay không.

```

11 num = int(sqrt(L))
12
13 if num**2 == L:
14     dem = 1
15 else:
16     dem = 0
17 num += 1
18
19 while num**2 <= R:
20     num += 1
21     dem += 1

```

Cuối cùng, ta ghi kết quả vào file “SQUARE.OUT” và đóng 2 file dữ liệu của bài toán.

```

23 print(dem)
24 file2.write(str(dem))
25
26 file.close()
27 file2.close()

```

## Bài 2. Hình chữ nhật

Cho  $n$  điểm tọa độ nguyên trên mặt phẳng. Điểm thứ  $i$  ( $1 \leq i \leq n$ ) có tọa độ  $(x_i, y_i)$ . Hãy xác định hình chữ nhật có diện tích nhỏ nhất chứa toàn bộ tất cả  $n$  điểm đã cho. 4 đỉnh hình chữ nhật phải có tọa độ nguyên, các cạnh song song với trục tọa độ và tất cả các điểm phải nằm hoàn toàn bên trong hình chữ nhật đó.

**Dữ liệu:** Nhập vào từ bàn phím gồm:

- + Số nguyên  $n$ .
- +  $n$  dòng tiếp, dòng thứ  $i$  chứa 2 số nguyên  $x_i$  và  $y_i$  xác định tọa độ điểm thứ  $i$ .

**Kết quả:** Ghi ra màn hình số nguyên duy nhất là diện tích hình chữ nhật tìm được.

INPUT	OUTPUT	Giải thích
<pre> 4 -1 0 0 1 -1 -1 2 0 </pre>	20	



**Phân tích:** Để giải bài toán, ta cần tìm giá trị tọa độ x lớn nhất ( $x_{max}$ ), x nhỏ nhất ( $x_{min}$ ), y lớn nhất ( $y_{max}$ ) và y nhỏ nhất ( $y_{min}$ ) của các điểm. Khi đã tìm được 4 giá trị, ta lấy giá trị lớn nhất trừ giá trị nhỏ nhất cộng với 2 sẽ được cạnh của hình chữ nhật. Vì theo yêu cầu của đề bài, các điểm phải nằm hoàn toàn bên trong hình chữ nhật, chính vì vậy ta cần cộng thêm 2.

Trước tiên, ta yêu cầu người dùng nhập vào giá trị  $n$  từ bàn phím, sau đó sử dụng vòng lặp for và yêu cầu nhập vào tọa độ của  $n$  điểm.

```

1 n = input("Nhập vào số nguyên N: ")
2 n = int(n)
3
4 for i in range(n):
5     data = input("Nhập vào tọa độ điểm thứ " + str(i+1) + ": ")
6     data = data.split()
7     x = int(data[0])
8     y = int(data[1])

```

Với mỗi cặp tọa độ nhập vào, ta kiểm tra và so sánh và tìm các giá trị lớn nhất, nhỏ nhất của  $x$  và  $y$ .

```

10     if i == 0:
11         xmin = x
12         xmax = x
13         ymin = y
14         ymax = y
15     else:
16         if xmin > x:
17             xmin = x
18         if xmax < x:
19             xmax = x
20         if ymin > y:
21             ymin = y
22         if ymax < y:
23             ymax = y

```

Cuối cùng, ta tính độ dài 2 cạnh của hình chữ nhật và in kết quả của bài toán là diện tích của hình chữ nhật bao quanh các điểm.

```

25 a = xmax - xmin + 2
26 b = ymax - ymin + 2
27
28 print (a*b)

```

### Bài 3. Điểm danh

Hội thi tin học trẻ năm nay có sự tham gia của  $N$  học sinh đến từ khắp mọi nơi trên địa bàn tỉnh Nghệ An. Một điều đặc biệt hơn nữa là tên của tất cả các em có thể biểu diễn bởi các xâu ký tự không dấu đói một khác nhau và có độ dài không quá 10 ký tự.

Ngay khi đến với hội thi, Ban tổ chức tiến hành phát phiếu ghi danh và đã thu về tổng cộng  $N - 1$  phiếu với  $N - 1$  tên khác nhau.

**Yêu cầu:** Cho biết danh sách các tên của tất cả  $N$  học sinh,  $N - 1$  tên học sinh đã ghi trong phiếu và các em đều ghi đúng tên của mình. Hãy xác định tên học sinh chưa tham gia ghi danh tại hội thi.

**Dữ liệu:**

- + Dòng đầu tiên chứa số nguyên dương  $N$ .
- +  $N$  dòng tiếp theo, mỗi dòng là một xâu ký tự xác định tên của từng em học sinh đăng ký ban đầu.
- +  $N - 1$  dòng cuối cùng, mỗi dòng là một xâu ký tự xác định tên của từng em học sinh đã ghi danh.

**Kết quả:**

Ghi ra màn hình tên của em học sinh chưa tham gia ghi danh tại hội thi.

**Chú ý:** Các tên đều cho dưới dạng xâu ký tự la tinh in hoa và không có dấu cách.

<b>INPUT</b>	<b>OUTPUT</b>
3 AN VIET TUNG AN TUNG	VIET

**Phân tích:** Để giải bài toán, ta tạo danh sách **Điểm danh**, danh sách này ban đầu ta thêm toàn bộ các giá trị là 0 tương ứng chưa có mặt. Khi duyệt các bạn đã ghi danh tại hội thi, ta đổi giá trị tại vị trí của bạn đó thành 1. Cuối cùng, giá trị 0 còn lại chính là bạn chưa ghi danh.

Trước tiên, ta mở 2 file dữ liệu của bài toán và đọc giá trị **n** tương ứng số học sinh đăng ký ban đầu.

```

1  file = open("INPUT.INP")
2  file2 = open("OUTPUT.OUT", "w")
3
4  data = file.readline()
5  n = int(data)

```

Ta tạo danh sách **name** để lưu tên các thí sinh đăng ký ban đầu và danh sách **DiemDanh** để lưu trạng thái ghi danh của các thí sinh.



```
6 name = []
7 DiemDanh = []
```

Tiếp đó ta đọc dữ liệu vào của bài toán là danh sách tên thí sinh đã đăng ký tham gia thi ban đầu. Tuy nhiên khi đọc file, Python sẽ đọc cả ký tự xuống dòng “\n”. Chính vì vậy, khi lưu tên học sinh, ta sẽ bỏ ký tự cuối cùng đi. Với mỗi học sinh, ta thêm trạng thái điểm danh ban đầu của thí sinh đó là 0 và danh sách **DiemDanh**.

```
9 for i in range(n):
10     data = file.readline()
11     name.append(data[0:-1])
12     DiemDanh.append(0)
```

Khi đã đọc được danh sách các thí sinh đăng ký tham gia dự thi ban đầu, ta tiến hành đọc dữ liệu gồm  $n - 1$  thí sinh đã ghi danh. Với mỗi thí sinh ta kiểm tra thứ tự ghi danh của thí sinh đó trong danh sách **name** ban đầu và chuyển trạng thái bên danh sách **DiemDanh** tại vị trí đó thành 1, nghĩa là thí sinh này đã điểm danh. Tuy nhiên chúng ta cần lưu ý trường hợp thí sinh cuối cùng sẽ không có ký tự xuống dòng.

```
14 for i in range(n-1):
15     data = file.readline()
16     if i < n-2:
17         s = data[0:-1]
18     else:
19         s = data
20     j = name.index(s)
21     DiemDanh[j] = 1
```

Cuối cùng, ta tìm vị trí thí sinh chưa điểm danh trong danh sách **DiemDanh** và in lên màn hình tên của thí sinh chưa ghi danh trong hội thi. Sau đó đóng 2 file dữ liệu của bài toán.

```
23 j = DiemDanh.index(0)
24 print(name[j])
25 file2.write(name[j])
26
27 file.close()
28 file2.close()
```

#### Bài 4. Đoạn con

Cho dãy số nguyên dương  $a_1, a_2, a_3, \dots, a_n$  và số nguyên  $k$ . Một đoạn con  $[l, r]$  được định nghĩa là dãy các số  $a_l, a_{l+1}, a_{l+2}, \dots, a_r$ . Độ dài của đoạn con này là  $r - l + 1$ .

**Yêu cầu:**

Trong số các đoạn con có độ dài lớn hơn hoặc bằng  $k$  của dãy  $a$ , hãy tìm đoạn con có trung bình cộng các phần tử là lớn nhất.

### Dữ liệu:

+ Dòng đầu chứa 2 số nguyên dương  $n, k$  ( $n \leq 10^5; k \leq 10^9$ ).

+ Dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$ .

**Kết quả:** Ghi ra một số nguyên duy nhất là phần nguyên của trung bình cộng lớn nhất tìm được.

INPUT	OUTPUT	Giải thích
5 3 3 7 9 2 8	6	Đoạn con cần tìm là 7, 9, 2, 8 có trung bình cộng là 6.5 và phần nguyên là 6.

**Phân tích:** Để giải bài toán, ta tiến hành duyệt từ giá trị đầu tiên của dãy tới giá trị thứ  $n - k + 1$ . Với mỗi vị trí, ta tính trung bình cộng dãy gồm  $k$  số từ vị trí đó. Từ đó ta sẽ tìm được giá trị trung bình cộng lớn nhất.

Trước tiên, ta mở 2 file dữ liệu, sau đó đọc dữ liệu vào của bài toán và lưu vào các biến tương ứng.

```

1  file = open("INPUT.INP")
2  file2 = open("OUTPUT.OUT", "w")
3
4  data = file.readline()
5  data = data.split()
6
7  n = int(data[0])
8  k = int(data[1])
9
10 data = file.readline()
11 a = data.split()
```

Để tính trung bình cộng của dãy gồm  $k$  số, ta tạo hàm **Tinh()** với tham số truyền vào là vị trí đầu tiên trong dãy số. Sau đó tính tổng các giá trị từ vị trí truyền vào, hàm sẽ trả về phần nguyên của trung bình cộng.

```

13 def Tinh(num):
14     TBC = 0
15     global k
16     global a
17     for i in range(k):
18         TBC += int(a[num+i])
19
20     TBC = int(TBC/k)
21     return TBC
```



Trong đoạn chương trình chính, ta sử dụng vòng lặp for từ vị trí đầu tới giá trị thứ  $n - k + 1$  trong dãy số. Với mỗi vị trí, ta tính trung bình cộng  $k$  số từ vị trí đó và so sánh để tìm giá trị trung bình cộng lớn nhất.

```
23 for i in range(n-k+1):
24     if i == 0:
25         Max = Tinh(i)
26     else:
27         temp = Tinh(i)
28         if temp > Max:
29             Max = temp
```

Cuối cùng, ta ghi dữ liệu ra và đóng 2 file dữ liệu của bài toán.

```
31 print(Max)
32 file2.write(str(Max))
33
34 file.close()
35 file2.close()
```

## Thư viện đề thi chọn lọc

Chương này bao gồm danh sách các đề thi Tin học trẻ chọn lọc của các tỉnh thành trên cả nước. Theo định hướng của Hội thi Tin học trẻ, những năm trước đề thi gồm nhiều nội dung khác nhau như lập trình Free Pascal, C/C++, soạn thảo văn bản MSWord, slide trình chiếu PowerPoint... Từ năm 2020 sẽ được bổ sung thêm ngôn ngữ lập trình Python, vì vậy với một số đề thi sử dụng Pascal hoặc C, bạn hoàn toàn có thể lập trình giải bài với ngôn ngữ lập trình Python. Với các dạng bài sử dụng ngôn ngữ lập trình Scratch, soạn thảo văn bản, ... các bạn có thể tự mình tìm hiểu cách giải trên Internet.

Bạn đọc được khuyến khích suy nghĩ, khám phá và giải bài toán với nhiều cách khác nhau.



## ĐỀ THI QUỐC GIA NĂM 2018

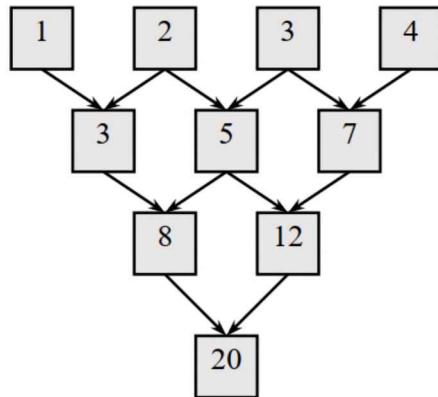
### Bài 1: Nén dãy số

Cho số nguyên dương  $N$ , ta có dãy số  $A$  gồm các số nguyên từ 1 đến  $N$ . Phép nén dãy số là tạo ra dãy số mới mà các phần tử được tạo ra bằng cách lần lượt cộng 2 số cạnh nhau của dãy số ban đầu.

Mỗi lần nén dãy số, dãy số mới sẽ ít hơn dãy trước 1 phần tử. Ta nén dãy số đến khi còn 1 phần tử, phần tử đó là giá trị nén dãy số.

**Yêu cầu:** in ra giá trị nén dãy số. Vì kết quả có thể lớn, nên chỉ cần in ra số dư của phép chia giá trị nén dãy số cho  $10^9$ .

Hình bên trên là ví dụ các phép nén dãy trong trường hợp  $N = 4$ , ta có kết quả cuối cùng là 20.



Input	Output
922	927707648

### Bài 2: Huấn luyện Pokemon

Công ty X sản xuất những con robot thông minh gọi là *pokemon*, các con *pokemon* ban đầu giống hệt nhau, mỗi con có  $n$  kỹ năng đánh số từ 1 tới  $n$  và tất cả các kỹ năng đều ở cấp độ 0 khi xuất xưởng. Các con *pokemon* sau đó sẽ được huấn luyện bằng một chương trình đặc biệt nhằm gia tăng cấp độ các kỹ năng, để tăng cấp độ kỹ năng thứ  $i$  lên 1 đơn vị cần thời gian huấn luyện đúng  $i$  giây ( $\forall i = 1, 2, \dots, n$ ). Ngoài ra do vấn đề kỹ thuật, không kỹ năng nào được huấn luyện vượt quá cấp độ  $m$ .

Công ty X nhận được đơn đặt hàng  $k$  con *pokemon* hoàn toàn phân biệt, tức là hai con *pokemon* bất kỳ phải có ít nhất một kỹ năng ở cấp độ khác nhau. Hãy cho biết tổng số giây ít nhất cần để huấn luyện  $k$  con *pokemon* thỏa mãn yêu cầu trên.

STT	Kỹ năng 1	Kỹ năng 2	Kỹ năng 3	Thời gian huấn luyện
1	0	0	0	0
2	1	0	0	1
3	2	0	0	2
4	0	1	0	2
5	3	0	0	3
6	0	0	1	3
7	1	1	0	3
8	4	0	0	4
9	2	1	0	4
10	0	2	0	4
				Tổng thời gian: 26

Ví dụ với số kỹ năng  $n = 3$ , giới hạn cấp độ kỹ năng  $m = 4$ , số con pokemon đặt hàng  $k = 10$ . Công ty có thể huấn luyện 10 con pokemon với các kỹ năng như sau:

**Dữ liệu vào:** Ba dòng chứa 3 số  $n, m, k$  mỗi số trên một dòng

**Dữ liệu ra:** Số tự nhiên duy nhất - kết quả bài toán

Input	Output
3	22
4	
9	

### Lập trình với ngôn ngữ Scratch

#### Bài 3: Trò chơi: Giải toán lấy quà

Chính giữa màn hình là một cây thông Noel lớn. Trên cây thông có rất nhiều loại quà khác nhau được gắn ngẫu nhiên trên các vị trí dọc theo cây. Nhiệm vụ của em là trong thời gian 1 phút, điều khiển nhân vật chính (HS) lấy được nhiều phần quà nhất có thể từ cây thông này. Nhấn phím Space để bắt đầu trò chơi.

Khi bắt đầu chơi, em sẽ điều khiển nhân vật chính bằng các phím lên, xuống, phải, trái, nhân vật chính sẽ di chuyển lên, xuống, phải, trái 5 bước tương ứng (Hình 1).

Khi va chạm vào một món quà đầu tiên, người chơi sẽ ôm quà này và phải đi đến vị trí có vòng tròn bên phải để bỏ đồ chơi vào bên trong vòng tròn mới được tính là nhận được món quà đó (Hình 2).

Nhưng mỗi khi đến vị trí vòng tròn, trước khi đưa được món quà này vào bên trong vòng tròn, Giáo viên và Bảng sẽ xuất hiện yêu cầu em làm một bài toán (Hình 3). Làm xong bài toán này thì món quà đó mới vào được bên trong vòng tròn và em được quyền đi lấy tiếp món quà khác.



Câu hỏi toán có thể là một trong 2 loại sau:

- Tính số các ước thực sự của một số tự nhiên cho trước.

- Trả lời yes/no cho câu hỏi: Số tự nhiên sau có phải số nguyên tố hay không?

Với mỗi câu hỏi em cần trả lời liên tục cho tới khi đúng. Nếu làm đúng, GV nói “đúng rồi” và sai 1 giây, GV, bảng biến mất, quà được đưa vào vòng tròn và em sẽ tiếp tục công việc tìm quà của mình.



Sau 60 giây, chương trình dừng lại, GV lại xuất hiện và thông báo em đã nhận được bao nhiêu phần quà (Hình 4).

Em hãy viết chương trình mô tả trò chơi trên.

Yêu cầu bắt buộc:

Số lượng các món quà phải lớn hơn hoặc bằng 10. Các hình ảnh quà lấy từ thư viện của Scratch.

Các món quà cần gắn và xếp ngẫu nhiên dọc theo thân cây thông. Hình ảnh minh họa trên chỉ là ví dụ.

### MỞ RỘNG

Trò chơi trên có thể mở rộng theo nhiều cách khác nhau để trở nên hay hơn, đa dạng hơn và hấp dẫn hơn. Gợi ý một số hướng mở rộng:

Tăng số lượng các quà sinh nhật lên cho thêm phần hấp dẫn.

Mở rộng thêm các dạng toán khác cho phong phú hơn. Ví dụ các dạng toán sau có thể đưa thêm vào chương trình:

+ Tính giá trị của một biểu thức toán học, ví dụ dạng  $(m + n) * d$ .

+ Tìm phần tử tiếp theo của một dãy số có quy luật cho trước, ví dụ dãy các số chẵn liên tiếp.

Bổ sung thêm chức năng: nhân vật chính có thể thay thế quà. Khi đã nhận một món quà trên cây, có thể bỏ lại quà đó và chọn quà khác.

Mỗi món quà có một âm thanh tương ứng. Khi nhận một món quà thì âm thanh tương ứng sẽ vang lên.

## ĐỀ THI SƠ KHẢO QUỐC GIA VÒNG TỰ DO NĂM 2021

### Bài 1. Cặp số đồng đội

Kì thi Tin học trẻ là một trong những kì thi lớn dành cho học sinh phổ thông Việt Nam. Để tổ chức thành công kỳ thi Tin học trẻ, ngoài Ban tổ chức, kì thi thì Hội đồng Ban giám khảo đóng vai trò rất quan trọng. Thầy Nguyễn Vũ Hoàng Vương là một thầy giáo trẻ nhưng đã tham gia Hội đồng Ban giám khảo nhiều năm nay. Nhắc đến thầy Vương, Ban giám khảo đều nhớ về một đồng đội xuất sắc và chân thành. Một bài toán số học lấy cảm hứng từ đồng đội được dùng làm đề thi Tin học trẻ năm 2021 như sau:

Một cặp số nguyên dương ( $a, b$ ) mà  $a$  chia hết cho  $b$  hoặc  $b$  chia hết cho  $a$  được gọi là cặp số đồng đội. Cặp số đồng đội ( $a, b$ ) và cặp số đồng đội ( $u, v$ ) được gọi là giống nhau khi  $a = u$  và  $b = v$ .

**Yêu cầu:** Cho số nguyên dương  $N$  ( $2 \leq N \leq 10^9$ ), hãy đếm số cặp số đồng đội mà  $a + b = N$ .

**Dữ liệu:** Vào từ thiết bị vào chuẩn gồm một số nguyên dương  $N$  duy nhất.

**Kết quả:** Ghi ra thiết bị ra chuẩn một số nguyên duy nhất là số cặp số đồng đội thỏa mãn yêu cầu.

Ví dụ:

Dữ liệu vào	Kết quả ra	Giải thích
10	5	Các cặp số đồng đội thỏa mãn: (1, 9); (2, 8); (5, 5); (8, 2); (9, 1)

Ràng buộc:

- + Có 50% số test ứng với 50% số điểm của bài thỏa mãn:  $N \leq 10^3$ ;
- + 30% số test khác ứng với 30% số điểm của bài thỏa mãn:  $N \leq 10^6$ ;
- + 20% số test còn lại ứng với 20% số điểm của bài không có ràng buộc gì thêm.

### Bài 2. Ước số

Một số nguyên dương  $n$  được phân tích thành thừa số nguyên tố như sau:

$$n = p_1^{k_1} \times p_2^{k_2} \times \dots \times p_m^{k_m}$$

**Yêu cầu:** Cho hai số nguyên không âm  $A \leq B$ , đếm số lượng ước của  $n$  trong  $[A, B]$ .



**Dữ liệu:**

- + Dòng đầu chứa số nguyên dương  $m$ ;
- + Tiếp theo là  $m$  dòng, dòng thứ  $i$  chứa hai số nguyên dương  $p_i$  và  $k_i$ , trong đó  $p_i, k_i$  không vượt quá  $10^9$  và các số  $p_i$  là số nguyên tố đôi một khác nhau;
- + Ba dòng cuối tương ứng với ba câu hỏi, mỗi dòng chứa hai số nguyên không âm  $A, B$  tương ứng với một câu hỏi.

**Kết quả:** Ghi ra chuẩn ba dòng, mỗi dòng ghi ước số tìm được.

Ví dụ:

Dữ liệu vào	Kết quả ra
3	5
2 4	9
3 4	5
5 4	
1 5	
1 10	
1 5	

Ràng buộc:

- + Có 40% số test tương ứng với 40% số điểm của bài có  $m \leq 5; 0 \leq A \leq B \leq 10^6$ ;
- + Có 40% số test tương ứng với 40% số điểm của bài có  $m \leq 10; 0 \leq A \leq B \leq 10^9$ ;
- + Có 20% số test còn lại ứng với 20% số điểm của bài có  $m \leq 25; 0 \leq A \leq B \leq 10^9$ .

### Bài 3. Cân đĩa

Cho một cân hai đĩa và  $n$  quả cân có khối lượng đôi một khác nhau  $w_1, w_2, \dots, w_n$ . Tiến hành đặt lần lượt từng quả cân lên một trong hai đĩa của cân và đảm bảo rằng tổng khối lượng bên trái luôn nhỏ hơn hoặc bằng tổng khối lượng bên phải.

**Yêu cầu:** Cho  $n$  quả cân có khối lượng  $w_1, w_2, \dots, w_n$ , hãy đếm số cách xếp  $n$  quả cân thỏa mãn. Hai cách được gọi là khác nhau nếu thứ tự xếp các quả cân khác nhau hoặc tồn tại một quả cân nằm ở đĩa khác nhau.

**Dữ liệu:**

- + Dòng 1: Chứa số nguyên  $n$ ;
- + Dòng 2: Chứa  $n$  số nguyên dương  $w_1, w_2, \dots, w_n$ .

**Kết quả:** Một dòng chứa một số nguyên là số cách xếp  $n$  quả cân lên đĩa.

Ví dụ:

Dữ liệu vào	Kết quả ra
2	3
1 2	

Dữ liệu vào	Kết quả ra
3	15
10 11 12	

*Giải thích: Ở ví dụ bên trái, có 8 cách sắp xếp các quả cân lên hai bàn cân như sau:*

1. *Đặt quả cân 1 bên trái rồi đặt quả cân 2 bên trái;*
2. *Đặt quả cân 1 bên trái rồi đặt quả cân 2 bên phải;*
3. *Đặt quả cân 1 bên phải rồi đặt quả cân 2 bên trái;*
4. *Đặt quả cân 1 bên phải rồi đặt quả cân 2 bên phải;*
5. *Đặt quả cân 2 bên trái rồi đặt quả cân 1 bên trái;*
6. *Đặt quả cân 2 bên trái rồi đặt quả cân 1 bên phải;*
7. *Đặt quả cân 2 bên phải rồi đặt quả cân 1 bên trái;*
8. *Đặt quả cân 2 bên phải rồi đặt quả cân 1 bên phải;*

*Tuy nhiên chỉ có 3 cách (cách 4, 7, 8) là đảm bảo trong toàn bộ quá trình sắp xếp các quả cân, đĩa bên trái luôn nhỏ hơn hoặc bằng đĩa cân bên phải.*

*Ràng buộc:*

- + Có 40% số test ứng với 40% số điểm của bài có  $n \leq 7$  và  $w_i \leq 1000$  ( $1 \leq i \leq n$ ).
- + Có 40% số test ứng với 40% số điểm của bài có  $n \leq 14$  và  $w_i \leq 1000$  ( $1 \leq i \leq n$ ).
- + Có 40% số test ứng với 40% số điểm của bài có  $n \leq 28$  và  $w_i = 2^{i-1}$  ( $1 \leq i \leq n$ ).



## ĐỀ THI THÀNH PHỐ HÀ NỘI NĂM 2021

### Bài 1. Ghép hình

Ghép hình là trò chơi mà các bạn nhỏ hay cả người lớn đều rất thích. Có một trò chơi ghép hình rất đơn giản như sau: cho ba mảnh ghép hình chữ nhật, hãy kiểm tra xem có thể ghép lại thành một hình chữ nhật to hay không, nếu có, hãy tính đường chéo của hình chữ nhật đó.

Chú ý: không được xếp các hình chữ nhật chồng lên nhau; nếu có nhiều cách xếp thì chọn cách xếp tạo ra đường chéo lớn hơn.

#### Dữ liệu:

Vào từ thiết bị vào chuẩn gồm 3 dòng, mỗi dòng gồm hai số nguyên dương  $a$ ,  $b$  mô tả kích hình chữ nhật.

#### Kết quả:

Ghi ra thiết bị ra gồm:

+ Nếu ghép được thành hình chữ nhật thì in ra độ dài của đường chéo hình chữ nhật đó (in ra phần nguyên được làm tròn xuống).

+ Nếu không ghép được thành hình chữ nhật thì in ra -1.

Ví dụ:

Dữ liệu	Kết quả	Giải thích	
2 2	4	Ghép được thành hình chữ nhật như hình bên. Đường chéo có độ dài là 4.	
1 2			
2 1			
4 3	-1	Không thể ghép được thành hình chữ nhật	
1 1			
1 1			

Ràng buộc:

- + Có 50% số test có  $0 < a, b \leq 10^3$ ;
- + Có 30% số test có  $0 < a, b \leq 10^6$ ;
- + 20% số test còn lại có  $0 < a, b \leq 2 \times 10^9$ .

## Bài 2. Robot tặng quà

Để khuyến khích các bạn trẻ nghiên cứu và chế tạo Robot, Thành Đoàn Hà Nội đã tổ chức một hoạt động có tên “ROBOT tặng quà” dành cho các bạn thí sinh của kỳ thi Tin học trẻ. Hoạt động đó diễn ra như sau:

Trên trục số có một Robot đặt tại điểm 0 và n bạn đánh số từ 1 tới n, bạn thứ i đứng tại điểm ai trên trục số (có thể có nhiều bạn đứng cùng một vị trí). Người chơi chính được lựa chọn một số nguyên d ( $d > 1$ ) và thiết đặt bước nhảy của Robot là d. Khi đó, từ một vị trí, Robot có thể nhảy tiến hoặc nhảy lùi một khoảng cách đúng bằng d trên trục số, tức là nếu Robot đang ở vị trí x, Robot chỉ có thể nhảy sang một trong hai vị trí  $x + d$  hoặc  $x - d$ . Một bạn sẽ được Robot tặng quà nếu tồn tại cách di chuyển Robot nhảy từ điểm 0 tới vị trí bạn đó sau một số hữu hạn lần nhảy.

Với mong muốn có nhiều bạn được Robot tặng quà, em hãy giúp người chơi chính chọn tham số nguyên d  $> 1$  để có nhiều bạn được Robot tặng quà nhất, cho biết số bạn được Robot tặng quà.

**Dữ liệu:** Vào từ thiết bị vào chuẩn:

- + Dòng đầu chứa số nguyên dương  $T \leq 10^5$  là số bộ dữ liệu.
- + T nhóm dòng tiếp theo, mỗi nhóm gồm hai dòng mô tả một bộ dữ liệu:
  - Dòng 1 chứa số nguyên dương  $n \leq 10^6$ .
  - Dòng 2 chứa n số nguyên:  $a_1, a_2, \dots, a_n$ , mỗi số cách nhau bởi dấu cách ( $|a_i| \leq 10^6$  với  $1 \leq i \leq n$ )

Tổng các giá trị n trong các bộ dữ liệu vào không vượt quá  $10^6$ .

**Kết quả:** Ghi ra thiết bị ra chuẩn

Với mỗi bộ dữ liệu ghi ra một số nguyên duy nhất trên một dòng là số bạn được Robot tặng quà theo phương án tìm được.

**Ví dụ:**

Dữ liệu	Kết quả	Giải thích
2	2	- Bộ dữ liệu thứ nhất: chọn d bằng 2, 2, 4 hoặc 5.
4	3	- Bộ dữ liệu thứ hai: chọn d bằng 5.
1 20 12 15		
3		
5 -5 15		

Ràng buộc:

- + Có 20% số test có  $n \leq 100$ ;  $|a_i| \leq 100$  với  $1 \leq i \leq n$ ;
- + Có 30% số test có  $n \leq 2000$ ;  $|a_i| \leq 2000$  với  $1 \leq i \leq n$ ;
- + Có 50% số test còn lại không có ràng buộc bổ sung.



### Bài 3. Ngày nghỉ phép

Một công ty lập trình lên một kế hoạch làm việc cho  $N$  ngày, với  $T$  dự án, dự án thứ  $i$  có thời gian kéo dài từ ngày thứ  $a_i$  đến ngày thứ  $b_i$ . Các nhân viên phải đi làm trong thời gian công ty có dự án. Để đảm bảo số nhân viên làm việc và vẫn để nhân viên được nghỉ ngơi, công ty có quy định là những ngày không có dự án thì nhân viên được nghỉ và trong mỗi dự án nhân viên được nghỉ phép không quá một ngày.

Là một nhân viên lười biếng, Dino muốn nghỉ thật nhiều nhưng vẫn phải đúng luật của công ty, nên Dino sẽ lên kế hoạch nghỉ để mỗi dự án đều có chứa đúng một ngày nghỉ.

Em hãy lập trình giúp Dino tính xem theo kế hoạch trong  $N$  ngày của công ty và theo mong muốn của Dino thì được nghỉ tối đa bao nhiêu ngày?

**Dữ liệu:** Vào từ thiết bị nhập chuẩn:

+ Dòng đầu tiên chứa hai số nguyên  $N$ ,  $T$  ( $2 \leq N, T \leq 10^6$ ) tương ứng là kế hoạch trong  $N$  ngày và  $T$  dự án của công ty.

+  $T$  dòng sau, dòng thứ  $i$  chứa hai số nguyên  $a_i$  và  $b_i$  mô tả thời gian bắt đầu và kết thúc dự án thứ  $i$  của công ty ( $1 \leq i \leq T; 1 \leq a_i \leq b_i \leq N$ ).

**Kết quả:** Ghi ra thiết bị ra chuẩn một dòng chứa một số nguyên là số ngày tối đa Dino có thể được nghỉ. Nếu không có cách chọn mà mỗi dự án đều có chứa đúng một ngày nghỉ thì in ra -1.

Ví dụ:

Dữ liệu	Kết quả	Giải thích
6 3 1 4 3 5 2 4	3	Có thể nghỉ 3 ngày: Ngày 2, ngày 5, ngày 6.
5 3 1 5 2 3 4 5	-1	Không có cách chọn để mỗi dự án nghỉ đúng 1 ngày.

Ràng buộc:

- + Có 20% test có  $2 \leq N, T \leq 20$ ;
- + Có 20% test có  $2 \leq N, T \leq 5000$ ;
- + Có 30% test có  $2 \leq N, T \leq 10^5$ ;
- + 30% test còn lại không có ràng buộc bổ sung.

## ĐỀ THI THÀNH PHỐ HÀ NỘI NĂM 2019

### Bài 1: Không hoàn hảo

Giá trị không hoàn hảo của một số nguyên dương  $N$  được định nghĩa như sau: là giá trị tuyệt đối của  $N$  trừ đi các ước của  $N$  (ước dương, bé hơn  $N$ ). Ví dụ:

- + Giá trị không hoàn hảo của 6 là:  $F(6) = |6 - 1 - 2 - 3| = 0$
- + Giá trị không hoàn hảo của 18 là:  $F(18) = |18 - 9 - 6 - 3 - 2 - 1| = |-3| = 3$
- + Giá trị không hoàn hảo của 7 là:  $F(7) = |7 - 1| = 6$
- + Giá trị không hoàn hảo của 8 là:  $F(8) = |8 - 4 - 2 - 1| = 1$

Cho 2 số nguyên dương  $L$  và  $R$ , tính tổng giá trị không hoàn hảo của các số lớn hơn hoặc bằng  $L$  và nhỏ hơn hoặc bằng  $R$ .

### Bài 2: Số liền sau

Số liền sau của một số nguyên dương  $N$  là một số nguyên dương  $M$  sao cho thỏa mãn các điều kiện sau:

- + Số  $M$  bé nhất mà  $M$  lớn hơn  $N$ .
- + Tổng các chữ số của  $M$  bằng tổng các chữ số của  $N$ .

Ví dụ về dãy số liền sau: 987; 996; 1599; 1689; 1698; 1779;... Ta có số liền sau thứ 4 của số 987 là số 1698.

Cho 2 số nguyên dương  $N$  và  $K$ , tìm số liền sau thứ  $K$  của  $N$ .

### Bài 3: Trò chơi Bắn bóng bay – Lập trình bằng ngôn ngữ Scratch

#### Phần 1: Cơ bản

Trò chơi gồm có 3 nhân vật chính là 3 quả bóng bay. Như hình sau:

##### Mô tả trò chơi:

Khi bắt đầu, các quả bóng bay có màu sắc ngẫu nhiên di chuyển từ phía dưới màn hình lên, trên mỗi quả bóng bay có thể hiện một câu đố tìm kết quả của một phép tính cộng hai số hạng (để cho đơn giản, các số hạng có giá trị nhỏ hơn 20).

Khi quả bóng bay di chuyển qua cạnh trên của màn hình hoặc khi người chơi nhập đáp án đúng với kết quả của câu đố trên quả bóng bay thì quả bóng bay đó sẽ biến mất và sau 1 đến 2 giây sẽ có quả bóng bay mới với câu đố mới và lại di chuyển từ dưới màn



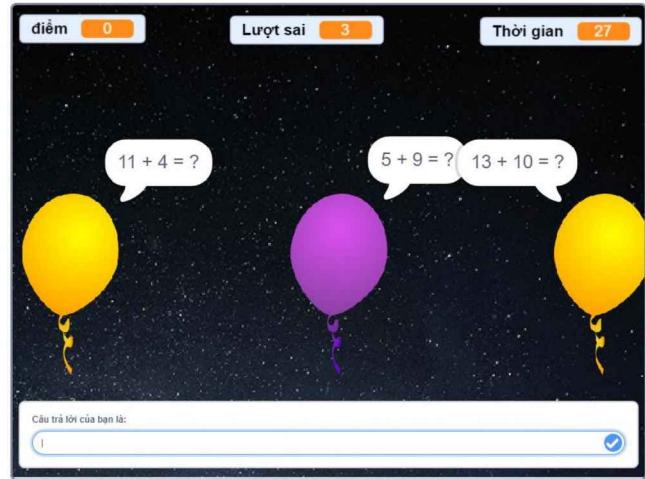
hình lên trên. Trò chơi sẽ lặp lại như vậy. Với mỗi câu đố của quả bóng trả lời đúng, người chơi sẽ được 1 điểm.

Thời gian của trò chơi là 30 giây. Thời gian sẽ đếm ngược, khi thời gian về 0 thì trò chơi sẽ kết thúc.

Phía trên bên phải của màn hình thể hiện thời gian còn lại của trò chơi. Phía trên bên trái của màn hình thể hiện số điểm của người chơi.

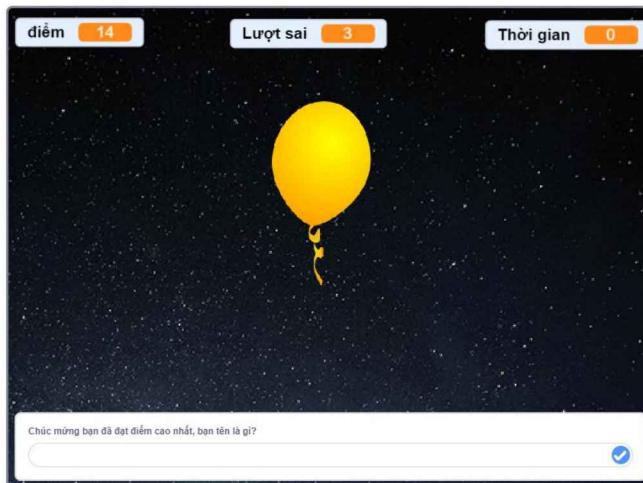
### Phần 2: Mở rộng

Để khuyến khích người chơi tính toán cẩn thận và không nhập sai nhiều, trò chơi có thêm một luật: Nếu nhập đáp án không đúng với bất kì đáp án của câu đố ở quả bóng nào, thì gọi là kết quả sai. Mỗi người chơi có 3 lần kết quả sai, số lần kết quả sai còn lại được thể hiện phía trên, ở giữa màn hình và đếm ngược từ 3 về 0. Vậy trò chơi sẽ kết thúc khi người chơi bị lượt sai về 0 hoặc hết thời gian 30 giây.



Tạo Highscore để lưu lại thông tin người chơi có điểm cao nhất. Thông tin này không bị khởi động lại sau mỗi lần chơi. Sau mỗi lần kết thúc trò chơi, nếu số điểm cao hơn số điểm ở Highscore thì đề nghị nhập tên người chơi và cập nhật thông tin Highscore. Highscore được hiển thị như sau:

**Highscore Dino - 11** Với Dino là tên người chơi có điểm cao nhất và 11 là số điểm cao nhất trong các lần chơi.



Hỏi tên khi điểm lớn hơn Highscore

## ĐỀ THI THÀNH PHỐ ĐÀ NẴNG NĂM 2020

### Bài 1. Biến đổi

Từ một số nguyên dương  $K$  ban đầu, ta thực hiện biến đổi số  $K$  theo quy tắc biến đổi sau đây: Nếu  $K$  chia hết cho 6 thì thay số  $K$  bởi thương  $K : 6$ ; còn nếu  $K$  không chia hết cho 6 thì thay số  $K$  bởi tích  $3 \times K$ .

**Yêu cầu:** Hãy xác định số lần biến đổi theo quy tắc trên để  $K$  bằng 1.

**Dữ liệu:** Một dòng chứa một số nguyên dương  $K$  ( $K \leq 10^9$ ).

**Kết quả:** In ra số nguyên dương  $m$  là số lần biến đổi để  $K$  bằng 1.

**Chú ý:** Trong trường hợp không thể biến đổi  $K$  bằng 1 theo quy tắc biến đổi trên thì in ra màn hình số  $-1$ .

INPUT	OUTPUT	INPUT	OUTPUT
12	3	10	-1

### Bài 2. Hình vuông

Cho một lưới hình vuông chứa rất nhiều ô vuông nhỏ. Mỗi ô vuông trong lưới hình vuông này được xác định vị trí bởi một cặp số  $(i, j)$ , trong đó  $i$  là chỉ số hàng và  $j$  là chỉ số cột. Các hàng được đánh chỉ số bởi các số tự nhiên bắt đầu từ 1, 2, 3, ... kể từ trên xuống dưới; các cột được đánh chỉ số bởi các số tự nhiên bắt đầu từ 1, 2, 3, ... kể từ trái sang phải. Các ô vuông trong lưới hình vuông được ghi một số tự nhiên bằng tích của chỉ số hàng và chỉ số cột của ô vuông đó.

Chọn ra hình vuông chứa  $K \times K$  ô vuông trong lưới hình vuông đã cho. Gọi  $T$  là tổng các số trong các ô vuông có trong hình vuông đã chọn.

**Yêu cầu:** Hãy tìm số dư trong phép chia  $T : 20192020$ .

**Ví dụ:** Cho lưới hình vuông, ta chọn một hình vuông gồm  $3 \times 3$  ô vuông, trong đó ô vuông ở góc bên trái có chỉ số hàng bằng 2 và chỉ số cột bằng 1 (hình minh họa).

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25



**Input:**

- + Một dòng duy nhất chứa 3 số nguyên  $i, j, K$  ( $1 \leq i, j \leq 1000; 1 \leq K \leq 10^8$ )
- + Trong đó  $i$  và  $j$  lần lượt là chỉ số hàng và chỉ số cột của ô vuông ở góc bên trái của hình vuông được chọn;  $K$  là số ô vuông trên một hàng (và là số ô vuông trên một cột) của hình vuông được chọn.

**Output:** Kết quả cần tìm.

INPUT	OUTPUT
2 1 3	54

**Bài 3. Số Py-ta-go**

Một số tự nhiên được gọi là số Py-ta-go là số được tạo thành từ việc ghép 3 số tự nhiên  $a; b; c$  theo một trật tự bất kì với nhau (không thay đổi trật tự các chữ số trong mỗi số  $a; b; c$ ), trong đó 3 số  $a; b; c$  thỏa mãn điều kiện tổng bình phương của 2 số nào đó trong 3 số này bằng bình phương của số còn lại.

Ví dụ: Với 3 số  $a = 6; b = 8; c = 10$ , ta có  $6^2 + 8^2 = 10^2$  và nếu ghép chúng lại với nhau theo một trật tự bất kì thì ta có được tất cả 6 số Py-ta-go như sau: 6810; 6108; 8610; 8106; 1068; 1086.

**Yêu cầu:** Cho  $X$  là một số Py-ta-go được ghép từ bộ ba số  $a; b; c$  như đã trình bày ở trên. Hãy tìm số lớn nhất trong 3 số  $a; b; c$ .

**Dữ liệu:** Một dòng chứa số nguyên dương  $X$  có ít nhất 3 chữ số và có nhiều nhất 24 chữ số.

**Kết quả:** Ghi ra số nguyên dương cần tìm theo yêu cầu của đề.

**Chú ý:** Nếu có nhiều kết quả thì ghi ra số lớn nhất trong các kết quả tìm được.

INPUT	OUTPUT
8106	10

## ĐỀ THI THÀNH PHỐ ĐÀ NẴNG NĂM 2019

### Bài 1. Tháp

Có một tháp các ô vuông bằng nhau có hình dạng giống một tam giác cân. Các hàng tính từ trên xuống dưới có số ô vuông lần lượt là: 1; 3; 5; 7; ... Một tháp ô vuông có  $n$  hàng gọi là tháp ô vuông bậc  $n$  ( $n$  là số tự nhiên).

**Yêu cầu:** Cho trước một tháp ô vuông bậc  $n$ . Hãy tính xem trong tháp ô vuông này có tất cả mấy hình vuông tạo thành từ các ô vuông đó.

**Dữ liệu:** Chứa một số  $n$  ( $n \leq 5 * 10^5$ ).

**Kết quả:** Ghi ra số nguyên  $m$  là số các hình vuông cần tìm.

<i>Input</i>	<i>Output</i>
3	11

### Bài 2. Lũy thừa

Mọi số nguyên dương  $a$  đều có thể viết được dưới dạng lũy thừa bậc  $n$  của số nguyên dương  $b$  (với  $n$  là số tự nhiên). Chẳng hạn:  $27 = 3^3$ ;  $8 = 2^3$ . Một số nguyên dương  $a$  có thể có nhiều cách biểu diễn dưới dạng một lũy thừa, chẳng hạn:

$$81 = 81^1 = 9^2 = 3^4.$$

**Yêu cầu:** Cho trước 3 số nguyên dương  $a$ ;  $b$ ;  $c$ . Gọi  $x$  là tích của 3 số  $a$ ;  $b$ ;  $c$ . Hỏi trong các cách viết số  $x$  thành một lũy thừa bậc  $n$  của một số nguyên dương thì số mũ  $n$  lớn nhất bằng bao nhiêu?

**Dữ liệu:**

Chứa 3 số  $a$ ;  $b$ ;  $c$  mỗi số nằm trên một dòng ( $a$ ;  $b$ ;  $c \leq 10^{12}$ ).

**Kết quả:** Ghi ra số  $n$  thỏa mãn yêu cầu trên.

<i>Input</i>	<i>Output</i>
3	4
3	
9	



### Bài 3. Tam giác số

Cho tam giác số có hình dạng một tam giác vuông cân gồm  $n$  cột và  $n$  hàng, trong đó hàng thứ  $i$  có  $i$  số (như hình sau):

2							
5	8						
11	14	17					
20	23	26	29				
32	35	38	41	44			
...	...	...	...	...	...	...	
...	...	...	...	...	...	...	

**Yêu cầu:** Tính tổng các số ở phần còn lại của tam giác sau khi đã xóa đi  $k$  cột liên tiếp (tính từ trái sang phải) của tam giác này.

**Dữ liệu:** Chứa 2 số nguyên dương  $n$  và  $k$  nằm trên 1 dòng, mỗi số cách nhau ít nhất một dấu cách, trong đó:

- + Số  $n$  là số hàng của tam giác ban đầu ( $n \leq 10^{16}$ ).
- + Số  $k$  là số cột được xóa ( $k < n$ ,  $k \leq 10^5$ ).

**Kết quả:** Ghi ra số nguyên dương  $m$  thỏa mãn yêu cầu.

Input	Output
5 3	114

**Giải thích:** Với số hàng ban đầu  $n = 5$  và xóa đi  $k = 3$  cột liên tiếp (tính từ trái sang phải) thì tổng các số ở phần còn lại của tam giác là  $29 + 41 + 44 = 114$ .

## ĐỀ THI BẮC GIANG NĂM 2021

### Bài 1. Tính tổng

Cho số nguyên dương  $n$ , tính giá trị biểu thức  $S = 1 + 2 + 3 + \dots + n$ .

**Dữ liệu vào:** đọc từ file SUM.INP một số nguyên dương  $n$  ( $n \leq 10^9$ )

**Kết quả:** ghi ra file văn bản SUM.OUT một số duy nhất là giá trị  $S$  tính được.

SUM.INP	SUM.OUT	Giải thích
5	15	$S = 1 + 2 + 3 + 4 + 5 = 15$

### Bài 2. Số may mắn

Ở đất nước Omega xinh đẹp, cuộc thi của các học sinh chỉ là những cuộc trải nghiệm vui vẻ. Các thí sinh dự thi tham gia nhiệt tình với tinh thần giao lưu, học hỏi, không nặng nề về kết quả. Kết thúc cuộc thi mặt dù có xếp giải nhưng tất cả các thí sinh dự thi đều có giải thưởng mà đôi khi thí sinh đạt giải cao lại được phần thưởng ít hơn những bạn không được giải nếu họ kém may mắn.

Ban tổ chức có  $n$  hộp quà, trong mỗi hộp quà ghi một số nguyên dương. Thí sinh chọn hộp quà nào thì được nhận phần thưởng có giá trị bằng tổng các ước số nguyên dương của số bên trong hộp quà đó. Ví dụ thí sinh chọn hộp quà có ghi số 6 thì phần thưởng của thí sinh là  $1 + 2 + 3 + 6 = 12$ .

Em hãy chọn một hộp quà sao cho giá trị của phần thưởng nhận được là lớn nhất!

**Dữ liệu vào:** đọc từ file văn bản MAYMAN.INP gồm:

+ Dòng 1: ghi số nguyên dương  $n$  ( $n \leq 10^4$ ).

+ Dòng 2: ghi  $n$  số nguyên dương  $a_1, a_2, \dots, a_n$  ( $a_i \leq 10^6$ ,  $i = 1, 2, \dots, n$ ). Trong đó  $a_i$  là số nguyên dương trong hộp quà thứ  $i$ . Hai số trên cùng dòng cách nhau ít nhất một khoảng trắng.

**Kết quả:** ghi ra file văn bản MAYMAN.OUT gồm một số duy nhất là giá trị phần thưởng mà em nhận được.

MAYMAN.INP	MAYMAN.OUT	Giải thích
5 3 6 9 11 4	13	Giá trị phần thưởng lớn nhất khi em chọn hộp thứ 3 là: $1 + 3 + 9 = 13$ .



### Bài 3. Xử lý xâu ký tự

Cho xâu ký tự  $S$  (không quá 200 ký tự) chỉ gồm chữ cái, chữ số và dấu gạch nối. Em hãy thực hiện lần lượt các thao tác sau:

- + Thao tác 1: xóa các ký tự trong xâu  $S$  không phải là chữ số.
  - + Thao tác 2: tìm số lớn nhất có k chữ số gồm k ký tự liên tiếp trong xâu  $S$ ;
- Dữ liệu vào:** đọc từ file văn bản XAU.INP gồm:
- + Dòng 1 ghi xâu ký tự  $S$ ;
  - + Dòng 2 ghi số lớn nhất tìm được sau khi thực hiện thao tác 2.

Lưu ý: bài toán luôn có nghiệm.

XAU.INP	XAU.OUT
A 0 9 a2b _89 3	092893
2	93

### Bài 4. Chia quà

Bố Mít vừa đi công tác xa về, bố mua rất nhiều kẹo bánh cho hai chị em. Sau khi đếm, Mít thấy có tổng  $n$  gói kẹo, gói kẹo thứ  $i$  có ai cái. Ban đầu Mít dành cho em phần nhiều nhưng mẹ Mít muốn có sự công bằng tương đối giữa hai chị em. Mẹ bảo Mít chia kẹo làm hai phần sao cho độ chênh lệch số kẹo giữa hai phần là ít nhất và Mít không được chia nhỏ số kẹo trong mỗi túi. Tất nhiên là số kẹo hai phần không bằng nhau thì Mít sẽ nhận phần ít hơn.

Em hãy giúp Mít chia quà và tính độ chênh lệch số kẹo ít nhất giữa hai phần quà.

**Dữ liệu vào:** đọc từ file văn bản CHIAQUA.INP gồm:

- + Dòng 1: ghi số nguyên dương  $n$  ( $n \leq 50$ ).
- + Dòng 2: ghi  $n$  số nguyên dương lần lượt là:  $a_1, a_2, a_3, \dots, a_n$  ( $a_i \leq 100; i = 1, 2, \dots, n$ ). Trong đó ai là số kẹo trong gói thứ  $i$ . Các số trên cùng một dòng cách nhau ít nhất một dấu cách.

**Kết quả:** ghi ra file CHIAQUA.OUT gồm một số duy nhất là độ lệch ít nhất giữa 2 phần quà.

CHIAQUA.INP	CHIAQUA.OUT	Giải thích
4	2	+ Tổng số kẹo của Mít là 12 (gói số 3)
3 4 12 7		+ Tổng số kẹo của em là 14 (gói 1, 2, 4)

## ĐỀ THI ĐÔNG TRIỀU – QUẢNG NINH NĂM 2021

### Bài 1. Lịch

An muốn tạo lịch cho tháng hiện tại. Với mục đích này, An vẽ một bảng trong đó các cột tương ứng với các tuần (một tuần là 7 ngày từ Thứ Hai đến Chủ Nhật), các hàng tương ứng với các thứ trong tuần và các ô chứa ngày của tháng. Ví dụ lịch cho tháng 5 năm 2021 sẽ giống như hình sau:

Thứ Hai		3	10	17	24	31
Thứ Ba		4	11	18	25	
Thứ Tư		5	12	19	26	
Thứ Năm		6	13	20	27	
Thứ Sáu		7	14	21	28	
Thứ Bảy	1	8	15	22	29	
Chủ Nhật	2	9	16	23	30	

Cho một tháng và thứ trong tuần của ngày đầu tiên của tháng đó, An muốn biết bảng của tháng đó có bao nhiêu cột, giả sử rằng năm không nhuận.

#### Dữ liệu:

Gồm một dòng chứa hai số nguyên  $m$  và  $d$  ( $1 \leq m \leq 12$ ,  $1 \leq d \leq 7$ ) lần lượt là số của tháng (1 là tháng 1, 2 là tháng 2,...) và thứ của ngày đầu tiên trong tháng đó (1 là Thứ Hai, 2 là Thứ Ba, ..., 7 là Chủ Nhật).

#### Kết quả:

Ghi ra một số nguyên là số cột mà bảng cần có.

CAL.INP	CAL.OUT
5 6	6
1 1	5
11 6	5



## Bài 2. Ngày lễ bình đẳng

Hôm nay ở Berland là ngày lễ của sự bình đẳng, để tôn vinh ngày lễ, nhà vua đã quyết định cân bằng phúc lợi của tất cả công dân ở Berland bằng ngân khố của vương quốc.

Ở Berland có  $n$  công dân, phúc lợi của mỗi người được ước tính bằng ai burles (burle là đơn vị tiền tệ ở Berland). Bạn là thủ quỹ hoàng gia, cần phải tính khoản tiền tối thiểu để cân bằng số tiền của tất cả các người dân. Nhà vua chỉ có thể ban tiền chứ không có quyền lấy tiền của người dân.

**Dữ liệu:** Dòng đầu tiên chứa số nguyên  $n$  ( $1 \leq n \leq 100$ ) là số người dân trong vương quốc. Dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$ . Trong đó ai là phúc lợi của người dân thứ  $i$  ( $0 \leq a_i \leq 10^6$ ).

**Kết quả:** Ghi ra một số nguyên  $S$  là số tiền tối thiểu mà nhà vua cần bỏ ra để cân bằng phúc lợi cho tất cả các người dân.

EQU.INP	EQU.OUT
5 0 1 2 3 4	10
5 1 1 0 1 1	1
3 1 3 1	4
1 12	0

## Bài 3. Xâu LCM

Ta định nghĩa một phép nhân giữa một xâu  $a$  và một số nguyên dương  $x$  (kí hiệu là  $a, x$ ) là một xâu kết quả của việc viết  $x$  lần liên tiếp xâu  $a$ .

Ví dụ “abc”.  $2 = “abcabc”$ .

Xâu  $a$  chia hết cho xâu  $b$  nếu tồn tại số nguyên  $x$  sao cho  $b.x = a$ .

Ví dụ “abababab” chia hết cho “ab”, nhưng không chia hết cho “ababab” hoặc “aa”.

LCM của hai xâu  $s$  và  $t$  (kí hiệu là  $LCM(s, t)$ ) là xâu khác rỗng ngắn nhất chia hết cho cả  $s$  và  $t$ .

Cho hai xâu  $s$  và  $t$ , hãy tìm  $LCM(s, t)$  hoặc thông báo rằng nó không tồn tại. Có thể chứng minh được rằng nếu tồn tại  $LCM(s, t)$  thì nó là duy nhất.

**Dữ liệu:** Dòng đầu tiên chứa xâu s và dòng thứ hai chứa xâu t. Mỗi ký tự của xâu s, t là “a” hoặc “b” ( $1 \leq$  độ dài xâu s, độ dài xâu t  $\leq 10^3$ ).

**Kết quả:** Ghi ra  $LCM(s, t)$  nếu nó tồn tại, ngược lại ghi ra -1.

LCM.INP	LCM.OUT
baba	baba
ba	
aa	aaaaa
aaa	
aba	-1
ab	

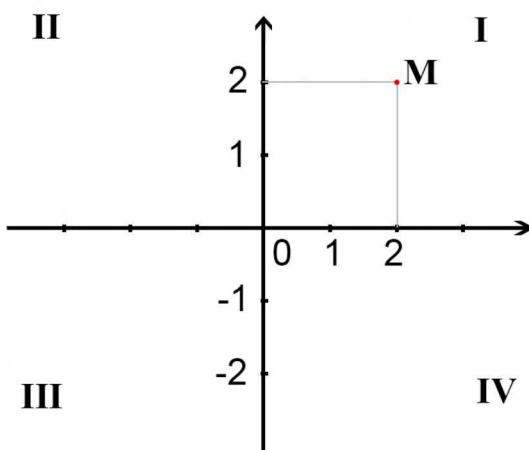


## ĐỀ THI HÀ TĨNH NĂM 2020

### Bài 1: Điểm tọa độ

Một Hệ tọa độ Descartes xác định vị trí của một điểm trên một mặt phẳng cho trước bằng một cặp số tọa độ  $(x, y)$ . Trong đó,  $x$  và  $y$  là 2 giá trị được xác định bởi hai đường thẳng có hướng vuông góc với nhau. Hai đường thẳng đó gọi là trục tọa độ, trục nằm ngang gọi là trục hoành, trục đứng gọi là trục tung, điểm giao nhau của 2 đường gọi là gốc tọa độ và nó có giá trị là  $(0, 0)$ . Hệ tọa độ đó được ký hiệu là Oxy.

Hệ tọa độ Oxy thường sẽ được chia làm bốn góc phần tư I, II, III và IV như hình dưới đây:



**Yêu cầu:** Viết chương trình xác định điểm  $M$  có tọa độ  $(x, y)$  được cho trong dữ liệu vào thuộc góc phần tư nào.

**Dữ liệu:** Vào từ file gồm một dòng duy nhất chứa hai số nguyên  $x$  và  $y$  là hoành độ và tung độ của điểm được cho ( $-10000 \leq x \leq 10000; x \neq 0; -10000 \leq y \leq 10000; y \neq 0$ ).

**Kết quả:** Một số La Mã I, II, III, IV là góc phần tư chứa điểm được cho trong dữ liệu vào.

Ví dụ:

INPUT	OUTPUT
2 2	I

## Bài 2: Số nhỏ nhất

Cho một dãy A gồm N số nguyên  $a_1, a_2, \dots, a_N$  đã được sắp xếp và Q truy vấn, mỗi truy vấn gồm ba số  $\ell, r, k$ .

**Yêu cầu:** Tìm số nhỏ nhất lớn hơn hoặc bằng  $k$  trong đoạn  $\ell$  đến  $r$ .

**Dữ liệu:** Vào từ file văn bản gồm:

+ Dòng đầu tiên gồm 2 số nguyên dương  $N, Q$  ( $1 \leq N \leq 10^5$ ,  $1 \leq Q \leq 10^3$ ).

+ Dòng tiếp theo gồm  $N$  số nguyên  $a_1, a_2, \dots, a_N$ , mỗi số có giá trị tuyệt đối không quá  $10^9$ .

+ Q dòng tiếp theo, mỗi dòng gồm 3 số nguyên thể hiện một truy vấn.

**Kết quả:** In ra Q dòng, mỗi dòng gồm một số nguyên để trả lời các truy vấn. Nếu không có kết quả thì in ra -1.

INPUT	OUTPUT
6 3	2
1 2 2 4 7 9	-1
1 3 2	4
1 4 5	
3 6 4	

## Bài 3: Dãy số

Cho dãy A gồm N số nguyên dương  $a_1, a_2, \dots, a_N$ . Hãy thực hiện các yêu cầu sau:

Tính đoạn con có trọng số lớn nhất. Trọng số của một đoạn con liên tiếp được tính bằng phần nguyên trung bình cộng của đoạn con đó.

Tìm số xuất hiện nhiều lần nhất trong dãy, nếu có nhiều đáp án thì lấy số nhỏ nhất.

**Dữ liệu:** Vào từ file văn bản gồm:

+ Dòng đầu gồm 1 số nguyên dương  $N$  ( $1 \leq N \leq 10^5$ ).

+ Dòng thứ hai gồm  $N$  số nguyên dương  $a_1, a_2, \dots, a_N$  ( $0 \leq a_i \leq 10^5$ ,  $1 \leq i \leq N$ ).

**Kết quả:**

+ Dòng 1 là trọng số lớn nhất của đoạn con.

+ Dòng 2 là số xuất hiện nhiều nhất trong dãy số.

INPUT	OUTPUT
5	4
2 3 3 4 4	3



## ĐỀ THI QUẢNG NINH NĂM 2020

### Bài 1: Xóa ký tự 0

Bạn được cho một xâu s gồm các ký tự 0 hoặc 1. Bạn muốn tất cả ký tự 1 trong xâu s liên tiếp nhau. Ví dụ nếu xâu là 0, 1, 00111 hoặc 011111100 thì tất cả các ký tự 1 liên tiếp nhau và nếu xâu là 0101, 100001 hoặc 1111111101 thì điều kiện này không được thỏa mãn.

Bạn có thể xóa một số (có thể bằng 0) tối thiểu ký tự 0 là bao nhiêu để tất cả ký tự 1 trong xâu s liên tiếp nhau?

**Dữ liệu:** Gồm một dòng chứa xâu s ( $1 \leq \text{độ dài của xâu s} \leq 10^5$ ), mỗi ký tự của xâu s là 0 hoặc 1.

**Kết quả:** Ghi ra một số nguyên là số tối thiểu ký tự 0 cần xóa để tất cả ký tự 1 trong xâu s liên tiếp nhau.

<i>Input</i>	<i>Output</i>
010011	2
1111000	0

### Bài 2. Tốt nhất

Thị trường bán lẻ đang là một lĩnh vực có tiềm năng lớn thu hút sự chú ý của nhiều nhà đầu tư. Tuy vậy, việc đảm bảo cho một siêu thị hay cửa hàng bán lẻ hoạt động có hiệu quả không phải là chuyện đơn giản. Lợi nhuận mỗi ngày của cửa hàng là tổng số tiền bán hàng thu được trong ngày, trừ tiền vốn mua hàng, tiền thuê mặt bằng, chi phí nhân công và điện nước, ...

Một siêu thị trong chi nhánh bán lẻ mở ở một địa điểm mới và đã hoạt động được  $n$  ngày, ngày thứ  $i$  ( $i = 1, 2, \dots, n$ ) có lợi nhuận là  $a_i$  ( $a_i$  có thể dương, âm hoặc bằng 0).

Giám đốc điều hành hệ thống bán lẻ muốn thu thập số liệu trong khoảng thời gian từ ngày  $i$  đến hết ngày  $j$  ( $1 \leq i \leq j \leq n$ ) có tổng lợi nhuận là lớn nhất và lớn hơn 0 để phân tích và rút kinh nghiệm chỉ đạo. Thông tin đầu tiên ông muốn biết là tổng lợi nhuận trong khoảng thời gian đó.

Hãy đưa ra tổng lợi nhuận tìm được hoặc một số 0 nếu không có ngày nào có lợi nhuận dương.

**Dữ liệu:** Dòng đầu tiên chứa số nguyên  $n$  ( $1 \leq n \leq 10^5$ ). Dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $|a_i| \leq 10^9$ ).

**Kết quả:** Ghi ra một số nguyên là tổng tìm được hoặc một số 0 nếu không có ngày nào có lợi nhuận dương.

Input	Output
5	13
12 -4 -10 4 9	

### Bài 3. Ước số

Bạn hãy tìm một số nguyên dương  $x$ , nếu biết danh sách tất cả các ước nguyên dương của nó trừ ước 1 và  $x$ .

Bạn cần trả lời cho  $t$  truy vấn độc lập.

**Dữ liệu:** Dòng đầu tiên chứa số nguyên  $t$  ( $1 \leq t \leq 25$ ) là số truy vấn. Các dòng tiếp theo mô tả  $t$  truy vấn, mỗi truy vấn gồm hai dòng: dòng thứ nhất chứa số nguyên  $n$  ( $1 \leq n \leq 300$ ) là số ước trong danh sách và dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $2 \leq a_i \leq 10^6$ ), ở đó  $a_i$  là ước thứ  $i$  của số cần tìm và tất cả các giá trị  $a_i$  là phân biệt.

**Kết quả:** Với mỗi truy vấn ghi ra câu trả lời trên một dòng. Nếu dữ liệu vào mâu thuẫn và không thể tìm thấy số  $x$  thì ghi ra -1, ngược lại ghi ra số  $x$  cần tìm.

Input	Output
3	48
8	-1
8 2 12 6 4 24 16 3	4
3	
2 3 4	
1	
2	



## ĐỀ THI VĨNH PHÚC NĂM 2020

### Bài 1. Chuyển đổi thời gian

*Bạn hãy lập chương trình chuyển đổi thời gian từ định dạng 12 giờ sang định dạng 24 giờ.*

**Lưu ý:** 12:00:00AM ở hệ 12 giờ được coi là 00:00:00 ở hệ 24 giờ và 12:00:00PM ở hệ 12 giờ được coi là 12:00:00 ở hệ 24 giờ.

**Dữ liệu vào:** từ tệp văn bản TimeCV.inp, một dòng duy nhất ghi thời gian ở hệ 12 giờ theo định dạng: hh:mm:ssAM hoặc hh:mm:ssPM, trong đó  $01 \leq hh \leq 12$ ;  $00 \leq mm, ss \leq 59$ .

**Dữ liệu ra:** In ra tệp văn bản TimeCV.out một dòng duy nhất ghi kết quả.

TimeCV.inp	TimeCV.out
07:05:45PM	19:05:45

### Bài 2 Số nguyên tố

Nam là học sinh đội tuyển học sinh giỏi Tin học của nhà trường, trong dịp nghỉ dịch COVID-19 thầy giáo giao cho Nam và các bạn hệ thống bài tập để ôn tập với rất nhiều dạng bài tập khác nhau. Trong đó có dạng bài tập về số nguyên tố, dạng bài tập này đã làm khó cho Nam. Nhưng Nam là một học sinh rất ham học hỏi từ các bạn của mình. Em hãy giúp Nam giải bài toán về số nguyên tố sau.

Số nguyên tố là số tự nhiên lớn hơn 1 chỉ có hai ước là 1 và chính nó.

**Yêu cầu:** Tìm số nguyên tố lớn nhất nhỏ hơn  $10^9$  xuất hiện trong xâu S.

**Dữ liệu vào:** Đọc từ file văn bản Prime.inp chỉ một dòng duy nhất chứa xâu S (không quá  $10^5$  ký tự).

**Dữ liệu ra:** Ghi ra file văn bản Prime.out chỉ một số duy nhất là kết quả phải tìm (Nếu không tìm thấy số nguyên tố nào ghi số -1).

Prime.inp	Prime.out
rgstf23756yfhf172ef122	37

### Bài 3. Kẹo

Thầy Việt Anh là một nhà giáo đầy nhiệt huyết và thầy thường đau đầu vì sự lười biếng của các học sinh trong lớp của mình, thầy rất muốn cải thiện tình trạng trên bằng các hình thức học mà chơi và có thường thông qua các câu đố hoặc trò chơi toán học. Lần này thầy viết một dãy gồm  $n$  số nguyên dương, mỗi số không vượt quá  $10^5$ .

Đưa dãy số đó cho các học sinh, thầy nói: "Các em có thể chọn 2 số bất kỳ trong dãy (gọi 2 số đó là A và B), chọn số nguyên tố X sao cho A chia hết cho X. Sau đó xóa A và thay nó bằng A/X, xóa B đi và thay nó bằng B.X (B nhân X). Các em có thể làm như vậy bao nhiêu lần cũng được, chừng nào còn thầy thích. Khi nào dừng, các em sẽ nhận được số kẹo bằng ước số chung lớn nhất của tất cả các số trong dãy."

Dĩ nhiên lớp của thầy Việt Anh rất thích kẹo và muốn làm sao để nhận được nhiều kẹo nhất từ thầy giáo của mình. Là học sinh đội tuyển Tin học, em hãy giúp thầy Việt Anh kiểm tra xem một học sinh có thể nhận được nhiều nhất bao nhiêu kẹo.

**Yêu cầu:** Cho  $n$  và dãy  $n$  số nguyên dương. Hãy xác định số kẹo tối đa mà một học sinh có thể nhận được.

**Dữ liệu vào:** từ file văn bản Candy.inp gồm 2 dòng:

- + Dòng đầu tiên chứa số nguyên dương  $n$  ( $n < 10^3$ ).
- + Dòng thứ 2 chứa  $n$  số nguyên dương.

**Dữ liệu ra;** Ghi ra file văn bản Candy.out gồm một số nguyên duy nhất là số kẹo tối đa.

Candy.inp	Candy.out
3 27 40 24	12



## ĐỀ THI ĐỒNG NAI NĂM 2020

### Bài 1: SỐ MAY MẮN

Để động viên thành tích học tập xuất sắc của các em học sinh lớp 6/3 trong năm học 2019 - 2020, thầy giáo chủ nhiệm đã chuẩn bị các món quà được đánh số từ 1 đến n. Sau đó thầy giáo sẽ cho các em lén bốc thăm để nhận món quà may mắn của mình.

Đầu tiên thầy giáo sẽ ghi tất cả số nguyên lẻ từ 1 đến n, sau đó sẽ ghi tất cả các số nguyên chẵn từ 2 đến n (theo thứ tự tăng dần) để tạo thành một dãy số phần thưởng.

Mỗi bạn sẽ bốc thăm một số k ứng với con số của món quà mình đạt được.

**Yêu cầu:** In số của món quà học sinh đạt được

**Dữ liệu vào:**

- Dòng duy nhất ghi số nguyên n và k ( $1 \leq k \leq n \leq 1000$ )

**Dữ liệu ra:**

In số của món quà học sinh đạt được.

SOMAYMAN.INP	SOMAYMAN.OUT
10 6	2

### Bài 2: SỐ THÂN THIẾT

Hai số nguyên dương được gọi là hai số thân thiết nếu chúng có cùng ước số nguyên tố lớn nhất.

**Yêu cầu:**

Hãy viết chương trình kiểm tra xem hai số nguyên dương có là hai số thân thiết hay không?

**Dữ liệu vào:**

Hai số nguyên dương a, b ( $1 < a, b < 10^9$ )

**Dữ liệu ra:**

- Dòng đầu tiên xuất ra ước nguyên tố lớn nhất của số a và ước nguyên tố lớn nhất của số b.

- Xuất ra “YES” nếu là 2 số thân thiết; xuất ra “NO” nếu không phải là hai số thân thiết.

SOTHANHTHIET.INP	SOTHANHTHIET.OUT
343 210	77 YES

### Bài 3: Tìm mã số

Một trò chơi tìm mã số được thiết kế sinh mã tự động. Chương trình cung cấp cho người chơi hai dãy ký tự, yêu cầu người chơi tìm ra mã khóa. Dãy mã số cần tìm theo quy tắc sau:

- + Gồm các ký tự số hai dãy ký tự;
- + Các ký tự số trong mã khóa chỉ xuất hiện duy nhất một lần;
- + Giá trị mã khóa nhận được là một số đạt giá trị lớn nhất.

#### Yêu cầu:

Hãy viết chương trình để tìm ra mã trò chơi, với hai dãy ký tự được nhập vào.

#### Dữ liệu vào:

Chứa dãy ký tự S1 và S2, mỗi dãy ký tự nằm trên một dòng.

#### Dữ liệu ra:

Là dãy ký tự số lớn nhất nhận được.

GAMES.INP	GAMES.OUT
190A1N23Y04 034A0S1N2	43210

### Bài 4: Số thứ k

Cho dãy số A gồm các số nguyên được sinh ra theo quy tắc sau: ban đầu dãy chỉ có đúng một phần tử bằng 1. Sau đó ta thực hiện  $(n - 1)$  bước. Mỗi bước ta lấy dãy số đó, nối vào sau chính nó và chèn vào chính giữa dãy mới một số nguyên dương nhỏ nhất chưa từng xuất hiện trong dãy.

#### Ví dụ:

Dãy ban đầu: 1

Sau bước 1: 1 2 1

Sau bước 2: 1 2 1 3 1 2 1

Sau bước 3: 1 2 1 3 1 2 1 4 1 2 1 3 1 2 1



**Yêu cầu:**

Tìm số hạng thứ k của dãy sau n – 1 bước như trên.

**Dữ liệu vào:**

Ghi 2 số nguyên dương n và k. ( $1 \leq n \leq 50$ ;  $1 \leq k \leq 2^n - 1$ )

**Dữ liệu ra:**

Số hạng thứ k của dãy.

<b>TIMSOK.INP</b>	<b>TIMSOK.OUT</b>
3 2	2
4 8	4

## ĐỀ THI THANH HÓA NĂM 2019

### Bài 1. Số nguyên

Nhập vào từ bàn phím số nguyên dương  $n$  ( $10 < N \leq 15000$ ), kiểm tra và in ra màn hình:

Số  $N$  có chia hết cho 7 hay không?

$N$  có bao nhiêu chữ số?

Tổng số các chữ số của  $N$  là bao nhiêu?

Số đảo ngược của  $N$ ?

**Ví dụ:** Nhập vào số  $N = 315$  thì có kết quả như sau:

Số 315 chia hết cho 7.

Số 315 có 3 chữ số.

Tổng các chữ số của 315 là 9.

Số đảo ngược của 315 là 513.

### Bài 2. Số Fibonacci

Fibonacci là dãy số kinh điển trong toán học được tìm thấy cách đây hơn 800 năm. Đến nay các nhà khoa học phát hiện nhiều trùng hợp thú vị về dãy số này trong tự nhiên. Dãy Fibonacci là dãy vô hạn các số tự nhiên bắt đầu bằng 1 và 1, sau đó các số tiếp theo sẽ bằng tổng của 2 số liền trước nó.

Cụ thể, các số đầu tiên của dãy Fibonacci là 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ...

Hãy lập trình thực hiện các công việc sau:

Đưa ra màn hình  $N$  số hạng đầu tiên của dãy Fibonacci, với  $N$  được nhập vào từ bàn phím ( $10 < N \leq 1000$ ).

Nhập vào từ bàn phím số  $M$ , đưa ra màn hình:

Số Fibonacci lớn nhất nhỏ hơn  $M$ .

Số Fibonacci bé nhất lớn hơn  $M$ .

**Ví dụ:** Nhập vào  $M = 100$  thì có kết quả như sau:

Số Fibonacci lớn nhất nhỏ hơn  $M$  là: 89

Số Fibonacci bé nhất lớn hơn  $M$  là: 144



### Bài 3. Phần tử yên ngựa

Cho mảng 2 chiều A có kích thước  $M \times N$  số nguyên. Phần tử  $A[i,j]$  được gọi là phần tử yên ngựa nếu nó là phần tử nhỏ nhất trong hàng  $i$  đồng thời là phần tử lớn nhất trong cột  $j$ .

Hãy lập chương trình tìm phần tử yên ngựa của mảng A.

**Dữ liệu vào:** ma trận A được nhập vào từ bàn phím với:  $M, N$  ( $0 \leq M, N \leq 100$ ) là kích thước của ma trận.

**Dữ liệu ra:** đưa ra màn hình vị trí của các phần tử yên ngựa (nếu có) hoặc dòng thông báo “Không có phần tử yên ngựa”.

Ví dụ:

Dữ liệu vào	Kết quả đưa ra màn hình
$M=3$ $N=3$ Các phần tử trong ma trận như sau: 15 3 9 55 4 6 76 1 2	(2,2)

Hoặc :

Dữ liệu vào	Kết quả đưa ra màn hình
$M=3$ $N=3$ Các phần tử trong ma trận như sau: 15 10 5 55 4 6 76 1 2	Không có phần tử yên ngựa

## ĐỀ THI ĐỒNG THÁP NĂM 2019

### Bài 1: Tổng hai phân số

*Tổng hai phân số  $\frac{a}{b}$  và  $\frac{c}{d}$  là một phân số  $\frac{x}{y}$ , x gọi là tử số và y gọi là mẫu số.*

**Yêu cầu:** Cho 4 số nguyên dương  $a, b, c, d$ , hãy tính tổng hai phân số  $\frac{a}{b}, \frac{c}{d}$  và rút gọn phân số được tính  $\frac{x}{y}$  sao cho phân số mới này tối giản. Phân số được gọi là tối giản nếu có tử số và mẫu số nguyên tố cùng nhau.

**Dữ liệu vào:** Một dòng chứa 4 số nguyên dương  $a, b, c, d$  giữa các số cách nhau một khoảng trắng.

**Kết quả:**

Dòng thứ nhất chứa số nguyên  $x$ .

Dòng thứ hai chứa số nguyên  $y$ .

Ví dụ:

INPUT	OUTPUT
4 3 7 3	11 3
4 5 10 4	33 10

### Bài 2: Đoạn con có tổng bằng 0

Cho một dãy số nguyên  $a$  gồm có  $N$  phần tử  $a_1, a_2, \dots, a_N$ . Một đoạn con liên tiếp của dãy  $a$  có điểm đầu  $L$  điểm cuối  $R$  với  $L \leq R$  là tập hợp tất cả các phần tử  $a_i$  với  $L \leq i \leq R$ .

**Yêu cầu:**

Đếm số lượng đoạn con có tổng các phần tử bằng 0.

**Dữ liệu vào:**

Dòng thứ nhất chứa số nguyên  $n$  ( $0 < n \leq 10^9$ ).

Dòng thứ hai chứa  $n$  số nguyên  $a_1, a_2, \dots, a_n$  ( $|a_i| \leq 10^9$ ;  $i = 1, \dots, n$ ).

**Kết quả:** Chỉ có một dòng chứa số nguyên là số lượng đoạn con có tổng bằng 0.



INPUT	OUTPUT	Giải thích
5 2 1 -1 2 0	4	Có 4 đoạn có tổng bằng 0 là: [2, 3], [1, 4], [1, 5], [5, 5]

### Bài 3: Tam giác Pascal

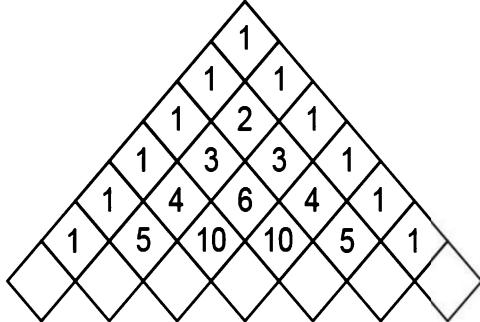
Mô hình tam giác Pascal là mô hình dùng để đưa ra các hệ số của triển khai nhị thức Newton bậc  $n$  của  $(x + 1)^n$ . Chẳng hạn: Trong triển khai  $(x + 1)^2 = x^2 + 2x + 1$  có các hệ số là 1, 2, 1 trong triển khai  $(x + 1)^3 = x^3 + 3x^2 + 3x + 1$  có các hệ số là 1, 3, 3, 1.

**Yêu cầu:** Hãy tìm các hệ số trong triển khai nhị thức Newton  $(x + 1)^n$ .

**Dữ liệu vào:** Chỉ có một dòng chứa số nguyên  $n$  ( $1 \leq n \leq 50$ ).

**Kết quả:** Một dòng ghi các số nguyên lần lượt là các hệ số trong triển khai nhị thức Newton  $(x + 1)^n$ , các số được ghi cách nhau một ký tự trắng.

Ví dụ:

INPUT	OUTPUT	Minh họa
5	1 5 10 10 5 1	

## ĐỀ THI BÌNH DƯƠNG NĂM 2021

### BÀI 1. PHÁT KHẨU TRANG

Kể từ khi được công bố là “Đại dịch toàn cầu” bởi WHO vào cuối tháng 1/2020, virus corona (COVID-19) đã có tác động không nhỏ đến tâm lý của một bộ phận người dân trong xã hội. Mỗi một nước đều có một cách phòng chống dịch bệnh khác nhau. Nước Thông Thái cũng có chính sách phòng chống dịch bệnh của riêng họ. Để đơn giản hóa, nhà của người dân được đặt trên một đường thẳng, đánh số từ 0 đến  $+\infty$ . Nước Thông Thái có rất nhiều nhà máy sản xuất khẩu trang. Với mỗi nhà máy tại vị trí  $x_i$  có thể cung cấp khẩu trang y tế cho một khu vực với bán kính  $r_i$ . Nhà nước muốn biết còn bao nhiêu điểm chưa được cung cấp khẩu trang y tế và bạn là một người được giao nhiệm vụ tính toán số địa điểm chưa được cung cấp khẩu trang y tế.

**Dữ liệu vào:**

Dòng đầu tiên chứa số nguyên dương  $n$  ( $n \leq 1000$ ) – Số địa điểm sản xuất.

$N$  dòng tiếp theo chứa hai cặp số  $x_i$  và  $r_i$  ( $(1 \leq x_i, r_i \leq 10^6)$ ).

**Dữ liệu ra:**

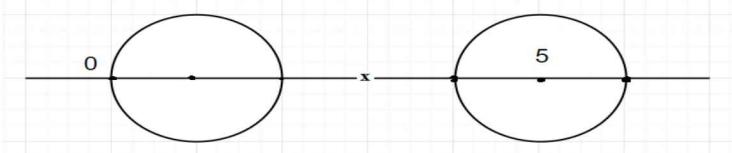
In một số nguyên dương – Tổng các địa điểm chưa được cung cấp khẩu trang từ vị trí 0 đến vị trí cung cấp khẩu trang cuối cùng.

**Ví dụ**

Dữ liệu vào	Dữ liệu ra
2	1
1 1	
5 1	
1	1
2 1	

*Giải thích:*

Ở ví dụ này, chỉ có địa điểm 3 chưa được cung cấp khẩu trang từ điểm 0 đến điểm 5.



## BÀI 2. CÁCH LY

Để ngăn chặn dịch Covid-19 đang diễn ra trên địa bàn, chính quyền cần phải cách ly những đối tượng mang bệnh và có nguy cơ nhiễm bệnh cao tại các khu cách ly tập trung. Các đối tượng có nguy cơ nhiễm bệnh cao được chia ra làm 2 nhóm. Nhóm F1 là các đối tượng tiếp xúc trực tiếp với người mang bệnh, F2 là đối tượng tiếp xúc trực tiếp với người thuộc nhóm F1.

### Dữ liệu đầu vào:

Dòng đầu tiên gồm  $N, M$  là số hàng và số cột. ( $1 \leq N, M \leq 100$ ).

$N$  dòng tiếp theo mỗi dòng có chính xác  $M$  ký tự biểu thị cho các đối tượng. Ký tự ('.') biểu thị cho người có nguy cơ nhiễm bệnh và ('\*') biểu thị cho người đang mang bệnh.

### Dữ liệu ra:

Một dòng duy nhất gồm: số lượng các đối tượng thuộc nhóm F1, số lượng các đối tượng thuộc nhóm F2 và số lượng người cần cách ly tại khu tập trung. (Cách nhau một khoảng trắng).

Ghi chú: 2 người được gọi là tiếp xúc nếu có cạnh chung trên ma trận.

### Ví dụ:

Dữ liệu vào:	Dữ liệu ra:
2 2 *. ..	2 1 4

Giải thích: 2 F1 và 1 F2 và 1 người bệnh.

## BÀI 3: TIẾT HỌC

Do đang trong thời gian cách ly nên trường X phải giảm bớt một số tiết học cho sinh viên sao cho thời gian sinh viên ở trường là ít nhất có thể.

Giả sử một tuần có  $n$  ngày, một ngày có  $m$  giờ học và mỗi tiết học chỉ kéo dài trong một giờ. Nếu trong một ngày sinh viên bắt đầu tiết học đầu tiên trong suốt giờ thứ  $i$ , và tiết học cuối cùng trong suốt giờ thứ  $j$ , thì ngày hôm đó sinh viên phải ở trường suốt  $j - i + 1$  giờ. Nếu ngày đó không có tiết học nào thì sinh viên chỉ dành 0 giờ ở trường.

Nhà trường chỉ có thể giảm nhiều nhất  $k$  tiết học trong một tuần, tìm ra số giờ ít nhất sinh viên phải ở trường trong một tuần.

### Dữ liệu vào

Dòng đầu tiên chứa 3 số nguyên  $n$ ,  $m$  và  $k$  ( $1 \leq n, m \leq 500$ ,  $0 \leq k \leq 500$ ) – số ngày trong một tuần, số giờ học trong một ngày, số tiết học nhà trường có thể giảm bớt.

$n$  dòng tiếp theo, mỗi dòng chứa một chuỗi nhị phân gồm  $m$  ký tự. Nếu ký tự thứ  $j$  của dòng thứ  $i$  là 1, thì có một tiết học trong suốt giờ thứ  $j$  của ngày thứ  $i$  (nếu nó là 0, thì không có tiết học nào trong giờ đó).

### Dữ liệu ra

In ra số giờ nhỏ nhất sinh viên phải ở trường trong một tuần.

Ví dụ:

Dữ liệu vào	Dữ liệu ra
2 5 1 01001 10110	5
2 5 0 01001 10110	8

## BÀI 4: VẮC XIN

Để ngăn chặn đại dịch Covid-19, các nhà khoa học hàng đầu thế giới đã tập trung lại với nhau để nghiên cứu các loại vắc-xin phòng bệnh. Các nhà khoa học được chia thành  $n$  nhóm.

Sau quá trình nghiên cứu vất vả, mỗi nhóm các nhà khoa học đưa ra  $n$  loại vắc-xin khác nhau, được đánh số theo thứ tự từ 1 đến  $n$ . Hiệu quả mỗi liều của loại vắc-xin thứ  $i$  là  $a_i$  và cần tốn chi phí  $c_i$  để sản xuất mỗi liều. Một liều sẽ được dùng để phòng bệnh cho một người duy nhất.

Do tình trạng dịch bệnh đang diễn ra trên toàn cầu, nguyên liệu sản xuất vắc-xin hạn chế nên loại vắc-xin thứ  $i$  chỉ sản xuất được tối đa  $d_i$  liều. Các nhà khoa học lúng túng khi đưa ra cách lựa chọn vắc-xin phòng bệnh sao cho hiệu quả là cao nhất với chi phí là thấp nhất.

Bạn hãy giúp các nhà khoa học đưa ra cách lựa chọn hiệu quả nhất.

### Dữ liệu vào:

Dòng đầu tiên gồm 2 số nguyên dương  $N, M$  ( $N, M \leq 10^5$ ).  $N$  là số nhóm và  $M$  là số người đang cần tiêm vắc-xin.

Dòng thứ 2 theo gồm  $N$  số nguyên dương, mỗi số nguyên  $a_i$  ( $a_i \leq 10^5$ ) là hiệu quả của vắc-xin do nhóm thứ  $i$  đưa ra.



Dòng thứ 3 gồm  $N$  số nguyên  $c_i$  dương, mỗi số nguyên  $c_i$  ( $c_i \leq 10^5$ ) là chi phí để sản xuất cho một liều vắc-xin thứ  $i$ .

Dòng thứ 4 gồm  $N$  số nguyên  $d_i$  dương, mỗi số nguyên  $d_i$  ( $d_i \leq 10^5$ ) là số liều tối đa có thể sản xuất của vắc-xin thứ  $i$ .

**Dữ liệu ra:**

Hai số nguyên  $x, y$  cách nhau một khoảng trắng. Trong đó  $x$  là tổng hiệu quả và  $y$  là tổng chi phí cần để phòng bệnh cho  $M$  người.

Nếu không đủ vắc xin để phòng bệnh, in ra -1.

Ví dụ:

Dữ liệu vào	Dữ liệu ra
4 3	10 10
5 2 1 3	
4 3 2 3	
1 2 2 1	
4 10	-1
8 11 2 3	
5 9 11 8	
2 2 2 3	

## ĐỀ THI GIA LAI NĂM 2019

### Bài 1. Số đảo ngược

Cho số nguyên dương  $X$ . Viết chương trình tìm số đảo ngược  $Y$  của  $X$ , biết  $Y$  gồm các chữ số của  $X$  và viết theo thứ tự ngược lại.

Ví dụ:

Nhập  $X$  là 356 thì kết quả  $X$  là 653.

### Bài 2. Phân số tối giản

Một chuỗi được gọi là có dạng phân số nếu có dạng như sau: “Tử\_số/Mẫu\_số”. Viết chương trình nhập vào chuỗi có dạng phân số, sau đó xuất ra dạng tối giản của phân số đó.

Ví dụ:

Chuỗi “12/15” biểu diễn cho phân số. Dạng tối giản của phân số đó là “3/5”.

### Bài 3. Tổng N chữ số

Khi viết các số nguyên dương tăng dần từ 1, 2, 3, ... liên tiếp nhau, ta nhận được một dãy các chữ số thập phân vô hạn, ví dụ: 123456789101112131415....

**Yêu cầu:** Hãy tính tổng  $n$  chữ số đầu tiên của dãy số vô hạn trên với  $n$  nhập từ bàn phím.

Ví dụ: Nhập  $n = 14$  thì trả về kết quả là 49.'

### Bài 4. Hình vuông đồng nhất

Cho một lưới ô vuông kích thước  $m \times n$ . Ô nằm trên giao của dòng  $i$  và cột  $j$  của lưới sẽ được gọi là ô  $(i, j)$  của lưới, người ta viết số nguyên không âm  $a_{ij}$ . Ta gọi hình vuông đồng nhất bậc 2 của lưới là tập gồm 4 ô nằm trên giao của hai dòng liên tiếp và 2 cột liên tiếp của lưới với các số viết trên chúng là như nhau.

**Yêu cầu:** Tính số lớn nhất các hình vuông đồng nhất bậc 2 chứa cùng một số.

**Dữ liệu vào:** được đặt trong file văn bản HINHVUONG.INP:

- + Dòng đầu tiên chứa các số nguyên dương  $m, n$  ( $m, n \leq 1000$ );
- + Dòng thứ  $i$  trong số  $m$  dòng tiếp theo chứa các số  $a_{i1}, a_{i2}, \dots, a_{in}$ ,  $i = 1, 2, \dots$ , hai số liên tiếp trên dòng được viết cách nhau một dấu cách.  $0 \leq a_{ij} \leq 255$ ,  $i, j = 1, 2, \dots$



**Kết quả ra:** đặt trong file văn bản HINHVUONG.OUT: số lớn nhất các hình vuông đồng nhất bậc 2 chứa cùng một số.

Ví dụ:

<i>Input</i>	<i>Output</i>
5 10 0 1 1 0 2 2 0 5 5 0 0 1 1 0 2 2 0 5 5 0 0 0 0 1 1 0 0 0 0 0 0 8 0 1 1 1 1 0 9 0 0 0 0 0 0 1 1 0 0 0	3

## PHỤ LỤC

# Thông tin mở rộng

Trong phần phụ lục sẽ giới thiệu thêm một số cuộc thi lập trình trong nước và quốc tế dành cho học sinh Trung học cơ sở; đồng thời giới thiệu gợi mở một số kiến thức, xu hướng và cộng đồng lập trình phồn thịnh trên thế giới.

## Các cuộc thi lập trình trên thế giới

### 1. American Computer Science League (ACSL)

**American Computer Science League (ACSL)** là cuộc thi lập trình máy tính dành cho các học sinh từ mầm non tới lớp 12. Năm 2020, cuộc thi diễn ra với hơn 500 đội tham gia dự thi tại Hoa Kỳ, Canada, Châu Âu và Châu Á.

Bài thi tại ACSL được phân chia theo từng khối lớp, phù hợp cho học sinh ở các độ tuổi và khả năng khác nhau.

Với thí sinh cấp trung học cơ sở, cuộc thi cho phép thí sinh sử dụng ngôn ngữ lập trình Python, C ++ hoặc Java để giải đề thi.

Để đăng ký tham gia dự thi tại Việt Nam, bạn cần liên hệ với đối tác của đơn vị tổ chức là Viện Nghiên cứu Giáo dục Sáng tạo.- RICE.

Để biết thêm thông tin về cuộc thi, bạn có thể tham khảo tại fanpage của RICE tại địa chỉ: <https://www.facebook.com/riceeducationvietnam>.

Ví dụ một bài mẫu trong đề thi năm 2020 của cuộc thi:

Given a positive integer (call it N), a position in that integer (call it P), and a transition integer (call it D). Transform N as follows:

- If the P th digit of N from the right is from 0 to 4, add D to it. Replace the P th digit by the units digit of the sum. Then, replace all digits to the right of the P th digit by 0.
- If the P th digit of N from the right is from 5 to 9, subtract D from it. Replace the P th digit by the leftmost digit of the absolute value of the difference. Then,



replace all digits to the right of the P th digit by 0.

**Example 1:** . The 2nd N = 7145032, P = 2, D = 8 digit from the right is 3; add 8 to it (3+8=11), and replace the 3 with 1 to get 7145012. Replace the digits to the right by 0s to get 7145010.

**Example 2:** . The 3rd N = 1540670, P = 3, D = 54 digit from the right is 6; the absolute value of 6-54 is 48; replace with the 4 to get 1540470. Replace the digits to the right with 0s to get 1540400.

**INPUT:** There will be 5 sets of data. Each set contains 3 positive integers: N, P, and D. N will be less than 10 ; P and D will be valid inputs. No input will cause an output to have a leading digit 15 of 0.

**OUTPUT:** Print the transformed number. The printed number may not have any spaces between the digits.

**SAMPLE INPUT:**

124987 2 3

540670 3 9

7145042 2 8

124987 2 523

4386709 1 2

**SAMPLE OUTPUT:**

1. 124950

2. 540300

3. 7145020

4. 124950

5. 4386707

## 2. Olympic Khoa học quốc tế HKICO (HongKong International Computational Olympiad)

Kỳ thi Olympic Tin học Quốc tế Hồng Kông được thành lập và tổ chức bởi Trung tâm Giáo dục Olympiad Champion Hong Kong (Olympiad Champion Education Centre from Hong Kong), một trung tâm giáo dục đã đăng ký hoạt động tại Bộ Giáo dục Hồng Kông.

Ngay trong lần tổ chức đầu tiên vào năm 2020 với điểm đến lý tưởng được lựa chọn là Đại học Quốc gia Singapore (NUS), Ban Tổ chức quốc tế HKICO (viết tắt HKICO 2020) đã nhận được sự quan tâm từ trên 14 quốc gia và vùng lãnh thổ trên thế giới trong đó có Việt Nam.

HKICO 2020 thu hút hơn hàng trăm ngàn thí sinh tham gia Vòng loại và Vòng Chung kết quốc gia tại các quốc gia và vùng lãnh thổ; những thí sinh xuất sắc nhất vượt qua các vòng thi cấp quốc gia sẽ tham dự Vòng Chung kết quốc tế vào tháng 09/2020 tại Đảo quốc xinh đẹp Singapore

Cuộc thi dành cho học sinh từ lớp 2 tới lớp 12. Tùy theo từng khối sẽ có hình thức và nội dung thi khác nhau. Cụ thể:

Khối	Ngôn ngữ lập trình	Ngôn ngữ đề thi
2, 3, 4	SCRATCH	Tiếng Anh
5, 6, 7	BLOCKLY	Tiếng Anh
8, 9, 10, 11, 12	PYTHON JAVA C++	Tiếng Anh

Để biết thêm thông tin chi tiết và liên tục cập nhật về Kỳ thi, tham khảo tại fanpage chính thức <https://www.facebook.com/HKICOVietnam>.

## 4. PLU Middle/High School Programming Contest (tại Mỹ)

Cuộc thi dành cho học sinh tại các trường trung học cơ sở và trung học phổ thông trên khắp tiểu bang Washington (Mỹ).

Cuộc thi kéo dài ba giờ, trong thời gian đó, mỗi đội cần gắng giải quyết một số (thường là sáu mươi) bài lập trình máy tính bằng bất kỳ ngôn ngữ lập trình nào (ví dụ: Java, Python 3.x, C ++).

Cuộc thi có phí đăng ký 15\$ đối với cá nhân và 30\$ đối với nhóm thi.



Ví dụ một bài lập trình trong kỳ thi dành cho khối trung học cơ sở (đề nguyên bản sử dụng ngôn ngữ Tiếng Anh):

Cha của Pi, Danny, điêu hành Rạp xiếc Hackerville. Anh ấy sẽ chuyển đến Rookie Ville và anh ta muốn đưa tất cả các con vật trong rạp xiếc đi cùng thông qua tàu hỏa. Anh ấy đang gấp rắc rối về cách sắp xếp chúng vì một số loài không thể sắp xếp chung 1 cabin.

Có  $N$  con vật xếp thành 1 hàng. Mỗi con vật được đánh số theo thứ tự từ 1 tới  $N$ . Có  $m$  cặp ( $a[i]$ ,  $b[i]$ ) trong đó con vật  $a[i]$  và  $b[i]$  là kẻ thù và không nên giữ trong cùng một cabin. Pi rất giỏi vẫn đề này và anh đã đưa ra thử thách: đếm số lượng các nhóm khác nhau sao cho trong nhóm không chứa bất kỳ cặp nào là kẻ thù. Một nhóm được xác định như một khoảng  $(x, y)$  sao cho tất cả các động vật trong phạm vi từ  $x$  đến  $y$  tạo thành một nhóm. Xác định số lượng nhóm có thể được hình thành theo giải pháp của Pi.

Ví dụ: cho  $N = 3$  con vật và  $m = 3$  cặp kẻ thù,  $a = [1, 2, 3]$  và  $b = [3, 3, 1]$ , con vật 1 là kẻ thù của con vật 3 và con vật 3 là kẻ thù của con vật 1 và 2. Vì 3 là kẻ thù của cả 1 và 2, nên nó phải ở trong cabin của chính nó. Động vật 1 và 2 có thể được ở cùng phòng hoặc riêng biệt. Vậy có bốn nhóm có thể đáp ứng các ràng buộc:  $\{1, 2\}$ ,  $\{1\}$ ,  $\{2\}$ ,  $\{3\}$ . Lưu ý rằng thứ tự ban đầu của các con vật luôn được đánh số liên tiếp từ 1 đến  $n$ , như trong ví dụ này là  $[1, 2, 3]$  và số thứ tự sẽ không được sắp xếp lại.

Yêu cầu: Tìm ra số lượng nhóm có thể hình thành theo giải pháp của Pi.

#### Dữ liệu vào:

- Dòng đầu tiên là giá trị  $N$ : số lượng con vật cần chia nhóm.
- Dòng thứ 2 chứa giá trị  $m$  là số lượng con vật trong  $a$ .
- M dòng tiếp theo, mỗi dòng chứa 1 số nguyên là các giá trị trong  $a$ .
- Dòng tiếp theo sau  $m$  dòng là giá trị  $m$  số lượng con vật trong  $b$ .
- M dòng tiếp theo, mỗi dòng chứa 1 số nguyên là các giá trị trong  $b$ .

INPUT	OUTPUT
4	7
2	
1	
2	
2	
3	
4	

## Giới thiệu một số website luyện thi trực tuyến

### 1. Online Judge



<https://onlinejudge.org/>

Online Judge (hay còn gọi là UVa Online Judge) là một website luyện thi trực tuyến do Đại học Valladolid (Tây Ban Nha) sáng lập, hiện đang được phát triển và duy trì bởi Đại học Baylor (Hoa Kỳ). Với kho lưu trữ có hơn 4300 bài toán và bất kỳ ai đăng ký tài khoản đều có thể tham gia làm bài, nộp bài trực tuyến.

Cùng với đó là hệ thống chấm bài tự động, bạn có thể biết ngay kết quả bài thi đúng hay sai sau khi gửi bài.

Bạn có thể lựa chọn và lập trình bằng nhiều ngôn ngữ lập trình khác nhau, ví dụ như: C/C++, Pascal, Java, Python, Swift, ...

Ngoài ra, UVa còn là đơn vị tổ chức cuộc thi lập trình ACM-ICPC – Cuộc thi lập trình quốc tế dành cho sinh viên trên toàn thế giới. Cuộc thi đã thu hút được sự tham gia của rất nhiều thế hệ sinh viên Việt Nam từ năm 2005 tới nay.

### 2. Sphere Online Judge



<https://www.spoj.com/>

Sphere Online Judge (SPOJ) là một website luyện thi trực tuyến được điều hành bởi công ty Sphere Research Labs của Ba Lan. Khác với Online Judge, các bài toán trên website được tải lên bởi cộng đồng người dùng hoặc được lấy từ các cuộc thi lập trình trước đó. Bởi vậy, SPOJ có kho lưu trữ hơn 20.000 bài toán với nhiều loại ngôn ngữ khác nhau (có cả tiếng Việt).

Cũng giống với UVa Online Judge, SPOJ cũng có hệ thống chấm bài tự động cũng như lựa chọn giải bài bằng nhiều ngôn ngữ khác nhau.



## Giới thiệu cộng đồng lập trình viên trên thế giới và kho mã nguồn

### 1. Github



**GitHub** là một kho lưu trữ mã nguồn dựa trên nền tảng web, các mã nguồn này thường được ứng dụng cho các dự án như phần mềm, ứng dụng, .... Bạn có thể truy cập và xem các tệp mã nguồn mở trên website một cách tự do và hoàn toàn miễn phí. Tính đến tháng 10 năm 2021, GitHub có hơn 65 triệu người sử dụng là các lập trình viên trên toàn thế giới với hơn 200 triệu kho mã nguồn, điều này khiến GitHub trở thành máy chủ chứa mã nguồn lớn trên thế giới.

Github đã trở thành một yếu tố có sức ảnh hưởng trong cộng đồng phát triển mã nguồn mở. Thậm chí nhiều đơn vị tuyển dụng thay vì yêu cầu ứng viên cung cấp sơ yếu lý lịch, họ chuyển thành yêu cầu ứng viên cung cấp một liên kết đến tài khoản Github để đánh giá ứng viên.

Vào năm 2018, Microsoft đã mua lại GitHub với giá 7,5 tỷ Đô la Mỹ.

### 2. Stackoverflow



**Stack Overflow** là một website hỏi đáp dành cho các lập trình viên chuyên nghiệp và đam mê. Stack Overflow chỉ chấp nhận các câu hỏi về lập trình và tập trung vào một vấn đề cụ thể. Bởi vậy khi gặp một vấn đề tương tự vẫn đề đã có, bạn hoàn toàn có thể tìm kiếm các câu trả lời hoặc giải pháp trên website. Tính đến tháng 3 năm 2021, Stack Overflow có hơn 14 triệu người dùng đã đăng ký, hơn 21 triệu câu hỏi và 31 triệu câu trả lời.

Stack Overflow đã giành được Giải thưởng 2020 Webby People's Voice Award for Community trong danh mục Web - Giải thưởng Webby là giải thưởng quốc tế hàng đầu tôn vinh sự xuất sắc trên Internet.

## Mã hóa, BlockChain và Tiền Ảo

Như chúng ta đã biết, tài sản của con người có thể là tiền, vàng, bất động sản,... Tuy nhiên với sự bùng nổ của công nghệ và nền tảng kết nối mạnh mẽ trên toàn thế giới, giờ đây đã xuất hiện những thứ còn giá trị hơn cả vàng và tiền đó là tiền ảo (như Bitcoin, Ethereum,...) hay tài sản kỹ thuật số được chứng thực (NFT),... Tất cả những thứ tưởng như chỉ xuất hiện trong phim khoa học viễn tưởng đang là xu hướng và làm thay đổi cả nền tài chính thế giới, những “tài sản” công nghệ đó được xây dựng dựa trên các thuật toán SHA256, RSA,... Trong phần này chúng ta sẽ cùng tìm hiểu về công nghệ và xu hướng này với Python.

## 1. Mã hóa và các thuật toán mã hóa

Ngày nay, hầu hết chúng ta đều sử dụng các ứng dụng hoặc dịch vụ cần có tính bảo mật cao, ví dụ như thực hiện các giao dịch ngân hàng, chữ ký số cho các doanh nghiệp, ví điện tử, ... Các ứng dụng, dịch vụ này đều lưu trữ các thông tin vô cùng quan trọng. Bởi vậy, các ứng dụng, dịch vụ đều cần phải có những giải pháp để đảm bảo tính bảo mật về lưu trữ thông tin cũng như tài sản của người dùng.

Các ứng dụng, website hiện nay hầu hết đều sử dụng phương pháp mã hóa dữ liệu trước khi lưu thông tin vào cơ sở dữ liệu. Mã hóa là quá trình dùng để biến thông tin từ dạng này sang dạng khác không giống ban đầu.



## Các phương pháp mã hóa thường dùng

#### a. Mã hóa một chiều (hash)

Phương pháp mã hóa một chiều dùng để mã hóa những thông tin không cần dịch lại. Ví dụ, khi bạn đăng nhập vào website hoặc ứng dụng, mật khẩu mà bạn nhập sẽ được chuyển thành một chuỗi dài các ký tự bằng một hàm gọi là hash function (tạm dịch: hàm băm).

Chuỗi sau khi chuyển đổi (hay còn gọi là chuỗi sau khi “băm”) sẽ được lưu vào cơ sở dữ liệu, mật khẩu ban đầu của bạn sẽ không được lưu lại nhằm tăng tính bảo mật. Trường hợp không may hacker truy cập được vào cơ sở dữ liệu thì cũng chỉ thấy những chuỗi đã được mã hóa mà không phải mật khẩu của bạn ví dụ như chuỗi: FJI2OAYSDH37XYB57G51JS7Z.

Mỗi khi bạn đăng nhập, hash function sẽ chuyển password bạn mới nhập thành chuỗi ký tự rồi so sánh với chuỗi đã lưu trong cơ sở dữ liệu, nếu khớp thì cho phép đăng nhập, không thì báo lỗi.



Hash function có nhiệm vụ chuyển một chuỗi có độ dài bất kỳ thành chuỗi ký tự có độ dài cố định. Ví dụ, nếu bạn quy định chuỗi ký tự sau khi chuyển đổi sẽ dài 10 ký tự thì dù đầu vào của bạn có bao nhiêu ký tự, kết quả nhận được sẽ luôn là 10 ký tự.

Đặc điểm của hash function là trong cùng 1 điều kiện, dữ liệu đầu vào như nhau thì kết quả sau khi băm cũng sẽ y hệt như nhau. Nếu chỉ đổi một chút, có khi chỉ là 1 ký tự nhỏ thì chuỗi kết quả sẽ khác hoàn toàn.

Cũng vì vậy, ta cũng có thể dùng hash function để kiểm tra tính toàn vẹn của dữ liệu. Ví dụ, trước khi gửi một tập tin văn bản (word) cho bạn bè hoặc đối tác, bạn dùng hàm hash mã hóa một chiều và tạo ra được chuỗi sau băm là DFYUBUfyefuefu. Khi bạn hoặc đối tác của bạn tải tập tin về máy, nếu “băm” tập tin và cũng nhận được chuỗi DFYUBUfyefuefu, có nghĩa tập tin của bạn còn nguyên vẹn, không bị mất hoặc không bị can thiệp và thay đổi nội dung bởi hacker, còn nếu kết quả khác, nghĩa là quá trình truyền tải có thể đã bị lỗi làm mất một phần dữ liệu, hoặc tệ hơn là có ai đó đã xén bớt hay thêm nội dung khác vào. Đặc biệt với các tập tin cài đặt, nếu bạn không tải bộ cài từ trang chủ của nhà phát hành, bạn sẽ cần chú ý tránh trường hợp hacker thêm vào bộ cài những nội dung nhầm mục đích theo dõi, đánh cắp thông tin hoặc có chứa virus...

Hiện nay, hai thuật toán hash function thường được sử dụng là MD5 và SHA. Nếu bạn tải tập tin trên mạng thì đôi khi sẽ thấy dòng chữ MD5 do tác giả cung cấp, mục đích là để bạn so sánh file đã tải về với file gốc xem có bị thay đổi hoặc lỗi trong quá trình tải về hay không.

Để hiểu thêm về hai thuật toán này, bạn có thể tìm kiếm trên Internet với từ khóa: mã hóa MD5, mã hóa SHA, ...

Ví dụ mã hóa MD5 và SHA với thư viện hashlib trong Python:

```
from hashlib import *

file = open("1.PNG", "rb")

data = file.read()
md5 = md5()
sha1 = sha1()
sha224 = sha224()
sha256 = sha256()
sha384 = sha384()

md5.update(data)
sha1.update(data)
sha224.update(data)
sha256.update(data)
sha384.update(data)

print('{}: {}'.format(md5.name, md5.hexdigest()))
print('{}: {}'.format(sha1.name, sha1.hexdigest()))
print('{}: {}'.format(sha224.name, sha224.hexdigest()))
```

```
print('{}: {}'.format(sha256.name, sha256.hexdigest()))
print('{}: {}'.format(sha384.name, sha384.hexdigest()))
```

### b. Mã hóa đối xứng (symmetric key encryption)

Khác với mã hóa một chiều, mã hóa đối xứng là phương pháp mã hóa có thể giải mã để lấy thông tin ban đầu. Để mã hóa và giải mã, ta cần phải có “key” – chìa khóa.

Ở phương pháp mã hóa đối xứng, chìa khóa để mã hóa và giải mã giống nhau nên ta gọi là đối xứng (tiếng Anh là symmetric). Theo một số tài liệu thì mã hóa đối xứng là giải pháp được sử dụng nhất phổ biến hiện nay.

Hai thuật toán mã hóa đối xứng thường được sử dụng gồm DES (Data Encryption Standard) và AES (Advanced Encryption Standard).

- DES được FIPS (Tiêu chuẩn Xử lý Thông tin Liên bang Hoa Kỳ) chọn làm chuẩn mã hóa vào năm 1976 và được sử dụng rộng rãi. Tuy nhiên, với công nghệ hiện nay, DES được xem là không đủ an toàn cho nhiều ứng dụng. Nguyên nhân do độ dài 56 bit của khóa là quá nhỏ. Khóa DES đã từng bị phá trong khoảng thời gian chưa tới 24 giờ.
- AES ngày nay được sử dụng phổ biến hơn và thay thế cho DES. Hiện nay nhiều cơ quan chính phủ trên thế giới quy định tài liệu khi được gửi qua mạng phải mã hóa AES. Thuật toán AES có thể dùng nhiều kích thước ô nhớ khác nhau để mã hóa dữ liệu, thường thấy là 128-bit và 256-bit, có một số lên tới 512-bit và 1024-bit. Kích thước ô nhớ càng lớn thì càng khó phá mã hơn, cùng với đó việc giải mã và mã hóa cũng cần nhiều năng lực xử lý hơn.

Để hiểu thêm về hai thuật toán này, bạn có thể tìm kiếm trên Internet với từ khóa: mã hóa DES, mã hóa AES, ...

Ví dụ mã hóa đối xứng trong Python:

Trước tiên, ta cần cài đặt thư viện PyCryptodome bằng cách mở cửa sổ Terminal và gõ cú pháp: `pip install PyCryptodome`

```
from Crypto.Cipher import AES
import binascii, os

#Mã hóa thông tin
def encrypt_AES_GCM(msg, secretKey):
    aesCipher = AES.new(secretKey, AES.MODE_GCM)
    ciphertext, authTag = aesCipher.encrypt_and_digest(msg)
    return (ciphertext, aesCipher.nonce, authTag)

#Giải mã thông tin
def decrypt_AES_GCM(encryptedMsg, secretKey):
    (ciphertext, nonce, authTag) = encryptedMsg
    aesCipher = AES.new(secretKey, AES.MODE_GCM, nonce)
    plaintext = aesCipher.decrypt_and_verify(ciphertext, authTag)
    return plaintext
```



```

secretKey = os.urandom(32) # khóa mã hóa ngẫu nhiên 256
print("Encryption key:", binascii.hexlify(secretKey))
msg = b'Thong diep can ma hoa'
encryptedMsg = encrypt_AES_GCM(msg, secretKey)

print('ciphertext: ', binascii.hexlify(encryptedMsg[0]))

decryptedMsg = decrypt_AES_GCM(encryptedMsg, secretKey)
print("decryptedMsg: ", decryptedMsg)

```

### c. Mã hóa bắt đối xứng (public key encryption)

Cũng giống với mã hóa đối xứng, mã hóa bắt đối xứng ta cũng có thể giải mã dữ liệu để lấy thông tin ban đầu, nhưng khóa mã hóa và khóa giải mã hoàn toàn khác nhau. Để phân biệt giữa hai khóa, người ta gọi khóa mã hóa là public key, khóa giải mã là private key. Public key là khóa có thể được sử dụng để mã hóa dữ liệu bởi bất kì ai. Tuy nhiên, chỉ người có private key mới giải mã được dữ liệu.

Quy trình mã hóa bắt đối xứng như sau:

- + Bên A tạo ra một cặp public và private key. Sau đó lưu lại private key và chuyển public key chuyển cho bên B (dưới hình thức email, copy qua USB, ...).
- + Bên B sử dụng public key để mã hóa dữ liệu, sau đó gửi file đã mã hóa cho bên A.
- + Bên A dùng private key đã lưu trước đó để giải mã dữ liệu và sử dụng.

Nhược điểm của mã hóa bắt đối xứng là tốc độ giải mã chậm hơn so với mã hóa đối xứng, chúng ta cần CPU có nhiều năng lực xử lý hơn, phải chờ lâu hơn.

Chính vì thế, thay vì phải mã hóa cả một file bằng phương pháp bắt đối xứng., chúng ta có thể sử dụng phương pháp bắt đối xứng để **mã hóa key** trong phương pháp **mã hóa đối xứng**.

Thuật toán mã hóa bắt đối xứng thường được sử dụng là RSA. Bạn có thể tìm hiểu thêm về thuật toán bằng cách tìm kiếm trên Internet với từ khóa: RSA.

Ví dụ mã hóa RSA trong Python:

```

from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii

keyPair = RSA.generate(3072)

pubKey = keyPair.publickey()
print(f"Public key: (n={hex(pubKey.n)}, e={hex(pubKey.e)})")
pubKeyPEM = pubKey.exportKey()
print(pubKeyPEM.decode('ascii'))

print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")

```

```
privKeyPEM = keyPair.exportKey()
print(privKeyPEM.decode('ascii'))

# Mã hóa
msg = b'Thong tin can ma hoa'
encryptor = PKCS1_OAEP.new(pubKey)
encrypted = encryptor.encrypt(msg)
print("Encrypted:", binascii.hexlify(encrypted))

# Giải mã
decryptor = PKCS1_OAEP.new(keyPair)
decrypted = decryptor.decrypt(encrypted)
print('Decrypted:', decrypted)
```

## 2. Công nghệ Blockchain

Blockchain là một công nghệ lưu trữ dữ liệu. Đơn vị lưu trữ dữ liệu cơ bản của blockchain là một khối dữ liệu (block). Dữ liệu sẽ được đóng gói thành một khối sau đó khóa lại bằng thuật toán mã hóa. Khối dữ liệu sau khi được khóa lại sẽ không thể thay đổi và tồn tại mãi mãi. Các khối sau khi được tạo ra sẽ liên kết với nhau thành một chuỗi khối (blockchain).

Công nghệ blockchain kết hợp thuật toán hash (băm) cùng mã hóa bắt đầu xứng RSA để đảm bảo dữ liệu an toàn và không thể thay đổi.

Cấu trúc một khối blockchain cơ bản gồm có:

- + Mã băm: Giống một mã định danh cho khối và mỗi khối có một mã khác nhau.
- + Dữ liệu lưu trữ: Thông tin được lưu trữ trong khối.
- + Mốc thời gian: Thời gian khối được tạo ra.
- + Mã băm khối trước: Mã băm của khối đứng trước.

Trong trường hợp một khối bị sửa đổi thông tin (hack), khi đó mã băm của khối đó sẽ bị thay đổi khiến mắt liên kết với các khối phía sau. Nếu muốn liên kết lại sẽ phải thay đổi tất cả các khối phía sau đó. Việc chuyển đổi đó sẽ phụ thuộc độ dài block và cấu hình máy tính. Nhưng tất cả các máy tính tham gia vào hệ thống đều có một bản sao của chuỗi các khối hợp lệ và nếu muốn chuỗi khối đã bị thay đổi nội dung được chấp nhận sẽ cần có sự đồng thuận của 51% các máy trong hệ thống nhờ cơ chế đồng thuận phi tập trung. Vì vậy nếu chỉ sửa đổi trên một máy sẽ không được chấp nhận.

Hiện nay có rất nhiều ứng dụng được xây dựng dựa trên nền tảng công nghệ blockchain như tiền điện tử (Bitcoin, Ethereum, ...); ứng dụng tài chính, hợp đồng thông minh; ứng dụng điều hành, giám sát (y tế, giáo dục, thương mại,...).



### 3. Thuật toán đào Bitcoin

#### a. Giới thiệu về Bitcoin

Bitcoin là một loại tiền mã hóa, được phát minh bởi Satoshi Nakamoto dưới dạng phần mềm mã nguồn mở từ năm 2009. Bitcoin có thể được trao đổi trực tiếp bằng thiết bị kết nối Internet mà không cần thông qua một tổ chức tài chính trung gian nào. Hiện nay 1 Bitcoin có giá trị khoảng hơn 1,3 tỷ VNĐ (Tháng 10 năm 2021).

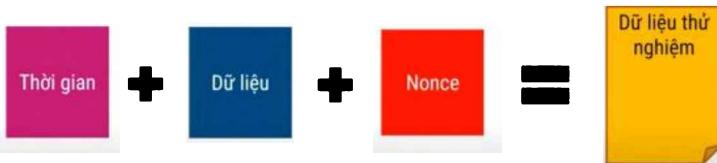
Bitcoin có cách hoạt động khác hẳn so với các loại tiền tệ điển hình: không có một ngân hàng trung ương nào quản lý nó và hệ thống hoạt động dựa trên một giao thức mạng ngang hàng trên Internet. Sự cung ứng Bitcoin là tự động, hạn chế, được phân chia theo lịch trình định sẵn dựa trên các thuật toán. Bitcoin được cấp tới các máy tính "đào" Bitcoin để trả công cho việc xác minh giao dịch Bitcoin và ghi chúng vào cuốn sổ cái được phân tán trong mạng ngang hàng, thông qua công nghệ Blockchain.

Tổng số lượng Bitcoin được phát hành trên toàn thế giới là 21 triệu đơn vị. Khi đã khai thác hết sẽ không tạo ra thêm. Chính vì vậy Bitcoin trở nên khan hiếm và có giá trị.

#### b. Thuật toán đào Bitcoin

Khi đào Bitcoin, máy tính sẽ ghép nhiều thông tin khác nhau thành một chuỗi dữ liệu, tuy nhiên các thông tin cơ bản gồm có:

- + Thời gian: Thời gian thực hiện giao dịch
- + Dữ liệu: Thông tin giao dịch
- + Nonce: Số nguyên dương ngẫu nhiên trong khoảng từ 0 tới 4294967295.



Các thông tin được nối thành một chuỗi ta gọi là “Dữ liệu thử nghiệm”. Ứng dụng thực hiện băm dữ liệu thử nghiệm thành mã băm và kiểm tra các thông tin trong mã băm gồm:

- + Các ký tự bắt đầu của mã băm phải nhỏ hơn hoặc bằng một giá trị mục tiêu nhất định. Hay nói cách khác, các giá trị bắt đầu trong mã băm cần là một loạt các số 0. Số lượng các số 0 cần có ta gọi là độ khó (ví dụ nếu cần 8 số 0 ở đầu, độ khó là 8).
- + Độ khó thay đổi theo thời gian và xác suất để hàm băm tạo ra mã băm gồm một loạt các số 0 là rất thấp.

Ví dụ:

Độ khó = 8

Nonce = 1

Mục tiêu: 000000005

Mã băm:

C60CD1994034D4BFB61C3660A544C3B0BA6798BD2DC0A7B5905CCD8AEAABEDD0D

Khi so sánh 8 ký tự đầu của mã băm trong trường hợp này không phải là 8 ký tự 0. Vậy mã này là không hợp lệ, máy tính sẽ tăng giá trịNonce lên 2, 3, 4, .... đến khi tìm được mã băm hợp lệ.

Ví dụ:

Độ khó = 8

Nonce = 5111980

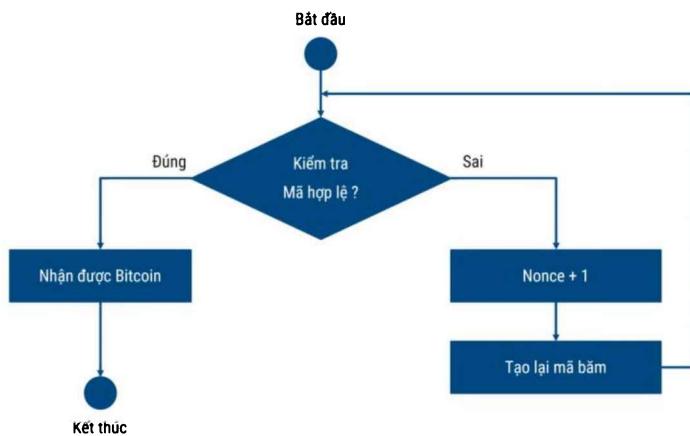
Mục tiêu: 000000005

Mã băm:

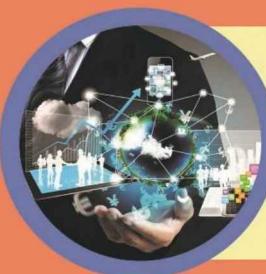
00000000469D910C18251C0F7361BB3613CBDB40F38F39B662355C51624FFB0

Khi này máy tính kiểm tra 8 ký tự đầu của mã băm thỏa mãn điều kiện đều là 0 và số tiếp theo là số 4 nhỏ hơn giá trị mục tiêu đề ra. Khi đó mã băm được tính là hợp lệ và người đào sẽ nhận được Bitcoin.

Sơ đồ luồng xử lý của thuật toán:



# TẠI SAO NÊN HỌC LẬP TRÌNH



## Làm chủ công nghệ, làm chủ tương lai

Học lập trình giúp các bạn trẻ hiểu rõ hơn và có cái nhìn sâu sắc hơn về một thế giới mà công nghệ hiện diện ở khắp mọi nơi, khả năng ứng dụng sức mạnh to lớn của công nghệ là điều kiện tiên quyết cho mọi thành công của hiện tại cũng như trong tương lai.

## Cơn khát nhân lực ngành CNTT

Liên tục trong 5 năm nhu cầu tuyển dụng ngành công nghệ thông tin mỗi năm tăng trên 50%, dự báo năm 2020 thiếu 100.000 trên tổng số nhu cầu 400.000 nhân sự ngành CNTT. Ngoài ra khoa học máy tính và kỹ thuật phần mềm cũng thuộc nhóm những ngành ít bị ảnh hưởng và thay thế bởi trí tuệ nhân tạo trong tương lai không xa.



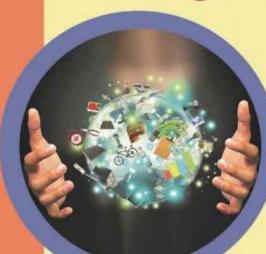
## Nhóm ngành có thu nhập cao nhất

Số liệu những năm gần đây cho thấy CNTT luôn thuộc nhóm ngành có mức lương cao nhất ở Việt Nam cũng như trên thế giới. Chỉ với một chút nỗ lực và sự đam mê công nghệ, đây sẽ là con đường ngắn và ít rủi ro nhất để các bạn trẻ nhanh chóng đạt tự chủ về tài chính cho bản thân.



## Một ngành đầy năng động và sáng tạo

Nhân sự trong lĩnh vực CNTT đều còn rất trẻ, tài năng, hoài bão và khát vọng. Làm việc trong một môi trường như vậy, bạn sẽ phát huy hết những tiềm năng, năng lực cũng như óc sáng tạo vốn có của bản thân.



## Ngành có nhiều cơ hội và thách thức, luôn tiếp cận những tri thức mới

Trong lĩnh vực CNTT, kiến thức và công nghệ của vài năm trước đây đã hoàn toàn lỗi thời so với hiện tại, các bạn sẽ luôn được cập nhật những tri thức mới nhất, công nghệ hiện đại nhất của nhân loại. Là ngành quy tụ những trí tuệ siêu việt, do vậy đây cũng là môi trường có tính cạnh tranh gay gắt và đào thải khốc liệt nhất. Nếu là người tài năng và hoài bão, say mê khám phá và ưa sự mới mẻ, bạn có thể vượt qua tất cả. Hầu hết những nhân vật nổi tiếng trong ngành CNTT đều khởi đầu từ hai bàn tay trắng, nhưng ngày nay họ được cả thế giới ngưỡng mộ như là những huyền thoại về trí tuệ, sức sáng tạo vô biên và sự quả cảm của con người như Bill Gates, Michael Dell, Steve Jobs, Elon Musk, Larry Page, Mark Zuckerberg,...

## HỌC VIỆN VIETSTEM



🌐 <https://vietstem.com>  
✉ support@vietstem.com  
☎ 02433 824 666  
🏡 Hà Nội - Việt Nam

Giá: 200.000VNĐ