Problem: Elementary Computer Science – Bài Tập Tin Học Sơ Cấp

Nguyễn Quản Bá Hồng*

Ngày 5 tháng 10 năm 2024

Tóm tắt nội dung

This text is a part of the series *Some Topics in Elementary STEM & Beyond*:
URL: https://nqbh.github.io/elementary_STEM.
Latest version:

- *Problems in Elementary Computer Science – Bài Tập Tin Học Sơ Cấp.*
  PDF: URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_computer_science/problem/NQBH_elementary_computer_science_problem.pdf.
  TEX: URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_computer_science/problem/NQBH_elementary_computer_science_problem.tex.
- *Problem & Solution: Elementary Computer Science – Bài Tập & Lời Giải: Tin Học Sơ Cấp.*
  PDF: URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_computer_science/problem/NQBH_elementary_computer_science_solution.pdf.
  TEX: URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_computer_science/problem/NQBH_elementary_computer_science_solution.tex.

Mục lục

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

General structure – Cấu trúc tổng quát

1 (General structure of a programming problem)**.** *(a) Viết thuật toán, vẽ lưu đồ, & viết chương trình* Pascal, Python, C/C++ *để:*

- *Tính*
- *Tìm*
- *Xác định*

(b) Mở rộng & tổng quát bài toán.

1 Basic – Cơ Bản

1.1 Elementary algebra – Đại số sơ cấp

2 ([DT06], VD1, p. 14). *(a) Cho $A = x^2 + y^2, B = x + y + A, C = xy + A - B^2$ với $x, y \in \mathbb{R}$. Tính A, B, C. (b) Mở rộng bài toán cho $A_i(x, y) \in \mathbb{R}[x, y]$ với $i = 1, 2, \ldots, n$, $n \in \mathbb{N}^\star$, là các đa thức 2 biến x, y. (c) Mở rộng bài toán cho $A_i(\mathbf{x}) \in \mathbb{R}[\mathbf{x}]$ với $i = 1, 2, \ldots, n$, là các đa thức hệ số thực với $n \in \mathbb{N}^\star$ biến số $\mathbf{x} = (x_1, x_2, \ldots, x_n)$.*

See also:

- *Problem: Algebraic Expression – Bài Tập: Biểu Thức Đại Số.*

  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 7/algebraic expression/problem: [pdf[1]][TEX[2]].

- *Problem & Solution: Algebraic Expression – Bài Tập & Lời Giải: Biểu Thức Đại Số.*

  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 7/algebraic expression/solution: [pdf[3]][TEX[4]].

3 ([DT06], VD2, p. 15). *(a) Tìm* GTLN $\max\{a, b, c\}$ *&* GTNN $\min\{a, b, c\}$ *với $a, b, c \in \mathbb{R}$. (b) Mở rộng bài toán cho $n \in \mathbb{N}^\star$ số $a_1, a_2, \ldots, a_n$.*

4 ([DT06], VD3, p. 16). *(a) Tính giá trị của hàm số $f : \mathbb{R}^2 \to \mathbb{R}$ xác định bởi công thức:*

$$f(x, y) = \begin{cases} x^3 + y^3 & \text{if } 0 \le x - y \le 10, \\ x^2 + y^2 & \text{if } x - y < 0,\ y \ge 0, \\ (x - y)^2 & \text{else.} \end{cases}$$

(b) Mở rộng bài toán cho các hàm số tương tự.

5 ([DT06], VD4, p. 17). *(a) Tìm nghiệm của phương trình bậc ≤ 2 có dạng $ax^2 + bx + c = 0$ với $a, b, c \in \mathbb{R}$. (b) Mở rộng bài toán cho phương trình bậc 3, bậc 4. (c) Mở rộng bài toán cho các phương trình đa thức có thể giải được nhờ quy về phương trình bậc 2, 3.*

See also:

- *Problem: 2nd-Order Function. Quadratic Equation – Bài Tập: Hàm Số Bậc 2 $y = ax^2$. Phương Trình Bậc 2 1 Ẩn $ax^2 + bx + c = 0$.*

  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 9/2nd-order function, quadratic equation/problem: [pdf[5]][TEX[6]].

- *Problem & Solution: 2nd-Order Function. Quadratic Equation – Bài Tập & Lời Giải: Hàm Số Bậc 2 $y = ax^2$. Phương Trình Bậc 2 1 Ẩn $ax^2 + bx + c = 0$.*

  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 9/2nd-order function, quadratic equation/solution: [pdf[7]][TEX[8]].

6 ([DT06], VD5, p. 18). *Tính tổng $S = \sum_{i=1}^{100} x_i$ với $x_i \in \mathbb{R}$, $\forall i = 1, 2, \ldots, 100$. (b) Mở rộng bài toán cho các tổng tương tự.*

7 ([DT06], VD6, p. 19). *Tính giá trị của đa thức $P(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \in \mathbb{R}[x]$ với hệ số thực $a_i \in \mathbb{R}$, $\forall i = 0, 1, \ldots, n$ với $n \in \mathbb{N}^\star$ bằng cách sử dụng thuật toán Horner.*

8 ([DT06], VD7, p. 21). *Tính giá trị hàm số mũ e^x với $x = a$ với sai số $\varepsilon > 0$ cho trước bằng cách sử dụng khai triển Taylor của hàm e^x.*

9 ([DT06], VD8, p. 22). *Tìm nghiệm của phương trình $f(x) = x^5 - 6x^4 - 15x^3 - 20x^2 + 14x - 4 = 0$ với giá trị đầu (initial value) $x_0 = 0.8$.*

[1] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_7/algebraic_expression/problem/NQBH_algebraic_expression_problem.pdf.
[2] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_7/algebraic_expression/problem/NQBH_algebraic_expression_problem.tex.
[3] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_7/algebraic_expression/problem/NQBH_algebraic_expression_solution.pdf.
[4] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_7/algebraic_expression/problem/NQBH_algebraic_expression_solution.tex.
[5] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/2nd_order_function/problem/NQBH_2nd_order_function_problem.pdf.
[6] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/2nd_order_function/problem/NQBH_2nd_order_function_problem.tex.
[7] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/2nd_order_function/solution/NQBH_2nd_order_function_solution.pdf.
[8] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/2nd_order_function/solution/NQBH_2nd_order_function_solution.tex.

10 ([DT06], VD9, p. 23). *Giải hệ phương trình tuyến tính cấp n có dạng*

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2, \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \end{cases} \tag{1}$$

See also:

- *Problem: System of 1st-Order Equations – Bài Tập: Hệ Phương Trình Bậc Nhất* $A\mathbf{x} = \mathbf{b}$.
  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 9/system of 1st-order equations/problem: [pdf[9]][TEX[10]].
- *Problem & Solution: System of 1st-Order Equations – Bài Tập & Lời Giải: Hệ Phương Trình Bậc Nhất* $A\mathbf{x} = \mathbf{b}$.
  Folder: Elementary STEM & Beyond/Elementary Mathematics/grade 9/system of 1st-order equations/solution: [pdf[11]][TEX[12]].
- Problem: Trigonometry in Triangles – Bài Tập: Hệ Thức Lượng Trong Tam Giác. URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/trigonometry/problem/NQBH_trigonometry_problem.pdf.

11 ([DT06], 10., p. 27). *Tính tổ hợp chập k trong n phần tử* C_n^k.

12 ([DT06], 11., p. 27). *Tính tích của n số thực* $x_1, x_2, \ldots, x_n \in \mathbb{R}$.

13 ([DT06], 12., p. 27). *Tính giá trị của biểu thức* $A = \sum_{i=1}^{100} x_i + \prod_{i=1}^{100} x_i$.

14 ([DT06], 13., p. 27). *Xác định các số dương trong chuỗi số* $x_1, x_2, \ldots, x_n$.

15 ([DT06], 14., p. 27). *Tính trung bình số học của các phần tử trong chuỗi số* $x_1, x_2, \ldots, x_n$ *trong đoạn* $[a, b]$ *với* $a, b \in \mathbb{R}$, $a < b$.

16 ([DT06], 15., p. 27). *Tính 100 số Fibonacci đầu tiên.*

17 ([DT06], 16., p. 27). *Tìm số lớn nhất trong dãy số* $x_1, x_2, \ldots, x_n$.

18 ([DT06], 17., p. 27). *Tính các phần tử của tam giác Pascal tới* C_{100}^{100}.

19 ([DT06], 18., p. 27). *Gọi* $x_1, x_2, \ldots, x_n$ *là lượng cholesterol trung bình của người đến khám bệnh. Người khỏe có lượng cholesterol nhỏ hơn tiêu chuẩn* $x < A$. *Xác định số người không mắc bệnh & lượng colesterol trung bình của người bệnh.*

20 ([DT06], 19., p. 27). *Viết các hoán vị với 5 số* $1, 2, 3, 4, 5$.

21 ([DT06], 20., p. 27). *Xếp các số theo thứ tự tăng dần hoặc giảm dần.*

22 ([DT06], 21., p. 27). *Lập danh bạ điện thoại 5 số.*

23 ([DT06], 22., p. 27). *Cho* $a, b, c > 0$ *bất kỳ. Nếu 3 số đó có thể là chiều dài 3 cạnh của 1 tam giác, tính chu vi & diện tích tam giác đó.*

24 ([DT06], 23., p. 28). *Tính* $n!$ *với* $n \in \mathbb{Z}$.

25 ([DT06], 24., p. 28). *Tìm ước chung lớn nhất* ƯCLN *của 2 số.*

26. *Tìm bội chung nhỏ nhất* BCNN *của 2 hay nhiều số.*

27 ([DT06], 25., p. 28). *Tính tổng & tích của 1 chuỗi số biết giá trị đầu & số các số hạng của chuỗi.*

28 ([DT06], 26., p. 28). *Tính tổng của chuỗi* $\sum_{i=1}^{n} \frac{1}{i} = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$.

29 ([DT06], 27., p. 28). *Xác định các số nguyên tố trong đoạn* $[1, n]$ *với* $n \in \mathbb{N}$ *cho trước.*

30 ([DT06], 28., p. 28). *Tính* $y = \log_a x$ *với cơ số* $a > 0$ *bất kỳ.*

[9] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/system_1st_order_equations/problem/NQBH_system_1st_order_equations_problem.pdf.
[10] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/system_1st_order_equations/problem/NQBH_system_1st_order_equations_problem.tex.
[11] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/system_1st_order_equations/solution/NQBH_system_1st_order_equations_solution.pdf.
[12] URL: https://github.com/NQBH/elementary_STEM_beyond/blob/main/elementary_mathematics/grade_9/system_1st_order_equations/solution/NQBH_system_1st_order_equations_solution.tex.

1.2 Vector, matrix, system of algebraic equations – Vector, ma trận, hệ phương trình đại số

31 ([DT06], 29., p. 28). *Thực hiện phép cộng, trừ, tích vô hướng, tích vector đối với 2 vector trong không gian 3 chiều.*

32 ([DT06], 30., p. 28). *Tính tích vô hướng của 2 vector n chiều* $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$.

33 ([DT06], 31., p. 28). *Cho 2 vector n chiều* $\mathbf{x} = (x_1, x_2, \ldots, x_n), \mathbf{y} = (y_1, y_2, \ldots, y_n)$. *Tính vector* $\mathbf{z} = (z_1, z_2, \ldots, z_n)$ *có các thành phần được xác định như sau:*

$$z_i = \begin{cases} 0 & \text{if } 0 < x_i + y_i < a, \\ x_i + y_i & \text{if } x_i + y_i \geq a, \\ (x_i + y_i)^2 & \text{if } x_i + y_i \leq 0, \end{cases} \qquad \forall i = 1, 2, \ldots, n,$$

& tích tích vô hướng $\mathbf{x} \cdot \mathbf{z}$.

34 ([DT06], 32., p. 28). *Tính ma trận chuyển vị của ma trận* $A = (a_{ij})_{i,j=1}^{m,n}$.

35 ([DT06], 33., p. 28). *Xác định phần tử lớn nhất & phần tử nhỏ nhất trong ma trận* $A = (a_{ij})_{i,j=1}^{m,n}$.

36 ([DT06], 34., p. 28). *Tìm ma trận tổng* $C = A + B$.

37 ([DT06], 35., p. 28). *Tìm ma trận tích* $C = AB$.

38 ([DT06], 36., p. 28). *Tính định thức D của hệ phương trình tuyến tính bậc* $n \in \mathbb{N}^\star$.

39 ([DT06], 37a., p. 28). *Tính ma trận nghịch đảo.*

40 ([DT06], 37b., p. 28). *Giải hệ phương trình đại số tuyến tính bậc* $n \in \mathbb{N}^\star$ *bằng phương pháp khử Gauss.*

1.3 Function – Hàm số

41 ([DT06], 38., p. 29). *Vẽ đồ thị hàm số theo từng điểm.*

42 ([DT06], 39., p. 29). *Vẽ đồ thị hàm số, thực hiện dịch chuyển tọa độ & vẽ các điểm.*

43 ([DT06], 40., p. 29). *Xác định hàm số theo đồ thị hàm số.*

44 ([DT06], 41., p. 29). *Tính giá trị liên tiếp của hàm số* $y_i = \sin e^{x_i}$ *trong đó* $x_i := a + i(b - a)$, $i = 1, 2, \ldots, 100$ *với* $a, b \in \mathbb{R}$ *cho trước.*

45 ([DT06], 42., p. 29). *Tính các giá trị của hàm số* $y_i = e^{x_i} + \sin x_i$ *với* $x_i = a + hi$, $i = 1, 2, \ldots, n$.

46 ([DT06], 43., p. 29). *Tính* $\sqrt[n]{x}$.

47 ([DT06], 44., p. 29). *Lập chương trình tính toán các lượng phức: cộng, trừ, nhân, chia, lũy thừa các số phức.*

48 ([DT06], 45., p. 29). *Lập chương trình đổi radian ra độ.*

49 ([DT06], 46., p. 29). *Lập chương trình đổi độ ra radian.*

50 ([DT06], 47., p. 29). *Tính giá trị của 2 hàm số* $\sinh x, \cosh x$.

51 ([DT06], 48., p. 29). *Tính giá trị hàm số* $y = a^x$.

1.4 Polynomials, interpolation polynomials – Đa thức, đa thức nội suy

52 ([DT06], 49., p. 29). *Tính giá trị của đa thức Chevbyshev bậc* $n \in \mathbb{N}^\star$

$$T_n(x) := \cos(n \arccos x),$$
$$U_n(x) := \frac{\sin((n+1)\arccos x)}{2^n \sqrt{1 - x^2}}.$$

53 ([DT06], 50., p. 29). *Tính giá trị của đa thức Legendre bậc* $n \in \mathbb{N}^\star$

$$L_n(x) = A_n \sum_{k=0}^{\lfloor n \rfloor} (-1)^k \frac{(2n-2k)!}{(n-2k)!} C_n^k x^{n-k}.$$

54 ([DT06], 51., p. 29). *Tính giá trị của đa thức Laguerre:*

$$L_n^\alpha(x) = (-1)^n x^{-\alpha} e^x \frac{d^n}{dx^n}(x^{\alpha+n} e^{-x}).$$

55 ([DT06], 52., p. 30). *Tính giá trị của đa thức Hermite:*

$$H_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{\frac{x^2}{2}}. \tag{2}$$

56 ([DT06], 53., p. 30). *Cho bảng các giá trị (x_i, y_i). Tìm đa thức nội suy Lagrange.*

57 ([DT06], 54., p. 30). *Cho bảng các giá trị (x_i, y_i). Tìm ma trận nội suy Lagrange.*

58 ([DT06], 55., p. 30). *Tìm đa thức nội suy Newton. Tính giá trị của đa thức nội suy tại điểm x_0 theo công thức Newton.*

59 ([DT06], 56., p. 30). *Xác định trị số của đa thức nội suy tại điểm x_0 theo công thức Horner.*

60 ([DT06], 57., p. 30). *Tính tổng 2 đa thức.*

61 ([DT06], 58., p. 30). *Nhân 2 đa thức.*

62 ([DT06], 59., p. 30). *Chia 2 đa thức.*

1.5 Differentiation & derivative – Phép tính vi phân & đạo hàm

63 ([DT06], 60., p. 30). *Tìm đạo hàm cấp $m \in \mathbb{N}$ của đa thức $P(x)$.*

64 ([DT06], 61., p. 30). *Tìm đạo hàm cấp $m \in \mathbb{N}$ của đa thức nội suy Newton.*

65 ([DT06], 62., p. 30). *Tìm đạo hàm 1 biến.*

66 ([DT06], 63., p. 30). *Tìm gradient ∇ hàm nhiều biến.*

67 ([DT06], 64., p. 30). *Tìm giá trị của đạo hàm cấp $m \in \mathbb{N}$ của đa thức $P(x)$ tại điểm $x_0 \in \mathbb{R}$.*

68 ([DT06], 65., p. 30). *Tìm đạo hàm cấp 2 của hàm theo $x_i x_j$ tại 1 điểm theo công thức đạo hàm có bước biến đổi & 1 quá trình ngoại suy.*

1.6 Integration & integral – Phép tính tích phân & tích phân

69 ([DT06], 66., p. 30). *Tính tích phân xác định bằng phương pháp Simpson.*

70 ([DT06], 67., p. 30). *Tính tích phân xác định bằng phương pháp hình thang.*

71 ([DT06], 68., p. 30). *Tính tích phân kép trong miền D bằng phương pháp Gauss–Legendre.*

72 ([DT06], 69., p. 30). *Tính tích phân kép bằng phương pháp Romberg.*

1.7 Nonlinear system – Phương trình phi tuyến

73 ([DT06], 70., p. 31). *Giải phương trình phi tuyến bằng phương pháp chia đôi cung.*

74 ([DT06], 71., p. 31). *Giải phương trình phi tuyến bằng phương pháp lặp cát tuyến.*

75 ([DT06], 72., p. 31). *Giải phương trình phi tuyến bằng phương pháp lặp Newton.*

1.8 Optimization – Tối ưu hóa

76 ([DT06], 73., p. 31). *Tìm cực trị của hàm số bằng phương pháp tiết diện vàng.*

77 ([DT06], 74., p. 31). *Tìm cực trị của hàm nhiều biến bằng phương pháp gradient.*

78 ([DT06], 75., p. 31). *Tìm cực trị của hàm 1 biến.*

1.9 Differential equations – Phương trình vi phân

79 ([DT06], 76., p. 31). *Giải phương trình vi phân bằng phương pháp Euler & Euler cải tiến.*

80 ([DT06], 77., p. 31). *Giải phương trình tích phân bằng phương pháp Romberg.*

81 ([DT06], 78., p. 31). *Giải phương trình vi phân bằng phương pháp Runge–Kutta.*

82 ([DT06], 79., p. 31). *Giải hệ phương trình vi phân cấp $n \in \mathbb{N}^\star$ bằng phương pháp Runge–Kutta.b*

1.10 Probability & Statistics – Xác suất & Thống kê

83 ([DT06], 80., p. 31)**.** *Tìm phân bố nhị thức biết số phép thử $n = 10$, xác suất xuất hiện biến cố ngẫu nhiên trong mỗi phép thử là $p = 0.45$, $x = 4$.*

84 ([DT06], 81., p. 31)**.** *Dùng luật phân bố Poisson để tính trong* 1000 *trang sách có* 100 *lỗi in sai, tính xác suất để khi lấy ngẫu nhiên 1 trang có không quá* 4 *lỗi.*

85 ([DT06], 82., p. 31)**.** *Biến ngẫu nhiên x_i có xác suất p_i cho bằng bảng phân bố x_i, p_i. Tìm kỳ vọng, phương sai, độ lệch bình phương trung bình, lấy $a = 2$.*

86 ([DT06], 83., pp. 31–32)**.** *Xác định hệ số tương quan giữa 2 biến ngẫu nhiên x, y. Kiểm tra xem mối liên hệ giữa x, y có đủ tin cậy không.*

1.11 Document managing & processing – Quản lý & xử lý văn bản

87 ([DT06], 84., p. 32)**.** *Lập chương trình in các ngày trong 1 tuần lễ.*

88 ([DT06], 85., p. 32)**.** *Lập chương trình in lịch thế kỷ.*

89 ([DT06], 86., p. 32)**.** *Quản lý 1 file nhập dữ liệu bằng bảng.*

90 ([DT06], 87., p. 32)**.** *Lập chương trình tìm kiếm 1 phần tử trong 1 danh sách.*

91 ([DT06], 88., p. 32)**.** *Lập chương trình làm 1 tờ hóa đơn.*

92 ([DT06], 89., p. 32)**.** *Lập chương trình quản lý danh sách cán bộ.*

93 ([DT06], 90., p. 32)**.** *Lập chương trình in danh sách cán bộ.*

94 ([DT06], 91., p. 32)**.** *Quản lý 1 file nhập dữ liệu bằng con trỏ.*

1.12 Technology – Bài toán kỹ thuật

95 ([DT06], 92., p. 32)**.** *Tính đặc tính tần số của tổng trở, góc pha, dòng điện, & điện áp trên các phần tử của mạch R-L-C nối tiếp.*

96 ([DT06], Chế độ quá độ trong 1 điện trở đốt nóng, 97., p. 35)**.** *1 điện trở kim loại có hệ số nhiệt $\alpha = 10^{-3}\mathrm{K}^{-1}$. Trong miền nhiệt độ T, điện trở phụ thuộc vào nhiệt độ theo quy luật $R = R_0(1 + \alpha(T - T_0))$. Điện trở truyền công suất nhiệt vào 1 môi trường xung quanh có nhiệt độ T_0: $P_T = G(T - T_0)$; nhiệt dung C của điện trở là $2\mathrm{JK}^{-1}$. Giả thiết nhiệt độ là đều ở mỗi thời điểm, điện áp đặt vào điện trở là $U = 220$ V. Tại thời điểm $t = 0$ nhiệt độ bằng nhiệt độ môi trường $T_0 = 300$ K, đặt $\theta = T - T_0$. (a) Viết phương trình vi phân của hàm $\theta(t)$. (b) Khi đạt tới chế độ xác lập ta đo được nhiệt độ* 800 K *& công suất tiêu tán* 500 W*. Xác định trị số R_0, G. (c) Giải phương trình vi phân bằng phương pháp số. Chọn bước sao cho kết quả chính xác với 2 chữ số. Xác định nhiệt độ ở mỗi giây từ* 0–10 s*. So sánh với các kết quả thu được khi cho rằng ngay lúc đầu điện trở có giá trị cuối.*

Các bài toán không được trình bày do bao gồm hình vẽ khá phức tạp: [DT06, 93., 94., 95., 96., 98., 99., 100., 101., pp. 32–38]. Bạn đọc tìm đến sách này để tham khảo thêm.

2 Number Theory – Số Học

97 ([Huy24], p. 6, số thân thiện – friendly number)**.** *(a) Tìm tất cả các số tự nhiên có 2 chữ số mà khi đảo trật tự của 2 chữ số đó sẽ thu được 1 số nguyên tố cùng nhau với số đã cho. (b) Mở rộng bài toán cho số tự nhiên có n chữ số với $n \in \mathbb{N}, n \geq 2$.*

98 ([Huy24], p. 13, cấp số cộng – arithmetic progression)**.** *(a) Tìm các số tự nhiên lẻ gồm 3 chữ số. 3 chữ số này, theo thứ tự từ trái qua phải tạo thành 1 cấp số cộng. (b) Mở rộng bài toán cho số tự nhiên (không nhất thiết phải lẻ) có n chữ số với $n \in \mathbb{N}, n \geq 2$.*

99 ([Huy24], p. 17, cấp số nhân – geometric progression)**.** *(a) Tìm các số tự nhiên gồm 3 chữ số. 3 chữ số này, theo thứ tự từ trái qua phải tạo thành 1 cấp số nhân với công bội $q \in \mathbb{N}^\star$. (b) Mở rộng bài toán cho số tự nhiên có n chữ số với $n \in \mathbb{N}, n \geq 2$.*

100 ([Tru23], 1., p. 13, HSG Lớp 10 Vĩnh Phúc 2020–2021, Square – Hình vuông)**.** *(a) Cho n điểm có tọa độ là các số nguyên trên hệ trục tọa độ Oxy. Tìm diện tích hình vuông nhỏ nhất có các cạnh song song với các trục tọa độ sao cho tất cả các điểm đã cho đều thuộc hình vuông đó (điểm nằm trên cạnh hình vuông cũng được coi là thuộc hình vuông đó).*

- Input. *Dòng 1: chứa số nguyên dương $n \in \mathbb{N}^\star$, $2 \leq n \leq 20$, là số lượng điểm có tọa độ là các số nguyên. n dòng tiếp theo, mỗi dòng ghi 2 số nguyên $x, y \in \mathbb{Z}$, $1 \leq x, y \leq 100$, là tọa độ của mỗi điểm.*
- Output. *Ghi diện tích hình vuông nhỏ nhất tìm được.*

- Sample.

square.inp	square.out
3 3 4 5 7 4 3	16

(b) Mở rộng bài toán từ 'hình vuông' sang 'hình chữ nhật', & từ 'tọa độ nguyên' sang 'tọa độ thực': (i) Cho n điểm có tọa độ là các số thực trên hệ trục tọa độ Oxy. Tìm diện tích hình vuông, hình vuông "nguyên", hình chữ nhật, & hình chữ nhật "nguyên" nhỏ nhất có các cạnh song song với các trục tọa độ sao cho tất cả các điểm đã cho đều thuộc hình chữ nhật đó (điểm nằm trên cạnh hình chữ nhật cũng được coi là thuộc hình chữ nhật đó), trong đó hình vuông, hình chữ nhật "nguyên" *lần lượt là các hình vuông & hình chữ nhật có các tọa độ của 4 đỉnh là 8 số nguyên.*

101 ([Tru23], 2., pp. 13–14, HSG Lớp 10 Vĩnh Phúc 2020–2021, Divisible by 3 – Chia hết cho 3)**.** *Cho dãy a gồm n số nguyên dương. Cho biết có bao nhiêu cặp số trong dãy có tổng chia hết cho 3, i.e., đếm xem có bao nhiêu cặp chỉ số i, j, $1 \le i < j \le n$, sao cho $a_i + a_j \vdots 3$.*

- Input. *Dòng 1: 1 số nguyên duy nhất n, $1 \le n \le 10^5$. Dòng 2: Ghi n số nguyên dương $a_1, a_2, \ldots, a_n$, $1 \le a_i \le 10^5$, $\forall i = 1, 2, \ldots, n$, là các phần tử của dãy.*
- Output. *1 dòng duy nhất ghi số lượng cặp số của dãy a có tổng chia hết cho 3.*
- Sample.

div3.inp	div3.out	Giải thích
5 3 6 9 12	3	3 cặp số tìm được có chỉ số: $(1,4), (2,3), (3,5)$.
4 3 6 9 12	6	6 cặp số tìm được có chỉ số: $(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)$.

102 ([Tru23], 3., p. 14, HSG Lớp 10 Vĩnh Phúc 2020–2021, Delete element – Xóa phần tử)**.** *Cho dãy gồm n số nguyên $a_1, a_2, \ldots, a_n$ với $1 \le a_i \le 3$, $\forall i = 1, 2, \ldots, n$. Có bao nhiêu cách để xóa đi 1 số phần tử của dãy (không xóa phần tử nào cũng được coi là 1 cách) mà vẫn giữ nguyên thứ tự ban đầu để được 1 dãy mới thỏa mãn 2 yêu cầu sau: (i) Dãy còn ít nhất 3 phần tử. (ii) Phần tử đầu tiên của dãy có giá trị 1, tiếp theo là 1 số phần tử có giá trị là 2 (ít nhất có 1 số 2), & kết thúc bằng đúng 1 phần tử có giá trị là 3. E.g., các dãy $1, 2, 2, 3$ & $1, 2, 3$ thỏa mãn yêu cầu, các dãy $1, 2, 3, 3$ & $1, 1, 2, 3$ không thỏa mãn yêu cầu.*

- Input. *Dòng 1: 1 số nguyên dương $n \in \mathbb{N}^\star$, $n \le 10^6$, là số lượng phần tử của dãy. Dòng 2: Ghi n số nguyên dương $a_1, a_2, \ldots, a_n$ là giá trị của các phần tử của dãy ban đầu.*
- Output. *Gồm 1 dòng duy nhất là số cách xóa để được dãy mới thỏa mãn yêu cầu của đề bài. Do số lượng cách xóa phần tử có thể rất lớn nên chỉ cần ghi ra số lượng cách xóa sau khi chia lấy dư cho $10^9 + 7$.*
- Sample.

delete_element.inp	delete_element.out
8 1 2 1 2 3 1 2 3	15

103 ([Tru23], 1., p. 15, HSG Lớp 11 Vĩnh Phúc 2020–2021, Game button – Trò chơi bấm nút)**.** *Người chơi đang tham gia 1 trò chơi như sau: Có 2 nút bấm A, B, trên nút A có ghi số m_A, trên nút B có ghi số m_B. Ở mỗi lượt chơi, người chơi phải chọn bấm 1 trong 2 nút & sẽ nhận được số điểm thưởng bằng với số ghi trên nút đó, sau đó số trên nút bấm giảm đi 1 đơn vị. Hỏi sau 2 lượt chơi, số điểm thưởng lớn nhất mà người chơi có thể nhận được là bao nhiêu?*

- Input. *1 dòng duy nhất ghi 2 số nguyên dương m_A, m_B với $3 \le A, B \le 20$, tương ứng với 2 số ghi trên 2 nút A & B.*
- Output. *Ghi số điểm thưởng lớn nhất mà người chơi có thể nhận được sau 2 lượt chơi.*
- Sample.

game_button.inp	game_button.out	Giải thích
5 3	9	Bấm 2 lần nút A & sẽ có tổng điểm thưởng: $5 + 4 = 9$.

104 ([Tru23], 2., p. 15, HSG Lớp 11 Vĩnh Phúc 2020–2021, Count number – Đếm số)**.** *Cho 4 số nguyên dương a, b, c, d. Đếm xem có bao nhiêu số nguyên dương $x \in \mathbb{N}^\star$ thỏa mãn các điều kiện sau: (i) $a \le x \le b$. (ii) $x \not\vdots c$. (iii) $x \not\vdots d$.*

- Input. *1 dòng duy nhất ghi 4 số a, b, c, d, $1 \le a, b \le 10^{18}$, $1 \le c, d \le 10^9$.*
- Output. *1 dòng duy nhất ghi số lượng số nguyên dương $x \in \mathbb{N}^\star$ thỏa mãn điều kiện đề bài.*
- Sample.

count_number.inp	count_number.out	Giải thích
4 9 2 3	2	Chỉ có số 5 & 7 thỏa mãn điều kiện đề bài.

105 ([Tru23], 3., p. 16, HSG Lớp 11 Vĩnh Phúc 2020–2021, Reverse & reverse – Lật qua lật lại)**.** *Cho dãy a gồm $n \in \mathbb{N}^\star$ phần tử $1, 2, \ldots, n$. Người ta thực hiện trên dãy số này đúng k lần 2 thao tác sau: (i) Đầu tiên, đảo ngược thứ tự (lật đối xứng) đoạn phần tử có chỉ số từ u đến v. (ii) Tiếp theo, đảo ngược thứ tự (lật đối xứng) đoạn phần tử có chỉ số từ l đến r. Với u, v, l, r là các hằng số cho trước. Đưa ra dãy a sau khi thực hiện k lần 2 thao tác nói trên.*

- Input. *Dòng 1: 2 số nguyên dương $n, k \in \mathbb{N}^\star$, $1 \le n \le 100$, $1 \le k \le 10^9$. Dòng 2: gồm 2 số nguyên dương u, v, $1 \le u < v \le n$. Dòng 3: gồm 2 số nguyên dương l, r, $1 \le l < r \le n$.*
- Output. *Ghi trên n dòng, dòng thứ i ghi giá trị của phần tử thứ i của dãy a sau khi thực hiện k lần 2 thao tác nói trên, $\forall i = 1, 2, \ldots, n$.*
- Sample.

reverse_reverse.inp	reverse_reverse.out	Giải thích
7 2 2 5 3 7	1 2 4 3 5 7 6	Dãy ban đầu: 1 2 3 4 5 6 7 Lần 1: 1 *5 4 3 2* 6 7 1 5 *7 6 2 3 4* Lần 2: 1 *2 6 7 5* 3 4 1 2 4 3 5 7 6

106 ([Tru23], 3., p. 17, HSG Lớp 12 Vĩnh Phúc 2020–2021, Max gift – Chọn quà mắc nhất)**.** *Cuối năm công ty tổ chức phát qua cho nhân viên. Có $n \in \mathbb{N}^\star$ gói quà với giá trị khác nhau được xếp liên tiếp thành 1 hàng, trong đó gói quà thứ i có giá trị là a_i. Mỗi nhân viên chỉ được chọn 2 gói quà liên tiếp. Mr. Bean là người may mắn được chọn đầu tiên. Giúp Mr. Bean chọn ra 2 gói quà liên tiếp có giá trị lớn nhất.*

- Input. *Dòng 1: chứa số nguyên dương $n \in \mathbb{N}^\star$, $2 \le n \le 10^6$. Dòng 2: Giá trị của n gói quà, $1 \le a_i \le 10^3$, $\forall i = 1, 2, \ldots, n$, mỗi giá trị cách nhau bởi dấu cách.*
- Output. *1 dòng duy nhất chứa tổng giá trị quà lớn nhất chọn được.*
- Sample.

max_gift.inp	max_gift.out
5 1 3 5 4 2	9

107 ([Tru23], 2., pp. 17–18, HSG Lớp 12 Vĩnh Phúc 2020–2021, Decrease value – Giảm giá trị)**.** *1 ngày rảnh rỗi, Mr. Bean chơi trò chơi với các con số. Mr. Bean lấy 1 số nguyên dương $n \in \mathbb{N}^\star$ rồi thực hiện không giới hạn số lần thao tác "Chọn 1 chữ số x của n rồi giảm n đi x đơn vị". Hỏi Mr. Bean phải thực hiện ít nhất bao nhiêu thao tác như vậy để giảm số n về 0. E.g., $n = 27$, Mr. Bean sẽ thực hiện 5 thao tác để biến đổi: (i) Chọn $x = 7 \to n = 27 - 7 = 20$. (ii) Chọn $x = 2 \to n = 20 - 2 = 18$. (iii) Chọn $x = 8 \to n = 18 - 8 = 10$. (iv) Chọn $x = 1 \to n = 10 - 1 = 9$. (v) Chọn $x = 9 \to n = 9 - 9 = 0$.*

- Input. *1 dòng: 1 số nguyên dương duy nhất n, $1 \le n \le 10^6$.*
- Output. *1 dòng duy nhất ghi số thao tác ít nhất để biến đổi n về 0.*
- Sample.

decrease_value.inp	decrease_value.out
27	5

108 ([Tru23], 3., p. 18, HSG Lớp 12 Vĩnh Phúc 2020–2021, Difference degree of substrings – Xâu con phân biệt)**.** *1 lần Mr. Bean được bạn gái gửi cho 1 dãy ký tự S độ dài n chỉ gồm các chữ cái in hoa ('A',...,'Z'). Bạn gái nhờ Mr. Bean xác định "độ phân biệt" của dãy ký tự trên. Trong đó* độ phân biệt *của dãy ký tự là số nguyên dương l nhỏ nhất sao cho tất cả các xâu con của S độ dài l là đôi một phân biệt. E.g., với $n = 7$, $S =$ 'ABCDABC' thì $l = 4$ do tất cả các xâu con độ dài 4 đều phân biệt. Giúp Mr. Bean việc đó.*

- Input. *Dòng 1: số nguyên dương $n \in \mathbb{N}^\star$, $n \leq 100$. Dòng 2: chứa xâu ký tự S.*
- Output. *Gồm 1 dòng duy nhất ghi 1 số nguyên duy nhất là "độ phân biệt" của dãy ký tự S.*
- Sample.

diff_substring.inp	diff_substring.out
7 ABCDABC	4

109 ([Tru23], 4., p. 18, HSG Lớp 12 Vĩnh Phúc 2020–2021, Ants meet – Kiến tha mồi)**.** *Trên đường đi làm về Mr. Bean quan sát thấy 2 ổ kiến cách nhau 1 khoảng l đơn vị. Các con kiến đang tha mồi về 2 tổ trên đường thẳng nối 2 tổ kiến với nhau. Các con kiến khi tha mồi về tổ nào thì ở lại tổ đó. Nếu 2 con kiến gặp nhau trên đường đi thì cả 2 sẽ đổi hướng di chuyển.*

Giả sử đường nối giữa 2 tổ kiến được gắn tọa độ từ 0 đến l. Tổ thứ nhất ở vị trí 0 & tổ thứ 2 ở vị trí l. Ở thời điểm Mr. Bean quan sát có n con kiến đang tha mồi về tổ. Con thứ i xuất phát ở tọa độ x_i, mang lượng mồi khối lượng w_i & có hướng di chuyển d_i. Nếu $d_i = 1$ thì con kiến thứ i đang di chuyển theo hướng 0 về l, $d_i = -1$ thì con kiến thứ i đang di chuyển theo chiều ngược lại. Tất cả các con kiến có tốc độ di chuyển bằng nhau & bằng 1 đơn vị đo độ dài trên giây.

Gọi t là thời điểm sớm nhất tính từ thời điểm quan sát mà tổng lượng mồi được tha về 2 tổ đạt ít nhất $\frac{1}{2}$ tổng lượng mồi của đàn kiến. Mr. Bean đếm được trong thời gian đó các con kiến gặp nhau đúng x lần, tính cả lần gặp nhau ở thời điểm t. Hỏi x bằng bao nhiêu?

- Input. *Dòng 1: 2 số nguyên dương $n, l \in \mathbb{N}^\star$, $1 \leq n \leq 5 \cdot 10^4$, $1 \leq l \leq 10^9$. Dòng 2,...,$n+1$: Dòng $i+1$ ghi 3 số nguyên w_i, x_i, d_i, $1 \leq w_i \leq 10^3$, $d_i = \pm 1$, $0 \leq x_i \leq l$, các số x_i, $i = 1, 2, \ldots, n$, đôi một phân biệt. Các số nguyên cách nhau 1 dấu cách.*
- Output. *1 dòng duy nhất chứa số nguyên $x \in \mathbb{N}^\star$ là số lần gặp nhau của các cặp kiến.*
- Sample.

ant_meet.inp	ant_meet.out
3 5 1 1 1 2 2 -1 3 3 -1	2

Giải thích: Thời điểm 0.5, kiến 1 gặp kiến 2 ở tọa độ 1.5, kiến 1 đổi hướng thành -1, kiến 2 đổi hướng thành 1. Thời điểm 1, kiến 2 gặp kiến 3 ở tọa độ 2, kiến 2 đổi hướng thành -1, kiến 3 đổi hướng thành 1. Thời điểm 2: kiến 1 về đến tổ ở tọa độ 0. Thời điểm 3: kiến 2 về đến tổ ở tọa độ 0, lúc này lượng mồi đạt được ở 2 tổ là 3, bằng $\frac{1}{2}$ tổng lượng mồi của cả 3 kiến.

110 ([Tru23], 1., p. 20, HSG Lớp 12 Nam Định 2020–2021, Nearly perfect number – Số gần hoàn hảo)**.** *1 số nguyên dương $a \in \mathbb{N}^\star$ được gọi là* số "gần hoàn hảo" *nếu thỏa mãn điều kiện: $2a \leq k$ với k là tổng các ước số của a, e.g., 12 là 1 số "gần hoàn hảo" vì $2 \cdot 12 < 1 + 2 + 3 + 4 + 6 + 12$ ($24 < 28$).*

- Input. *Dòng đầu tiên chứa số nguyên dương $n \in \mathbb{N}^\star$, $1 \leq n \leq 10^4$. n dòng tiếp theo, mỗi dòng là 1 số nguyên dương có giá trị $\leq 10^6$.*
- Output. *Dòng đầu tiên ghi số lượng số "gần hoàn hảo". Các dòng tiếp theo, mỗi dòng ghi 1 số "gần hoàn hảo", số gặp trước thì viết trước.*
- Sample.

near_perfect_number.inp	near_perfect_number.out
5 8 16 12 6 7	2 12 6

111 ([Tru23], 2., pp. 20–21, HSG Lớp 12 Nam Định 2020–2021, Special number – Số đặc biệt). *Cho 1 dãy gồm n số nguyên $a_1, a_2, \ldots, a_n$. Đếm & đưa ra số đặc biệt trong dãy a.* Số đặc biệt *là số chỉ xuất hiện đúng 1 lần trong dãy số.*

- Input. *Dòng đầu tiên là số $n \in \mathbb{N}^\star$, $1 \leq n \leq 10^6$. n dòng tiếp theo, dòng thứ i là số a_i, $|a_i| \leq 10^9$, $\forall i = 1, 2, \ldots, n$.*
- Output. *Dòng đầu tiên ghi số lượng số đặc biệt. Các dòng tiếp theo, mỗi dòng ghi 1 số đặc biệt tính từ đầu dãy a.*
- Sample.

special_number.inp	special_number.out
8 9 9 7 7 6 11 9 5	3 6 11 5

112 ([Tru23], 3., p. 21, HSG Lớp 12 Nam Định 2020–2021, Game of gifts – Trò chơi tặng quà). *1 công ty có tổ chức trò chơi, tặng $n \in \mathbb{N}^\star$ gói quà đã được chuẩn bị theo giá trị phần quà từ thấp đến cao, để tri ân cho n khách hàng. Công ty đó đã chuẩn bị 1 chiếc hộp đựng n mảnh giấy, mỗi mảnh giấy được bí mật ghi 1 mã hóa gồm nhiều ký tự số & chữ. Mỗi khách hàng được chọn 1 mảnh giấy trong chiếc hộp đó. Viết chương trình tặng quà từ thấp đến cao theo số lượng các ký tự số của mã hóa trong tờ giấy, nếu số lượng ký tự số trong mã hóa bằng nhau thì khách hàng chọn trước được tặng quà trước.*

- Input. *Dòng đầu tiên chứa số nguyên dương $n \in \mathbb{N}^\star$, $1 \leq n \leq 10^4$. n dòng tiếp theo, mỗi dòng chứa 1 mã hóa không dài quá 255 ký tự tương ứng cho từng khách hàng.*
- Output. *Thứ tự tặng quà của trò chơi này cho n khách hàng trên.*
- Sample.

game_gift.inp	game_gift.out
5 N123456Cao A89Dat G2Chuc L512Ket E3689Qua	G2Chuc A89Dat L512Ket E3689Qua N123456Cao

113 ([Tru23], 4., pp. 21–22, HSG Lớp 12 Nam Định 2020–2021, Work – Công việc). *Trong 1 dây chuyền làm việc của công ty có n công nhân làm n việc. Người ta đánh số cho công nhân từ 1 đến n theo thứ tự đứng trong dây chuyền. Thời gian hoàn thành 1 công việc của người thứ i là t_i phút. Mỗi người cần làm xong công việc của mình nhưng được quyền làm tối đa 2 việc. Vì thế họ có thể phối hợp với người đứng ngay trước mình cùng làm, nếu người thứ i & người thứ $i+1$ phối hợp thì thời gian làm xong việc cho 2 người là p_i. Tìm phương án sao cho n công việc đều hoàn thành với thời gian ít nhất.*

- Input. *Dòng 1 ghi số n, $1 < n \leq 10^6$. Dòng 2 ghi thời gian làm xong việc của từng công nhân tương ứng trong dây chuyền $t_1, t_2, \ldots, t_n$, $1 \leq t_i \leq 60$, $\forall i = 1, 2, \ldots, n$. Dòng 3 ghi $n-1$ số thời gian cùng làm tương ứng cho số cặp công nhân nếu phối hợp $p_1, p_2, \ldots, p_{n-1}$, $1 \leq p_i \leq 100$, $\forall i = 1, 2, \ldots, n$.*
- Output. *1 số duy nhất ghi tổng thời gian hoàn thành công việc ít nhất của n công nhân.*
- Sample.

work.inp	work.out
5 2 5 7 8 4 3 9 10 10	17

3 CSES Problem Set

3.1 Introductory Problems

Problem 1 (Weird Algorithm). *Consider an algorithm that takes as input a positive integer $n \in \mathbb{N}^\star$. If n is even, the algorithm divides it by 2, & if n is odd, the algorithm multiplies it by 3 & adds 1. The algorithms repeats this, until n is 1. E.g., the sequence*

for $n = 3$ is as follows: $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$. Your task is to simulate the execution of the algorithm for a given value of n.

- Input. *The only input line contains an integer $n \in \mathbb{Z}$.*
- Output. *Print a line that contains all values of n during the algorithm.*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

weird_algorithm.inp	weird_algorithm.out
3	3 10 5 16 8 4 2 1

Source: CSES Problem Set/weird algorithm, & [Laa20, Sect. 1.3, pp. 5–7].

Problem 2 (Missing Number). *You are given all numbers between $1, 2, \ldots, n$ except one. Your task is to find the missing number.*

- Input. *The 1st input line contains a positive integer $n \in \mathbb{N}^\star$. The 2nd line contains $n - 1$ numbers. Each number is distinct & between 1 & n (inclusive).*
- Output. *Print the missing number.*
- Constraints. $2 \leq n \leq 2 \cdot 10^5$.
- Sample.

missing_number.inp	missing_number.out
5 2 3 1 5	4

Source: CSES Problem Set/missing number.

Problem 3 (Repetitions). *You are given a DNA sequence: a string consisting of characters A, C, G, & T. Your task is to find the longest repetition in the sequence. This is a maximum-length substring containing only 1 type of character.*

- Input. *The only input line contains a string of $n \in \mathbb{N}^\star$ characters.*
- Output. *Print 1 integer: the length of the longest repetition.*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

repetition.inp	repetition.out
ATTCGGGA	3

Source: CSES problem Set/repetition.

Problem 4 (Nondecreasing Array). *You are given an array of n integers. You want to modify the array so that it is non-decreasing, i.e., every element is at least as large as the previous element. On each move, you may increase the value of any element by 1. What is the minimum number of moves required?*

- Input. *The 1st input line contains an integer n: the size of the array. The 2nd line contains n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*
- Output. *Print the minimum number of moves.*
- Constraints. $1 \leq n \leq 2 \cdot 10^5$, $1 \leq x_i \leq 10^9$.
- Sample.

nondecreasing_array.inp	nondecreasing_array.out
5 3 2 5 1 7	5

Source: CSES problem Set/increasing array.

Problem 5 (Permutations). *A permutation of integers* $1, 2, \ldots, n$ *is called* beautiful *if there are no adjacent elements whose difference is* 1. *Given* n, *construct a beautiful permutation if such a permutation exists.*

- Input. *The only input line contains an integer* n.
- Output. *Print a beautiful permutation of integers* $1, 2, \ldots, n$. *If there are several solutions, you may print any of them. If there are no solutions, print* `"no solution"`.
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

permutations.inp	permutations.out
5	4 2 5 3 1
3	NO SOLUTION

Source: CSES problem Set/permutations.

Problem 6 (Number Spiral). *A* number spiral *is an infinite grid whose upper-left square has number* 1. *Here are the 1st 5 layers of the spiral:*

1	2	9	10	25
4	3	8	11	24
5	6	7	12	23
16	15	14	13	22
17	18	19	20	21

Your task is to find out the number in row y *&* *column* x.

- Input. *The 1st input line contains an integer* t: *the number of tests. After this, there are* t *lines, each containing integers* y, x.
- Output. *For each test, print the number in row* y *&* *column* x.
- Constraints. $1 \leq t \leq 10^5$, $1 \leq x, y \leq 10^9$.
- Sample.

number_spiral.inp	number_spiral.out
3 2 3 1 1 4 2	8 1 15

Source: CSES problem Set/number spiral.

Problem 7 (2 Knights). *Your task is to count for* $k = 1, 2, \ldots, n$ *the number of ways 2 knights can be placed on a* $k \times k$ *chessboard so that they do not attack each other.*

- Input. *The only input line contains a positive integer* $n \in \mathbb{N}^\star$.
- Output. *Print* n *integers: the results.*
- Constraints. $1 \leq n \leq 10^4$.
- Sample.

knight.inp	knight.out
8	0 6 28 96 252 550 1056 1048

Source: CSES problem Set/2 knights.

Problem 8 (2 Sets). *Your task is to divide the numbers* $1, 2, \ldots, n$ *into 2 sets of equal sum.*

- Input. *The only input line contains a positive integer* $n \in \mathbb{N}^\star$.
- Output. *Print "yes", if the division is possible, & "no" otherwise. After this, if the division is possible, print an example of how to create the sets. 1st, print the number of elements in the 1st set followed by the elements themselves in a separate line, & then, print the 2nd set in a similar way.*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

set2.inp	set2.out
7	yes 4 1 2 4 7 3 3 5 6
6	no

Source: CSES problem Set/2 sets.

Problem 9 (Bit String). *Your task is to calculate the number of bit strings of length* n. *E.g., if* $n = 3$, *the correct answer is* 8, *because the possible bit strings are* `000`, `001`, `010`, `011`, `100`, `101`, `110`, *&* `111`.

- Input. *The only input line contains a positive integer* $n \in \mathbb{N}^\star$.
- Output. *Print the result modulo* $10^9 + 7$.
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

bit_string.inp	bit_string.out
3	8

Source: CSES problem Set/bit string.

Problem 10 (Trailing Zeros). *Your task is to calculate the number of trailing zeros in the factorial* $n!$, *e.g.,* $20! = 2432902008176640000$ *& it has* 4 *trailing zeroes.*

- Input. *The only input line contains a positive integer* $n \in \mathbb{N}^\star$.
- Output. *Print the number of trailing zeros in* $n!$.
- Constraints. $1 \leq n \leq 10^9$.
- Sample.

trailing_zero.inp	trailing_zero.out
20	4

Source: CSES problem Set/trailing zeroes.

Problem 11 (Coin Piles). *You have 2 coin piles containing* a *&* b *coins. On each move, you can either remove 1 coin from the left pile & 2 coins from the right pile, or 2 coins from the left pile & 1 coin from the right pile. Your task is to efficiently find out if you can empty both the piles.*

- Input. *The 1st input line has an integer* t: *the number of tests. After this, there are* t *lines, each of which has 2 integers* a *&* b: *the numbers of coins in the piles.*
- Output. *For each test, print "yes" if you can empty the piles & "no" otherwise.*
- Constraints. $1 \leq t \leq 10^5$, $0 \leq a, b \leq 10^9$.

- Sample.

max_gift.inp	max_gift.out
3 2 1 2 2 3 3	yes no yes

Source: CSES problem Set/coin piles.

Problem 12 (Palindrome Reorder). *Given a string, your task is to reorder its letters in such a way that it becomes a palindrome (i.e., it reads the same towards & backwards).*

- Input. *The only input line has a string of length n consisting of characters A-Z.*
- Output. *Print a palindrome consisting of the characters of the original string. You may print any valid solution. If there are no solutions, print "no solution".*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

palindrome_reorder.inp	palindrome_reorder.out
AAAACACBA	AACABACAA

Source: CSES problem Set/palindrome reorder.

Problem 13 (Gray Code). *A Gray code is a list of all 2^n bit strings of length n, where any 2 successive strings differ in exactly 1 bit (i.e., their Hamming distance is 1). Your task is to create a Gray code for a given length n.*

- Input. *The only input line has a positive integer $n \in \mathbb{N}^\star$.*
- Output. *Print 2^n lines that describe the Gray code. You can print any valid solution.*
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

gray_code.inp	gray_code.out
2	00 01 11 10

Source: CSES problem Set/gray code.

Problem 14 (Tower of Hà Nội). *The Tower of Hanoi game consists of 3 stacks (left, middle, & right) & n round disks of different sizes. Initially, the left stack has all the disks, in increasing order of size from top to bottom. The goal is to move all the disks to the right stack using the middle stack. On each move you can move the uppermost disk from a stack to another stack. In addition, it is not allowed to place a larger disk on a smaller disk. Your task is to find a solution that minimizes the number of moves.*

- Input. *The only input line has a positive integer$n \in \mathbb{N}^\star$: the number of disks.*
- Output. *1st print a positive integer $k \in \mathbb{N}^\star$: the minimum number of moves. After this, print k lines that describe the moves. Each line has 2 integers $a, b \in \{1, 2, 3\}$: you move a disk from stack a to stack b.*
- Constraints. $1 \leq n \leq 16$.
- Sample.

.inp	.out
2	3 1 2 1 3 2 3

Source: CSES problem Set/tower of Hà Nội.

Problem 15 (Creating Strings)**.** *Given a string, your task is to generate all different strings that can be created using its characters.*

- Input. *The only input line has a string of length $n \in \mathbb{N}^\star$. Each character is between a–z.*
- Output. *1st print a positive integer $k \in \mathbb{N}^\star$: the number of strings. Then print k lines: the strings in alphabetical order.*
- Constraints. $1 \leq n \leq 8$.
- Sample.

create_string.inp	create_string.out
aabac	20 aaabc aaacb aabac aabca aacab aacba abaac abaca abcaa acaab acaba acbaa baaac baaca bacaa bcaaa caaab caaba cabaa cbaaa

Source: CSES problem Set/creating strings.

Problem 16 (Apple Division)**.** *There are n apples with known weights. Your task is to divide the apples into 2 groups so that the difference between the weights of the groups is minimal.*

- Input. *The first input line has an integer $n \in \mathbb{N}^\star$: the number of apples. The next line has n integers $p_1, p_2, \ldots, p_n$: the weight of each apple.*
- Output. *Print 1 integer: the minimum difference between the weights of the groups.*
- Constraints. $1 \leq n \leq 20$, $1 \leq p_i \leq 10^9$.
- Sample.

apple_division.inp	apple_division.out
5 3 2 7 4 1	1

Explanation: Group 1 has weights $2, 3, 4$ (total weight 9), & group 2 has weights 1 & 7 (total weight 8).

Source: CSES problem Set/apple division.

Problem 17 (Chessboard & Queens)**.** *Your task is to place 8 queens on a chessboard so that no 2 queens are attacking each other. As an additional challenge, each square is either free or reserved, & you can only place queens on the free squares. However, the reserved squares do not prevent queens from attacking each other. How many possible ways are there to place the queens?*

- Input. *The input has 8 lines, and each of them has 8 characters. Each square is either free (.) or reserved (*).*
- Output. *Print 1 integer: the number of ways you can place the queens.*
- Sample. *Input:*

```
........
........
..*.....
........
........
.....**.
...*....
........
```

Output: 65

Source: CSES problem Set/chessboard & queens.

Problem 18 (Digit Queries). *Consider an infinite string that consists of all positive integers in increasing order:*

$$12345678910111213141516171819202122232425262728293 0\ldots$$

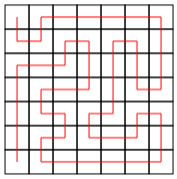
Your task is to process q queries of the form: what is the digit at position k in the string?

- Input. *The 1st input line has a positive integer $q \in \mathbb{N}^\star$: the number of queries. After this, there are q lines that describe the queries. Each line has an integer k: a 1-indexed position in the string.*
- Output. *For each query, print the corresponding digit.*
- Constraints. $1 \leq q \leq 1000$, $1 \leq k \leq 10^{18}$.
- Sample.

digit_query.inp	digit_query.out
3	7
7	4
19	1
12	

Source: CSES problem Set/digit queries.

Problem 19 (Grid Path). *There are 88418 paths in a 7×7 grid from the upper-left square to the lower-left square. Each path corresponds to a 48-character description consisting of characters* `D` *(down),* `U` *(up),* `L` *(left), & * `R` *(right). E.g., the path*



corresponds to the description `DRURRRRRDDDLUULDDDLDRRURDDLLLLLURULURRUULDLLDDDD`*. You are given a description of a path which may also contain characters* `?` *(any direction). Your task is to calculate the number of paths that match the description.*

- Input. *The only input line has a 48-character string of characters* `?`*,* `D`*,* `U`*,* `L`*,* `R`*.*
- Output. *Print 1 integer: the total number of paths.*
- Sample.

grid_path.inp	grid_path.out
??????R??????U??????????????????????????LD????D?	201

3.2 Mathematics

Problem 20 (Josephus Queries). *Consider a game where there are n children (numbered $1, 2 \ldots, n$) in a circle. During the game, every second child is removed from the circle, until there are no children left. Your task is to process q queries of the form: "when there are n children, who is the kth child that will be removed?"*

- Input. *The 1st input line has a positive integer $q \in \mathbb{N}$: the number of queries. After this, there are q lines that describe the queries. Each line has 2 integers $n, k \in \mathbb{N}$: the number of children & the position of the child.*
- Output. *Print q integers: the answer for each query.*

- Constraints. $1 \le q \le 10^5$, $1 \le k \le n \le 10^9$.
- Sample.

Josephus_query.inp	Josephus_query.out
4 7 1 7 3 2 2 1337 1313	2 6 1 1107

4 Notes on Commands

4.1 Notes on C/C++ commands

Template:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
        ios::sync_with_stdio(0);
        cin.tie(0);
        // Input & Output
        freopen("prob.inp", "r", stdin);
        freopen("prob.out", "w", stdout);
}
// Terminal: g++ -std=c++11 -O2 -Wall test.cpp -o test
```

4.2 Notes on Pascal commands

4.3 Notes on Python commands

- Để sử dụng các hàm toán học trong Python, cần import thư viện `math` vào chương trình: `from math import *`
- Để mở file dữ liệu vào `prob.inp` chỉ để đọc dữ liệu & mở file dữ liệu ra `prob.out` để thay đổi dữ liệu trong file: `file = open("prob.inp")` & `file2 = open("prob.out", "w")`.
- Sắp xếp trong Python có thể thực hiện 1 cách đơn giản nhờ phương thức `sort()` hoặc `sorted()`. Cú pháp:

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

- Trong Python, để lấy giá trị ngẫu nhiên, sử dụng phương thức `ranint(a, b)` với `a, b` là giới hạn của giá trị cần lấy ngẫu nhiên.
- Cf.: `**0.5` or `**(1/2)` vs. `math.sqrt(), cmath.sqrt()`:
  - Python Square Root: Real & Complex – Calculating the Square Root in Python.[13]
  - StackOverflow/which is faster in Python: x**.5 or math.sqrt(x)?.[14]

5 Problems in Elementary Mathematics – Bài Toán Tin Học Trong Toán Học Sơ Cấp

114 (Even vs. odd)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xét tính chẵn lẻ của* $n \in \mathbb{Z}$ *được nhập từ bàn phím.*

Pascal: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Pascal/even_odd.pas.

```
program even_odd;
var n : integer;
begin
write('Enter n = '); readln(n);
if n mod 2 = 0 then writeln(n,' is even') else writeln(n, ' is odd');
end.
```

[13] https://www.learndatasci.com/solutions/python-square-root/.
[14] https://stackoverflow.com/questions/327002/which-is-faster-in-python-x-5-or-math-sqrtx.

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/even_odd.py

```
        n = int(input())
        if n % 2 == 0:
        print(n, " is even")
        else:
        print(n, " is odd")
```

C: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/even_odd.c.

```
        #include <stdio.h>
        int main() {
                int n;
                scanf("%d", &n);
                // 1st solution
                if (n % 2 == 0) printf("%d is even.\n", n);
                else printf("%d is odd.\n", n);
                // 2nd solution
                (n % 2 == 0) ? printf("%d is even.\n", n) : printf("%d is odd.\n", n);
                // 3rd solution
                if ((n & 1) == 0) printf("%d is even.\n", n);
                else printf("%d is odd.\n", n);
                return 0;
        }
        // Terminal: g++ -std=c++11 -O2 -Wall even_odd.c -o even_odd
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/even_odd.cpp.

```
        #include <iostream>
        using namespace std;
        int main() {
                int n;
                cin >> n;
                // 1st solution
                if (n % 2 == 0) cout << n << " is even.\n";
                else cout << n << " is odd.\n";
                // 2nd solution
                (n % 2 == 0) ? cout << n << " is even.\n" : cout << n << " is odd.\n";
                // 3rd solution using bitwise AND
                if ((n & 1) == 0) cout << n << " is even.\n";
                else cout << n << " is odd.\n";
                return 0;
        }
        // Terminal: g++ -std=c++11 -O2 -Wall even_odd.cpp -o even_odd
```

115 (Divisible by)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để kiểm tra liệu* $a \vdots b$ *hay không, với* $a, b \in \mathbb{Z}$ *được nhập từ bàn phím & in ra số dư* r *trong phép chia* a *cho* b*.*

Pascal: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Pascal/divisible_by.pas.

```
        program divisible_by;
        var a, b : integer;
        begin
        write('Enter a = ');
        readln(a);
        write('Enter b = ');
        readln(b);
        if a mod b = 0 then writeln(a,' is divisible by ', b, '.')
        else writeln(a,' is not divisible by ', b, '.');
        end.
```

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/divisible_by.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/divisible_by.py.

```
        a = int(input())
        b = int(input())
        if a % b == 0:
        print(a, ' is divisible by ', b)
        else:
        print(a, ' is not divisible by ', b)
```

C: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/divisible_by.c.

```
        #include <stdio.h>
        int main() {
                int a, b;
                scanf("%d", &a);
                scanf("%d", &b);
                if (a % b == 0) printf("%d is divisible by %d.\n", a, b);
                else printf("%d is not divisible by %d.\n", a, b);
                return 0;
        }
        // Terminal: g++ -std=c++11 -O2 -Wall divisible_by.c -o divisible_by
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/divisible_by.cpp.

```
        #include <iostream>
        using namespace std;
        int main() {
                int a, b;
                cin >> a >> b;
                if (a % b == 0) cout << a << " is divisible by " << b << ".\n";
                else cout << a << " is not divisible by " << b << ".\n";
                return 0;
        }
        // Terminal: g++ -std=c++11 -O2 -Wall divisible_by.cpp -o divisible_by
```

116 (Triangle)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để liệu* a, b, c *có phải là độ dài của: (a) 1 tam giác. (b) 1 tam giác nhọn. (c) 1 tam giác vuông. (d) 1 tam giác tù.*

Pascal:

- Python script: GitHub/NQBH/hobby/elementary computer science/Python/triangle.

```
        try:
        a,b,c = float(input('Nhap a = ')), float(input('Nhap b = ')), float(input('Nhap c = '))
        if (a + b > c) and (a + c > b) and  (b + c > a):
        loai_tg = ''
        if (a*a + b*b == c*c) or (a*a + c*c == b*b) or (b*b + c*c == a*a):
        loai_tg = 'vuong'
        elif (a*a + b*b > c*c) and (a*a + c*c > b*b) and (b*b + c*c > a*a):
        loai_tg = 'nhon'
        else:
        loai_tg = 'tu'
        print('Day la do dai 3 canh cua 1 tam giac ' + loai_tg)
        else:
        print('Day khong la do dai 3 canh cua 1 tam giac.')
        except:
        print('Day khong phai la 1 so.')
```

117 (A special cubic equation – 1 dạng phương trình bậc 3 đặc biệt)**.** *(a) Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc 3 có dạng đặc biệt* $x^3 + 3a^2x + 2b = 0$ *(khuyết số hạng* x^2*), với* $p, q \in \mathbb{R}$ *được nhập từ bàn phím. Biết phương trình này có 1 nghiệm duy nhất là*[15]*:*

$$x = \sqrt[3]{\sqrt{a^6 + b^2} - b} - \sqrt[3]{\sqrt{a^6 + b^2} + b}.$$

(b) Chứng minh chặt chẽ bằng toán học công thức nghiệm đã cho.

C: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/special_cubic_eqn.c.

[15] See. e.g., [Thư+21, pp. 68–69].

```
#include <math.h>
#include <stdio.h>
int main() {
        double a, b, x, delta, test;
        printf("a = ");
        scanf("%lf", &a);
        printf("b = ");
        scanf("%lf", &b);
        delta = pow(a, 6.0) + pow(b, 2.0);
        delta = sqrt(delta);
        x = pow(delta - b, 1.0/3) - pow(delta + b, 1.0/3);
        printf("Root x = %lf.\n", x);
        test = x*x*x + 3*a*a*x + 2*b;
        printf("x^3 + 3a^2x + 2b = %lf.\n", test);
}
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/special_cubic_eqn.cpp.

```
#include <cmath>
#include <iostream>
using namespace std;
int main() {
        double a, b, x, delta;
        cout << "a = ";
        cin >> a;
        cout << "b = ";
        cin >> b;
        delta = pow(a, 6.0) + pow(b, 2.0);
        delta = sqrt(delta);
        x = pow(delta - b, 1.0/3) - pow(delta + b, 1.0/3);
        cout << "Root x = " << x << ".\n";
        double test = x*x*x + 3*a*a*x + 2*b;
        cout << "x^3 + 3a^2x + 2b = " << test << ".\n";
}
```

118 (Polynomial equation – Phương trình đa thức)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc nhất, bậc 2, bậc 3, & bậc 4 với các hệ số thực được nhập từ bàn phím.*

119 (Fibonacci sequence)**.** *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xuất ra màn hình, với $n \in \mathbb{N}$ được nhập từ bàn phím: (a) Số Fibonacci thứ n. (b) n số Fibonacci đầu tiên.*

120 (1st n square roots)**.** *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 2 của n số tự nhiên đầu tiên với $n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

121 (Số chính phương – Square number)**.** *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số $n \in \mathbb{N}^\star$ được nhập từ bàn phím có phải là số chính phương hay không.*

122 (1st n cube roots)**.** *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 3 của n số tự nhiên đầu tiên với $n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

123. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số $n \in \mathbb{N}^\star$ được nhập từ bàn phím có phải là lập phương của 1 số tự nhiên hay không.*

124 (1st n nth roots)**.** *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc n của m số tự nhiên đầu tiên với $m, n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

125. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số $m \in \mathbb{Z}$ được nhập từ bàn phím có phải là lũy thừa bậc $n \in \mathbb{N}$ của 1 số tự nhiên hay không với m, n được nhập từ bàn phím.*

126 ([Đức22], 1., p. 18, Thay chữ số – Digit replacement)**.** *Lập trình vào số nguyên $n \in \mathbb{Z}$, thực hiện thay thế các chữ số 0 trong biểu diễn thập phân của n thành các chữ số 5 & in ra kết quả.*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên T cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa 1 số nguyên $n \in \mathbb{Z}$.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, chương trình cần in ra số n sau khi thay thế các chữ số của n theo yêu cầu đề bài.*
- Constraint. $1 \leq T \leq 10^5$, $0 \leq n \leq 10^{12}$.

- Sample.

digit_replacement.inp	digit_replacement.out
2 1005 1234	1555 1234

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/Duc_200_BTLT_Python/digit_replacement.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/digit_replacement.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/digit_replacement.py.
- Input: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/digit_replacement.inp.
- Output: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/digit_replacement.out.

1st solution [Đức22, 1., p. 202]:

```
def convert5(n):
if n == 0:
return 5
y = [c for c in n]
for i in range(len(y)):
if y[i] == '0':
y[i] = '5'
return ''.join(y)
t = int(input())
while t != 0:
t -= 1
n = input()
print(convert5(n))
```

2nd solution:

```
def digit_replacement(n):
ans = ''
for i in range(len(n)):
if n[i] == '0':
ans += '5'
else:
ans += n[i]
return ans
t = int(input())
for _ in range(t):
n = input()
print(digit_replacement(n))
```

3rd solution:

```
t = int(input())
A = []
for _ in range(t):
s = input()
A += [s.replace('0', '5')]
print('\n'.join(A))
```

Nhận xét 1. *Bài toán trên có thể mở rộng cho việc thay chữ số* 0 *thành* 5 *thành việc thay chữ số* a *thành chữ số* b *với* $a, b \in \{0, 1, 2, \dots, 9\}$ *là các chữ số bất kỳ, chỉ cần đổi các dòng lệnh tương ứng thành:*

```
if n[i] == '<a>':
ans += '<b>'
```

```
if n[i] == '<a>':
ans += '<b>'
```

```
A += [s.replace('<a>', '<b>')]
```

trong đó `<a>`, `<b>` *lần lượt là 2 giá trị của cặp chữ số cần đổi thành:* $a \to b$. *Tương tự, có thể thay 1 bộ nhiều chữ số thành 1 bộ chữ số khác, i.e.,* $(a_1, a_2, \dots, a_n) \to (b_1, b_2, \dots, b_n)$, *với* $n \in \mathbb{N}$, $a_i, b_i \in \{0, 1, 2, \dots, 9\}$, $\forall i = 1, 2, \dots, n$ *thay vì chỉ 1 chữ số.*

127 ([Đức22], 2., pp. 18–19, Dice – Xúc xắc)**.** *Bạn được tặng 1 con xúc xắc hình lập phương với* 6 *mặt. Mỗi mặt của con xúc xắc có in 1 số chấm, số lượng chấm* $\in \{1, 2, 3, 4, 5, 6\}$. *Bạn được biết số chấm trên 1 mặt của xúc xắc, nhiệm vụ là đoán số ở mặt đối diện của xúc xắc.*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên dương* $t \in \mathbb{N}^*$ *cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa 1 số nguyên* $n \in \mathbb{Z}$ *cho biết số chấm trên 1 mặt của con xúc xắc.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, in ra số ở mặt đối diện tương ứng.*
- Constraint. $1 \le t \le 500$, $1 \le n \le 6$.

- Sample.

dice.inp	dice.out
2 6 2	1 5

Tổng số chấm trên 2 mặt đối nhau của 1 con xúc xắc luôn bằng 7, nên đối diện với mặt có số chấm là $n \in \{1, 2, 3, 4, 5, 6\}$ là mặt có số chấm $7 - n \in \{1, 2, 3, 4, 5, 6\}$.

Pascal:

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/dice.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/Duc_200_BTLT_Python/dice.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/dice.py.
- Input: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/dice.inp.
- Output: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/dice.out.

1st solution [Đức22, p. 202–203]:

```
t = int(input())
while t != 0:
t -= 1
n = int(input())
print(7 - n)
```

2nd solution:

```
t = int(input())
for _ in range(t):
n = int(input())
print(7 - n)
```

128 ([Đức22], 3., pp. 19–20, Doctor appointment – Lịch khám bệnh). *Có $n \in \mathbb{N}^*$ bệnh nhân đến khám bệnh tại phòng khám. Giả sử cứ sau x phút thì lại có 1 bệnh nhân mới đến phòng khám. Ngoài ra, bác sĩ sẽ chỉ dành 10 phút để khám cho mỗi bệnh nhân. Tính toán thời gian bằng phút mà bệnh nhân cuối cùng cần phải chờ đến lượt mình được bác sĩ khám bệnh.*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên dương $t \in \mathbb{N}^*$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa 2 số nguyên $n, x \in \mathbb{Z}$.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, in ra 1 dòng chứa số m là số phút mà bệnh nhân cuối cùng cần chờ bác sĩ tại phòng khám.*
- Constraint. $1 \le t \le 500$, $1 \le n \le 100$, $0 \le x \le 30$.
- Sample.

doctor_appointment.inp	doctor_appointment.out
5 4 5 5 3 6 5 7 6 8 2	15 28 25 24 56

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/doctor_appointment.py.

1st solution [Đức22, 3., p. 203]

```
t = int(input())
while t != 0:
t -= 1
n, x = map(int, input().split())
ans = x + 10*(n - 1) - x*n
if ans < 0:
ans = 0
print(ans)
```

2nd solution:

```
t = int(input())
while t != 0:
t -= 1
n, x = map(int, input().split())
if x >= 10:
print(0)
else:
print((n - 1)*(10 - x))
```

129 ([Đức22], 4., p. 20, Bit even, bit odd – Tính chẵn lẻ nhị phân). *Cho số nguyên không dấu $n \in \mathbb{N}$. Kiểm tra "tính chẵn lẻ" của n, với* tính chẵn lẻ *của n là số bit 1 trong biểu diễn nhị phân của n, i.e., nếu trong biểu diễn nhị phân của n có chứa 1 số chẵn các bit 1 thì n được coi là có tính chẵn, & ngược lại n có tính lẻ.*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên dương $t \in \mathbb{N}^\star$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa số nguyên $n \in \mathbb{N}$.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, in ra 1 dòng chứa thông báo* `"odd"` *nếu n có tính lẻ &* `"even"` *nếu n có tính chẵn.*
- Constraint. $1 \leq t \leq 500$, $1 \leq n \leq 10^{12}$.
- Sample.

bit_even_odd.inp	bit_even_odd.out
2 13 9	odd even

Đổi n sang biểu diễn nhị phân, đếm các số 1.
Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/bit_even_odd.py.

1st solution [Đức22, 4., p. 204]:

```
t = int(input())
while t != 0:
t -= 1
s = 0
n = int(input())
while n != 0:
s += n % 2
n //= 2
if s % 2 != 0:
print("odd")
else:
print("even")
```

2nd solution:

```
t = int(input())
while t != 0:
t -= 1
s = 0
n = bin(int(input()))
for c in n:
if c == '1':
s += 1
if s % 2 != 0:
print("odd")
else:
print("even")
```

130 ([Đức22], 5., pp. 20–21, Message – Nhắn tin)**.** *Có $n \in \mathbb{N}^\star$ học sinh trong 1 lớp học, mỗi người nghĩ ra 1 câu chuyện hài hước khác nhau. Trong 1 giờ vắng giáo viên, họ quyết định nghĩa ra 1 trò chơi để giết thời gian. Họ muốn chia sẻ những câu chuyện hài hước với nhau bằng cách gửi tin nhắn điện tử. Giả sử 1 người luôn gửi tất cả những câu chuyện hài hước mà người ấy biết tại thời điểm tin nhắn được gửi & 1 tin nhắn chỉ có thể gửi đến 1 người nhận. Số lượng tin nhắn tối thiểu họ cần gửi là bao nhiêu để đảm bảo tất cả n người đều nhận được tất cả các câu chuyện?*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên dương $t \in \mathbb{N}^\star$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa số nguyên $n \in \mathbb{N}$.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, in ra 1 dòng chứa số lượng tin nhắn cần gửi để n học sinh đều nhận được tất cả n câu chuyện hài.*
- Constraint. $1 \leq t \leq 100$, $1 \leq n \leq 10^5$.
- Sample.

message.inp	message.out
1 2	2

Pascal:
Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/message.py.

```
t = int(input())
for _ in range(t):
n = int(input())
print(2*n - 2)
```

C++:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/message.cpp.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/HTP/message.cpp.
- Input: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/message.inp.
- Output: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/message.out.

1st solution:

```
#include <iostream>
using namespace std;

int main() {
        int t, n;
        cin >> t;
        int A[t];
        for (int i = 0; i < t; ++i) {
                cin >> n;
                A[i] = n*(n - 1);
                cout << A[i] << '\n';
        }
}
```

2nd solution:

```
#include <iostream>
using namespace std;

int main() {
        int t, n;
        cin >> t;
        while (t--) {
                cin >> n;
                cout << 2*n - 2 << '\n'
        }
}
```

131 ([Đức22], 6., pp. 21–22, Triangle number – Số tam giác). *1 số được gọi là* số tam giác *nếu ta có thể biểu diễn nó dưới dạng lưới hình tam giác gồm các điểm sao cho các điểm tạo thành 1 tam giác đều, i.e., hàng đầu tiên có 1 điểm, hàng thứ 2 có 2 điểm, ... hàng thứ i có i điểm. Các số tam giác đầu tiên là $1, 3 = 1 + 2, 6 = 1 + 2 + 3, 10 = 1 + 2 + 3 + 4$. Cho số nguyên dương $n \in \mathbb{N}^\star$, kiểm tra xem N có phải là số tam giác hay không.*

- Input. *Dòng đầu tiên của đầu vào chứa số nguyên dương $t \in \mathbb{N}^\star$ cho biết số bộ dữ liệu cần kiểm tra. Mỗi bộ dữ liệu gồm 1 dòng chứa số nguyên $n \in \mathbb{N}$.*
- Output. *Ứng với mỗi bộ dữ liệu đầu vào, in ra 1 dòng chứa số 1 nếu n là số tam giác, ngược lại thì in ra số 0.*
- Constraint. $1 \leq t \leq 100$, $1 \leq n \leq 10^7$.
- Sample.

triangle_number.inp	triangle_number.out
5	1
3	0
4	1
6	1
55	0
345	

See, e.g., Wikipedia/số tam giác or Wikipedia/triangular number.
Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/Duc_200_BTLT_Python/triangle_number.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/triangle_number.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/triangle_number.py.
- Input: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/triangle_number.inp.
- Output: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/triangle_number.out.

1st solution [Đức22, 6., pp. 204–205]:

```
from math import sqrt, floor
t = int(input())
for _ in range(t):
n = int(input())
a, b = 1, 1
c = -2*n
d = b*b - 4*a*c
x1 = (-b + sqrt(d))/(2*a)
x2 = (-b - sqrt(d))/(2*a)
if x1 > 0 and floor(x1) == x1:
print('1')
continue
if x2 > 0 and floor(x2) == x2:
print('1')
continue
print('0')
```

2nd solution:

```
t = int(input())
for _ in range(t):
n = int(input())
i = 1
sum = 0
while sum < n:
sum += i
i += 1
if n == sum:
print(1)
else:
print(0)
```

3rd solution:

```
from math import sqrt
def is_triangle_number(x):
x = x*8 + 1
if int(sqrt(x))**2 != x:
return 0
return int(sqrt(x)) % 2
t = int(input())
A = []
for _ in range(t):
n = int(input())
A += [str(is_triangle_number(n))]
print('\n'.join(A))
```

4th solution:

```
from math import sqrt, floor
t = int(input())
for _ in range(t):
n = int(input())
k = floor(sqrt(2*n))
if n == k*(k + 1)/2:
print(1)
else:
print(0)
```

132. *Cho $n \in \mathbb{N}^\star$ nhập từ bàn phím. (a) In số tam giác thứ n. (b) In n số tam giác đầu tiên. (c) Cho $a, b \in \mathbb{R}$ nhập từ bàn phím, in tất cả các số tam giác thuộc đoạn $[a, b]$.*

5.1 Algebraic Expression – Biểu Thức Đại Số

133 ([Vie21], 1., p. 15, Vũng Tàu 2020). *Cho $a, b, c \in \mathbb{N}^\star$.* Yêu cầu: *Tính giá trị của biểu thức $S = \dfrac{a^2+b^2+c^2}{abc} + \sqrt{abc}$.*

- Input. *File* `algebraic_expression.inp` *chứa 3 số nguyên dương a, b, c. Mỗi số trên 1 dòng.*
- Kết quả: *Ghi vào File* `algebraic_expression.out` *kết quả S tính được (làm tròn lấy 2 chữ số sau phần thập phân). E.g.:*

algebraic_expression.inp	algebraic_expression.out
2 1 2	4.25

Python script:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/algebraic_expression.py.

Python:

```
from math import sqrt
def algebraic_expression(a, b, c):
return (a**2 + b**2 + c**2)/(a*b*c) + sqrt(a*b*c) # function f(a,b,c) can be modified
a = int(input())
b = int(input())
c = int(input())
print(algebraic_expression(a,b,c))
```

C:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/algebraic_expression.c.

See also C-code for $f(x, y) = x + \sqrt{1 + y^2}$:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/sqrt_algebraic_expression.c.

```
#include <math.h>
#include <stdio.h>
int main() {
        double a, b, c;
        scanf("%lf %lf %lf", &a, &b, &c);
        printf("%.2f.\n", (a*a + b*b + c*c)/(a*b*c) + sqrt(a*b*c));
        return 1;
}
```

C++:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/algebraic_expression.cpp.

  See also C-code for $f(x,y) = x + \sqrt{1+y^2}$:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/sqrt_algebraic_expression.cpp.

```
#include <cmath>
#include <iostream>
using namespace std;
int main() {
        double a, b, c;
        cin >> a >> b >> c;
        printf("%.2f\n", (a*a + b*b + c*c)/(a*b*c) + sqrt(a*b*c));
        return 1;
}
```

Lưu ý 1. *Tương tự, ta có thể tính hầu như bất kỳ hàm số $f(a,b,c)$ 3 biến a,b,c với f là 1 hàm số có thể viết được nhờ thư viện* `math` *của Python. Tổng quát hơn, ta có thể tính bất kỳ hàm số nhiều biến $f(x_1,x_2,\dots,x_n)$ với x_i, $i=1,2,\dots,n$, $n\in\mathbb{N}^\star$ là các biến, với f là 1 hàm số có thể viết được nhờ thư viện* `math` *của Python.*

134 ([Vie21], 2., p. 19, Bắc Giang 2020)**.** *Nhà An có 1 trang trại rộng lớn. Do sở thích của An nên bố An chỉ nuôi gà & chó. 1 hôm bố An đố con gái nhà mình nuôi bao nhiêu gà, bao nhiêu chó? Bố An cho biết nhà có tổng số gà & chó là x con. Do số lượng nhiều & khó đếm từng loại nên An chỉ đếm được tổng số chân của gà & chó là y chân. Giúp An trả lời câu đố.*

- Input. *Dọc từ file văn bản* `toanco.inp` *gồm 2 số nguyên dương x,y trên 1 dòng. 2 số cách nhau 1 khoảng trống ($x\leq 10^5$, $y\leq 4\cdot 10^5$).*
- Kết quả: *Ghi ra file văn bản* `toanco.out` *gồm 2 số tương ứng là số gà & số chó tìm được. 2 số cách nhau 1 khoảng trống. Giả sử bài toán luôn có nghiệm.*

toanco.inp	toanco.out
36 100	22 14

Python script: GitHub/NQBH/hobby/elementary computer science/Python/toanco.py[16]. Input: toanco.inp. Output: toanco.out.

```
file_in = open("toanco.inp")
file_out = open("toanco.out", "w")
s = file_in.readline()
s = s.split()
x = int(s[0])
y = int(s[1])
a = int(2*x - y/2)
b = int(y/2 - x)
file_out.write(str(a) + " " + str(b))
file_in.close()
file_out.close()
```

135 ([Vie21], 4., p. 26, Quãng Ngãi 2020, Lãi suất– Interest rate)**.** *1 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).*

- Yêu cầu: *Tính số tiền người đó nhận được sau t tháng. Biết 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.*
- Input. *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 1 ký tự trắng.*
- Output. *Tệp văn bản* `bl2.out` *ghi 1 số là số tiền nhận được (làm tròn đến 1 số lẻ sau dấu chấm thập phân). E.g.:*

bl2.inp	bl2.out
12 13 100 1.0 0.2	112.9
0 10 100 1.0 0.2	102.0

[16] URL: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/toanco.py.

5.2 Number Theory – Số Học

136 ([Vie21], 3., p. 20, Yên Bái 2020, Tổng nguyên tố)**.** *Viết chương trình nhập vào 2 số nguyên* $a, b \in \mathbb{Z}$, $0 < a < b$. *(a) Tìm & tính tổng các số nguyên tố của dãy số từ* a *đến* b. *(b) Xuất ra màn hình các số chia hết cho* 5 *của dãy số từ* a *đến* b. *(c)* (Bội của $n \in \mathbb{N}^\star$) *Xuất ra màn hình các số chia hết cho* n *của dãy số từ* a *đến* b *với* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím. E.g., nhập* $a = 6$, $b = 22$. *Kết quả tổng các số nguyên tố trong dãy số từ* 6 *đến* 22*:* $7 + 11 + 13 + 17 + 19 = 67$. *Các số chia hết cho* 5 *của dãy số từ* 6 *đến* 22*:* $10, 15, 20$.

137 ([Vie21], 4., p. 22, Hải Dương 2020, Số mạnh mẽ)**.** Số mạnh mẽ *là số khi nó chia hết cho 1 số nguyên tố thì cũng chia hết cho cả bình phương của số nguyên tố đó, i.e.,* $a \in \mathbb{N}^\star$ *là số mạnh mẽ* $\Leftrightarrow (a \vdots p \Rightarrow a \vdots p^2, \forall p$*: prime). E.g.,* 25 *là số mạnh mẽ, vì nó chia hết cho số nguyên tố* 5 *& chia hết cho cả* $5^2 = 25$. *Viết chương trình liệt kê các số mạnh mẽ không vượt quá* 1000.

See, e.g., Wikipedia/powerful number, MathWorld/powerful number.

138 ([Vie21], 5., p. 23, Việt Nam 2020, Bội chính phương)**.** *Cho 1 dãy số* A *có* n *phần tử. Tìm số nguyên dương* a *nhỏ nhất thỏa mãn:* a *là số chính phương &* a *chia hết cho tất cả các phần tử của dãy số* A.

- Yêu cầu: *In ra phần dư của phép chia khi chia* a *cho* $10^9 + 7$.
- Input. *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 1 dòng được ghi cách nhau bởi dấu cách.*
- Kết quả: *Ghi ra thiết bị ra gồm 1 số nguyên duy nhất là kết quả của bài toán. E.g.:*

Dữ liệu vào	Dữ liệu ra
3 2 1 3	36

Python:

```
1
```

139 ([Vie21], 1., p. 25, Hải Dương 2020, Số hạnh phúc & số buồn bã – Happy- & sad numbers)**.** *Với 1 số nguyên dương bất kỳ, thay thế số đó bằng tổng bình phương các chữ số của nó & cứ lặp lại quá trình đó sẽ có các trường hợp sau xảy ra: Kết thúc bằng* 1 *– ta gọi số đó là* số hạnh phúc/happy number. *Kết thúc bằng* 0 *– ta gọi số đó là* số buồn bã/sad number. *Lặp lại vô hạn lần – số đó không hạnh phúc cũng không buồn bã. E.g., số* 44*: lần 1:* $4^2 + 4^2 = 32$, *lần 2:* $3^2 + 2^2 = 13$, *lần 3:* $1^2 + 3^2 = 10$, *lần 4:* $1^2 + 0^2 = 1$, *nên* 44 *là số hạnh phúc. Viết chương trình để kiểm tra xem ngày sinh của 1 người bất kỳ có phải là số hạnh phúc không?*

Python:

```
1         max_iter = 1000
2         def sum_digit_sqr(n):
3         sum = 0
4         for i in str(n):
5         sum += int(i)**2
6         return sum
7         n0 = n = int(input())
8         count = 0
9         while n > 1 and count <= max_iter:
10        n = sum_digit_sqr(n)
11        print(n)
12        count += 1
13        if n == 1:
14        print(n0, " is a happy number.")
15        elif n == 0:
16        print(n0, " is a sad number.")
17        else:
18        print(n0, " is neither a happy nor a sad number.")
```

140 ([Vie21], 2., p. 25, Gia Lai 2019, Phân số tối giản – Irreducible fraction)**.** *1 chuỗi được gọi là có dạng phân số nếu nó có dạng* `'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. E.g., Chuỗi* `'12/15'` *biểu diễn cho phân số. Dạng tối giản của phân số đó là* `'3/5'`.

Python:

```
        from math import gcd
        frac = input()
        frac = frac.split("/")
        num = int(frac[0])
        den = int(frac[1])
        gcd = gcd(num, den)
        num = int(num/gcd)
        den = int(den/gcd)
        print(str(num) + "/" + str(den))
```

141 (Tổng tất cả, tổng phần tử chẵn, lẻ, bình phương, lập phương, lũy thừa bậc n, căn bậc 2, 3, & căn bậc n, nghịch đảo, nghịch đảo bình phương, nghịch đảo lập phương, nghịch đảo lũy thừa bậc n, nghịch đảo căn bậc 2, 3, & nghịch đảo căn bậc n – Sums of all, odds, evens, squares, cubes, nth powers, square roots, cube roots, nth roots, reciprocals of square, of cubes, of nth powers, of square roots, of cube roots, of nth roots)**.** *Cho 1 dãy gồm n số nguyên: $(a_i)_{i=1}^m = a_1, a_2, \ldots, a_m$, $m \in \mathbb{N}^\star$, $a_i \in \mathbb{Z}$, $\forall i = 1, 2, \ldots, m$, mỗi số có giá trị không vượt quá 10^9.*

- Yêu cầu: *Tính tổng S tất cả các phần tử, tổng S_{even} các số chẵn, tổng S_{odd} các số lẻ, tổng S_{sqr} bình phương, tổng $S_{\text{sqr,even}}$ bình phương các số chẵn, tổng $S_{\text{sqr,odd}}$ bình phương các số lẻ, tổng S_{cb} lập phương, tổng $S_{\text{cb,even}}$ lập phương các số chẵn, tổng $S_{\text{cb,odd}}$ lập phương các số lẻ, tổng $S_{\text{pwr},n}$ lũy thừa bậc n, tổng $S_{\text{pwr,even},n}$ lũy thừa bậc n các số chẵn, tổng $S_{\text{pwr,odd},n}$ lũy thừa bậc n các số lẻ, tổng S_{sqrt} căn bậc 2, tổng $S_{\text{sqrt,even}}$ căn bậc 2 các số chẵn, tổng $S_{\text{sqrt,odd}}$ căn bậc 2 các số lẻ, tổng S_{cbrt} căn bậc 3, tổng $S_{\text{cbrt,even}}$ căn bậc 3 các số chẵn, tổng $S_{\text{cbrt,odd}}$ căn bậc 3 các số lẻ, tổng $S_{\text{rt},n}$ căn bậc n của các số, tổng $S_{\text{rt,even},n}$ căn bậc n của các số chẵn, tổng $S_{\text{rt,odd},n}$ căn bậc n của các số lẻ trong dãy $(a_i)_{i=1}^m$.*
- Dữ liệu: *Dòng đầu tiên chứa $m \in \mathbb{N}^\star$, $1 \le m \le 10^9$. Dòng thứ 2 chứa $n \in \mathbb{N}^\star$. m dòng tiếp theo, dòng thứ $i+2$ chứa a_i, $\forall i = 1, 2, \ldots, m-1$.*

Giải. Công thức toán học tính các tổng:

$$S \coloneqq \sum_{i=1}^{m} a_i = a_1 + a_2 + \cdots + a_m,\ S_{\text{even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i,$$

$$S_{\text{sqr}} \coloneqq \sum_{i=1}^{m} a_i^2 = a_1^2 + a_2^2 + \cdots + a_m^2,\ S_{\text{sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^2,\ S_{\text{sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^2,$$

$$S_{\text{cb}} \coloneqq \sum_{i=1}^{m} a_i^3 = a_1^3 + a_2^3 + \cdots + a_m^3,\ S_{\text{cb,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^3,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^3,$$

$$S_{\text{pwr},n} \coloneqq \sum_{i=1}^{m} a_i^n = a_1^n + a_2^n + \cdots + a_m^n,\ S_{\text{pwr,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^n,\ S_{\text{pwr,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^n,\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{sqrt}} \coloneqq \sum_{i=1}^{m} \sqrt{a_i} = \sqrt{a_1} + \sqrt{a_2} + \cdots + \sqrt{a_m},\ S_{\text{sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt{a_i},\ S_{\text{sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt{a_i},$$

$$S_{\text{cbrt}} \coloneqq \sum_{i=1}^{m} \sqrt[3]{a_i} = \sqrt[3]{a_1} + \sqrt[3]{a_2} + \cdots + \sqrt[3]{a_m},\ S_{\text{cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[3]{a_i},\ S_{\text{cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[3]{a_i},$$

$$S_{\text{rt},n} \coloneqq \sum_{i=1}^{m} \sqrt[n]{a_i} = \sqrt[n]{a_1} + \sqrt[n]{a_2} + \cdots + \sqrt[n]{a_m},\ S_{\text{rt,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[n]{a_i},\ S_{\text{rt,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[n]{a_i},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_m},\ S_{\text{rcpc,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i},\ S_{\text{rcpc,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i},$$

$$S_{\text{rcpc,sqr}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^2} = \frac{1}{a_1^2} + \frac{1}{a_2^2} + \cdots + \frac{1}{a_m^2},\ S_{\text{rcpc,sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^2},\ S_{\text{rcpc,sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^2},$$

$$S_{\text{rcpc,cb}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^3} = \frac{1}{a_1^3} + \frac{1}{a_2^3} + \cdots + \frac{1}{a_m^3},\ S_{\text{rcpc,cb,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^3},\ S_{\text{rcpc,cb,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^3},$$

$$S_{\text{rcpc,pwr},n} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^n} = \frac{1}{a_1^n} + \frac{1}{a_2^n} + \cdots + \frac{1}{a_m^n},\ S_{\text{rcpc,even,pwr},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{a_i^n},\ S_{\text{rcpc,odd,pwr},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^n},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc,sqrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt{a_i}} = \frac{1}{\sqrt{a_1}} + \frac{1}{\sqrt{a_2}} + \cdots + \frac{1}{\sqrt{a_m}},\ S_{\text{rcpc,sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{\sqrt{a_i}},\ S_{\text{rcpc,sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt{a_i}},$$

$$S_{\text{rcpc,cbrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[3]{a_i}} = \frac{1}{\sqrt[3]{a_1}} + \frac{1}{\sqrt[3]{a_2}} + \cdots + \frac{1}{\sqrt[3]{a_m}},\ S_{\text{rcpc,cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i \neq 0}^{m} \frac{1}{\sqrt[3]{a_i}},\ S_{\text{rcpc,cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[3]{a_i}},$$

$$S_{\text{rcpc,rt},n} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[n]{a_i}} = \frac{1}{\sqrt[n]{a_1}} + \frac{1}{\sqrt[n]{a_2}} + \cdots + \frac{1}{\sqrt[n]{a_m}},\ S_{\text{rcpc,even,rt},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt[n]{a_i}},\ S_{\text{rcpc,odd,rt},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[n]{a_i}},\ \forall n \in \mathbb{N}^\star.$$

Dựa vào các công thức này, sử dụng vòng lặp `for` hoặc `while` để tính các tổng này. □

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/compute_sums.py.

```
        m = int(input())
        A = list(map(int, input().split()))[:m]
        n = int(input('n > 0: '))
        while  n <= 0:
        n = int(input(' n > 0: '))
        S_all = sum(A)
        print(S_all)
        S_odd, S_even, S_sqr, S_cb, S_nthp, S_sqrt, S_cbrt, S_nthrt = 0, 0, 0, 0, 0, 0, 0, 0
        S_r, S_r_sqr, S_r_cb, S_r_nthp, S_r_sqrt, S_r_cbrt, S_r_nthrt = 0, 0, 0, 0, 0, 0, 0
        for i in A:
        if i % 2 == 0:
        S_even += i
        else:
        S_odd +=i
        S_sqr += i**2
        S_cb += i**3
        S_nthp += i**n
        S_sqrt += i**(1/2)
        S_cbrt += i**(1/3)
        S_nthrt += i**(1/n)
        S_r += 1/i # S_sqrt += i**-1
        S_r_sqr += 1/(i**2) # S_r_sqr += i**-2
        S_r_cb += 1/(i**3) # S_r_cb += i**-3
        S_r_nthp += 1/(i**n) # S_r_nthp += i**-n
        S_r_sqrt += 1/(i**(1/2)) # S_r_sqrt += i**(-1/2)
        S_r_cbrt += 1/(i**(1/3)) # S_r_cbrt += i**(-1/3)
        S_r_nthrt += 1/(i**(1/n)) # S_r_nthrt += i**(-1/n)
        print(f'{S_odd}, {S_even}, {S_sqr}, {S_cb}, {S_nthp}, {S_sqrt}, {S_cbrt}, {S_nthrt}, {S_r}, {S_r_sqr}
        {S_r_cb}, {S_r_nthp}, {S_r_sqrt}, {S_r_cbrt}, {S_r_nthrt}.')
```

Nhận xét 2. *Nếu chỉ tính tổng S_{odd} các số lẻ của dãy $(a_i)_{i=1}^n \subset \mathbb{Z}$ thì ta có bài toán* [Vie21, 3., p. 25, Tây Ninh 2019].

Nhận xét 3 (Mở rộng $\mathbb{Z}$ ra $\mathbb{R}, \mathbb{C}$)**.** *Các tổng $S, S_{\text{sqr}}, S_{\text{cb}}, S_{\text{pwr},n}, S_{\text{sqrt}}, S_{\text{cbrt}}, S_{\text{rt},n}, S_{\text{rcpc}}$ (i.e., các tổng không có liên quan đến tính chẵn lẻ) vẫn có thể áp dụng cho các dãy số thực thay vì chỉ cho dãy số nguyên, i.e., áp dụng cho $(a_i)_{i=1}^n \subset \mathbb{R}$, $a_i \in \mathbb{R}$, $\forall i = 1, 2, \ldots, n$, thay vì chỉ cho $(a_i)_{i=1}^n \subset \mathbb{R}$, $a_i \in \mathbb{Z}$, $\forall i = 1, 2, \ldots, n$, thậm chí có thể áp dụng cho các dãy số phức $(a_i)_{i=1}^n \subset \mathbb{C}$, $a_i \in \mathbb{C}$, $\forall i = 1, 2, \ldots, n$.*

Nhận xét 4 (Mở rộng từ dãy hữu hạn dãy vô hạn & chuỗi)**.** *Bài toán trên có thể mở rộng từ dãy hữu hạn (finite sequence) ra dãy vô hạn (infinite sequence) các số nguyên $(a_n)_{i=1}^\infty \subset \mathbb{Z}$, dãy vô hạn các số thực $(a_n)_{i=1}^\infty \subset \mathbb{R}$, & dãy vô hạn các số phức $(a_n)_{i=1}^\infty \subset \mathbb{C}$, cũng như các chuỗi (series) "xác định" (i.e., có giới hạn để có thể tính được) $S \coloneqq \sum_{i=1}^\infty a_i \in \overline{\mathbb{R}}$. Dương nhiên, 1 chương trình máy tính chỉ có thể lặp (e.g., `for, while` loops) hữu hạn lần chứ không thể lặp vô hạn lần (infinite loop error) nên ta chỉ có thể tính tổng riêng S_m của 1 chuỗi S xác định để xấp xỉ chuỗi S tới 1 độ chính xác (tolerance) nào đó (tolerance thường có dạng 10^{-N} với $N \in \mathbb{N}^\star$ thích hợp), e.g.,*

$$S_m \coloneqq \sum_{i=1}^{m} a_i \to S \coloneqq \sum_{i=1}^{\infty} a_i \text{ as } n \to \infty, \text{ i.e. } \lim_{m\to\infty} S_m = S \text{ if } S \in \overline{\mathbb{R}},$$

trong đó $\overline{\mathbb{R}} \coloneqq \mathbb{R} \cup \{\pm\infty\}$ ký hiệu tập số thực mở rộng bao gồm tập số thực, âm- & dương vô cực.

142 ([Vie21], 5., p. 26, Nghệ An 2019, Giả thuyết Goldbach cho số nguyên tố – Goldbach conjecture for primes)**.** *Cho 1 số chẵn $k \in \mathbb{N}$, $2 \leq k \leq 1000$, tìm 2 số nguyên tố sao cho tổng của chúng bằng số chẵn k đã cho. Viết chương trình* Pascal, Python, C/C++ *để trả lời câu hỏi.*

- Input. *Tệp văn bản* `prime.inp`*: Dòng đầu tiên chứa $n \in \mathbb{N}^\star$ tương ứng số test. n dòng tiếp theo, mỗi dòng chứa 1 số k, i.e., k_i, $i = 1, 2, \ldots, n$.*
- Output. *(a) Tệp văn bản* `prime.out` *gồm n dòng tương ứng n kết quả. Mỗi kết quả hiển thị tổng 2 số nguyên tố bằng số k nhập vào. (b) Gồm tất cả các biểu diễn $x + y$ với $x \leq y$.*

- Sample.

prime.inp	prime.out
2	$8 = 5 + 3$
8	$24 = 19 + 5$
24	

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/Goldbach_conjecture.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/Goldbach_conjecture.py.

1st solution:

```
from math import sqrt
def is_prime(x):
if x == 1:
return 0
for i in range (2, int(sqrt(x)) + 1):
if x % i == 0:
return 0
return 1
t = int(input())
for i in range(t):
n = int(input())
if n % 2 != 0:
print(f'{n} is odd. Must input an even natural number.')
else:
for j in range(3, n//2):
if is_prime(j) == 1 and is_prime(n - j) == 1:
print(f'{n} = {j} + {n - j}')
# break # uncomment if want only 1 representation,
# comment if want all satisfied representations
```

2nd solution:

```
def prime(x):
d = 0
for i in range(1, (x//2) + 1):
if x % i == 0:
d += 1
return d == 1
n = int(input())
for i in range(n):
k = int(input())
for j in range(k - 2, 1, -1):
if prime(j) == True and prime(k - j) == True:
print(f'{k} = {j} + {k - j}')
break # uncomment if want only 1 representation, comment if want all satisfied representations
```

143 ([Vie21], 6., p. 27, Tây Ninh 2019, Số hoàn hảo – Perfect number)**.** Số hoàn hảo *là 1 số tự nhiên mà tổng tất cả các ước tự nhiên thực sự của nó bằng chính nó. Trong đó ước thực sự của 1 số là các ước dương không bằng số đó. Lập trình nhập vào 1 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 nguyên dương của số đó (i.e., bao gồm tất cả các ước tự nhiên thực sự & chính số đó).*

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/perfect_number.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/perfect_num.py.

1st solution:

```
def is_perfect_number(n):
sum_proper_divisor = 0
list_divisor = []
for i in range(1, round(n/2 + 1)):
```

```
        if n % i == 0:
        list_divisor.append(i)
        sum_proper_divisor += i
        if sum_proper_divisor == n:
        list_divisor.append(n)
        print(f'{n} is a perfect number.')
        print(f'List of of all divisor of {n}:', *list_divisor, '.')
        else:
        print(f'{n} is not a perfect number.')
        n = int(input())
        is_perfect_number(n)
```

2nd solution:

```
        def perfect(x):
        S = 0
        for i in range(1, x):
        if x % i == 0:
        S += i
        return S == x
        n = int(input())
        if perfect(n) == True:
        print(f'{n} is a perfect number')
        for i in range(n):
        if n % (i + 1) == 0:
        print(i + 1, end = ' ')
        else:
        print(f'{n} is not a perfect number')
```

144 ([Vie21], 7., p. 27, Đồng Nai 2020, Số may mắn – Lucky number)**.** *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 1 dãy số phần thưởng. Mỗi bạn sẽ bốc thăm 1 số* k *ứng với con số của món quà mình đạt được.*

- Yêu cầu: *In số của món quà học sinh đạt được.*
- Input. *Dòng duy nhất ghi số nguyên* n *&* k*,* $1 \leq k \leq n \leq 1000$*.*
- Output. *In số của món quà học sinh đạt được:*

lucky_number.inp	lucky_number.out
10 6	2

Python:

```
        from math import floor
        def luck_number(n, k):
        if k <= floor((n + 1)/2):
        print(2*k - 1)
        else:
        print(2*(k - round((n + 1)/2) + 1))

        nk = input()
        nk = nk.split()
        n = int(nk[0])
        k = int(nk[1])
        luck_number(n, k)
        """
        # Print all
        for i in range(1, k + 1):
        luck_number(n, i)
        """
```

145 ([Vie21], 8., p. 27, Ninh Bình 2019, Ước chung lớn nhất ƯCLN – greatest common divisor gcd)**.** *Nhập vào* 3 *số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *của* 3 *số này. E.g., nhập vào* 3 *số:* $4, 6, 12$ *thì kết quả* ƯCLN *là* 2*.*

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/gcd3.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/UCLN.py.

1st solution:

```
def UCLN(a, b):
while a != b:
if a > b:
a -= b
else:
b -= a
return a
a, b, c = map(int,input().split())
if (a <= 0 or b <= 0 or c <= 0):
print('Error: Non-positive.')
else:
a = UCLN(a,b)
a = UCLN(a,c)
print(a)
```

2nd solution:

```
def GCD(a, b):
a, b = abs(a), abs(b)
while a*b != 0:
if a > b:
a = a % b
else:
b = b % a
return max(a,b)
a, b, c = map(int,input().split())
print(GCD(GCD(a, b), c))
```

We can also use `math.gcd()` function to test the result (only for testing purpose).

```
from math import gcd
print(gcd(a, b, c))
```

146 (Bội chung nhỏ nhất BCNN – least common multiplier lcd)**.** *Nhập vào $n \in \mathbb{N}^\star$ số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *&* BCNN *của n số này.*

Python:

```
from math import lcm
def lcm3(a, b, c):
return lcm(a, b, c)
a = int(input())
b = int(input())
c = int(input())
print(lcm3(a, b, c))
```

We are interested in various types of numbers whose beautiful properties are studied in number theory.

5.2.1 Square number

Definition 1 (Square number)**.** *In mathematics, a* square number *or* perfect square *is an integer that is the square of an integer, i.e., it is the product of some integer with itself $n \cdot n = n^2$, for some $n \in \mathbb{Z}$.*

The set of all square numbers is given by $A = \{n^2 | n \in \mathbb{Z}\} = \{n^2 | n \in \mathbb{N}\} = \{0^2, 1^2, 2^2, 3^2, \dots\}$.

"The usual notation for the square of a number $n \in \mathbb{Z}$ is not the product $n \cdot n$, but the equivalent exponentiation n^2, usually pronounced as "n squared". The name *square* number comes from the name of the shape. The unit of area is defined as the area of a unit square 1×1. Hence, a square with side length n has area n^2. If a square number is represented by n points, the points can be arranged in rows as a square each side of which has the same number of points as the square root of n, thus, square numbers are a type of figurate numbers (other examples being cube numbers & triangular numbers).

In the real number system $\mathbb{R}$, square numbers are nonnegative. A nonnegative integer is a square number when its square root is again an integer, e.g., $\sqrt{9} = 3$, so 9 is a square number. A positive integer has no square divisors except 1 is called square-free.

For a nonnegative integer $n \in \mathbb{N} \equiv \mathbb{Z}_{\geq 0}$, the nth square number is n^2, with $0^2 = 0$ being the 0th one. The concept of square can be extended to some other number systems. If rational numbers are included, then a square is the ratio of 2 square integers, &, conversely, the ratio of 2 square integers is a square $\frac{m^2}{n^2} = \left(\frac{m}{n}\right)^2$, $\forall m, n \in \mathbb{Z}$, $n \neq 0$.

Starting with 1, there are $\lfloor \sqrt{m} \rfloor$ square numbers up to & including m, where the expression $\lfloor x \rfloor$ represents the floor of the number $x \in \mathbb{R}$." – Wikipedia/square number

Example 1 (Set of squares: $\{n^2 | n \in \mathbb{N}, n \leq 60\}$)**.** *"The squares (sequence A000290 in the OEIS) smaller than $60^2 = 3600$ are:* $0^2 = 0$, $1^2 = 1$, $2^2 = 4$, $3^2 = 9$, $4^2 = 16$, $5^2 = 25$, $6^2 = 36$, $7^2 = 49$, $8^2 = 64$. $9^2 = 81$, $10^2 = 100$, $11^2 = 121$, $12^2 = 144$, $13^2 = 169$, $14^2 = 196$, $15^2 = 225$, $16^2 = 256$, $17^2 = 289$, $18^2 = 324$, $19^2 = 361$, $20^2 = 400$, $21^2 = 441$, $22^2 = 484$, $23^2 = 529$, $24^2 = 576$, $25^2 = 625$, $26^2 = 676$, $27^2 = 729$, $28^2 = 784$, $29^2 = 841$, $30^2 = 900$, $31^2 = 961$, $32^2 = 1024$, $33^2 = 1089$, $34^2 = 1156$, $35^2 = 1225$, $36^2 = 1296$, $37^2 = 1369$, $38^2 = 1444$, $39^2 = 1521$, $40^2 = 1600$, $41^2 = 1681$, $42^2 = 1764$, $43^2 = 1849$, $44^2 = 1936$, $45^2 = 2025$, $46^2 = 2116$, $47^2 = 2209$, $48^2 = 2304$, $49^2 = 2401$, $50^2 = 2500$, $51^2 = 2601$, $52^2 = 2704$, $53^2 = 2809$, $54^2 = 2916$, $55^2 = 3025$, $56^2 = 3136$, $57^2 = 3249$, $58^2 = 3364$, $59^2 = 3481$.

The difference between any perfect square & its predecessor is given by the identity $n^2 - (n-1)^2 = 2n - 1$, $\forall n \in \mathbb{Z}$. Equivalently, it is possible to count square numbers by adding together the last square, the last square's root, & the current root, i.e., $n^2 = (n-1)^2 + (n-1) + n$, $\forall n \in \mathbb{Z}$." – Wikipedia/square number/examples

"The numebr $m \in \mathbb{N}$ is a square iff one can arrange m points in a square. The expression for the nth square number is n^2. This is also equal to the sum of the 1st n odd numbers, where a square results from the previous one by adding an odd number of points. The formula follows: $n^2 = \sum_{i=1}^{n}(2i-1)$, $\forall n \in \mathbb{N}^\star$.

Example 2 (Square number n^2 = sum of 1st n odds)**.** $1^2 = 1$, $2^2 = 4 = 1+3$, $3^2 = 1+3+5$, $4^2 = 16 = 1+3+5+7$, $5^2 = 25 = 1+3+5+7+9$, $6^2 = 36 = 1+3+5+7+9+11$, $7^2 = 49 = 1+3+5+7+9+11+13$, $8^2 = 64 = 1+3+5+7+9+11+13+15$, $9^2 = 81 = 1+3+5+7+9+11+13+15+17$, $10^2 = 100 = 1+3+5+7+9+11+13+15+17+19$.

There are several recursive methods for computing square numbers. E.g., the nth square number can be computed from the previous square by $n^2 = (n-1)^2 + (n-1) + n = (n-1)^2 + (2n-1)$. Alternatively, the nth square number can be calculated from the previous two by douling the $(n-1)$th square, subtracting the $(n-2)$th square number, & adding 2, because $n^2 = 2(n-1)^2 - (n-2)^2 + 2$, e.g., $2 \cdot 5^2 - 4^2 + 2 = 2 \cdot 25 - 16 + 2 = 50 - 16 + 2 = 36 = 6^2$.

The square minus 1 of a number $m \in \mathbb{R}$ is always the product of $m-1$ & $m+1$, i.e., $m^2 - 1 = (m-1)(m+1)$, $\forall m \in \mathbb{R}$, e.g., since $7^2 = 49$, one has $6 \cdot 8 = 48$. Since a prime number has factors of only 1 & itself, & since $m = 2$ is the only nonzero value of m to give a factor of 1 on the RHS of the equation, it follows that 3 is the only prime number 1 less than a square, i.e., $3 = 2^2 - 1$.

More generally, the difference of the squares of 2 numbers is the product of their sum & their difference, i.e., $a^2 - b^2 = (a+b)(a-b)$. This is the difference-of-squares formula, which can be usful for mental arithmetic, e.g., $47 \cdot 53$ can be easily computed as $50^2 - 3^2 = 2500 - 9 = 2491$. A square number is also the sum of 2 consecutive triangular numbers. The sum of 2 consecutive square numbers is a centered square number. Every odd square is also a centered octagonal number.

Another property of a square number is that (except 0) it has an odd number of positive divisors, while other natural numbers have an even number of positive divisors. An integer root is the only divisor that pairs up with itself to yield the square number, while other divisors come in pairs.

Lagrange's 4-square theorem states that any positive integer can be written as the sum of 4 or fewer perfect squares. 3 squares are not sufficient for numbers of the form $4^k(8m+7)$. A positive integer can be represented as a sum of 2 squares precisely if its prime factorization contains no odd powers of primes of the form $4k+3$. This is generalized by Waring's problem.

The sum of the n 1st square numbers is

$$\boxed{\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}, \ \forall n \in \mathbb{N}^\star.}$$

" – Wikipedia/square number/properties

5.2.2 Square-free integer

Definition 2 (Square-free integer)**.** *In mathematics, a* square-free integer *(or* squarefree integer*) is an integer which is divisible by no square number other than* 1*, i.e., its prime factorization has exactly* 1 *factor for each prime that appears in it.*

Example 3. $10 = 2 \cdot 5$ *is square-free, but* $18 = 2 \cdot 3^2$ *is not, because* $18 \vdots 3^2$.

Example 4. *The smallest positive square-free numbers are* 1, 2, 3, 5, 6, 7, 10, 11, 13, 14, 15, 17, 19, 21, 22, 23, 26, 29, 30, 31, 33, 34, 35, 37, 38, 39, ... *(sequence A005117 in the OEIS).*

"Every positive integer $n \in \mathbb{N}^\star$ can be factored in a unique way as $n = \prod_{i=1}^{k} q_i^k$, where the q_i different from one are square-free integers that are pairwise coprime. This is called the *square-free factorization* of n.

To construct the square-free factorization, let $n = \prod_{j=1}^{h} p_j^{e_j}$ be the prime factorization of n, where the p_j are distinct prime numbers. Then the factors of the square-free factorization are defined as $q_i = \prod_{j:e_j=i} p_j$.

An integer is square-free iff $q_i = 1$ for all $i > 1$. An integer greater than 1 is the kth power of another integer iff k is a divisor of all i such that $q_i \neq 1$.

The use of the square-free factorization of integers is limited by the fact that its computation is as difficult as the computation of the prime factorization. More precisely every known algorithm for computing a square-free factorization computes also the prime factorization. This is a notable difference with the case of polynomials for which the same definitions can be given, but, in this case, the square-free factorization is not only easier to compute than the complete factorization, but it is the 1st step of all standard factorization algorithms." – Wikipedia/square-free integer/square-free factorization

5.2.3 Figurate number

5.2.4 Cube number

5.2.5 Triangular number

5.2.6 Powerful number

Definition 3 (Powerful number)**.** *A* powerful number *is a positive integer $n \in \mathbb{N}^\star$ such that for every prime number p dividing n, p^2 also divides n. Equivalently, a powerful number is the product of a square & a cube, i..e, a number n of the form $n = a^2b^3$, where $a, b \in \mathbb{N}^\star$. Powerful numbers are also known as* squareful, square-full, *or* 2-full.

Example 5. *The following is a list of all powerful numbers between* 1 *&* 1000*:* 1, 4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 72, 81, 100, 108, 121, 125, 128, 144, 169, 196, 200, 216, 225, 243, 256, 288, 289, 324, 343, 361, 392, 400, 432, 441, 484, 500, 512, 529, 576, 625, 648, 675, 676, 729, 784, 800, 841, 864, 900, 961, 968, 972, 1000, ... *(sequence A001694 in the OEIS).*

Definition 4 (k-powerful number)**.** *A* k-powerful number *(or* k-ful number*, or* k-full number*) is an integer all of whose prime factors have exponents a least* k*.*

5.2.7 Highly powerful number

5.2.8 Achilles number

"An Achilles number is a positive integer that is powerful (in the sense that each prime factor occurs with exponent > 1) but imperfect (in the sense that the number is not a perfect power)." – Wolfram MathWorld/Achilles number

Definition 5 (Achilles number)**.** *"An* Achilles number *is a number that is powerful but not a perfect power.*

A positive integer $n \in \mathbb{N}^\star$ is a powerful number if, for every prime factor p of n, p^2 is also a divisor. I.e., every prime factor appears at least squared in the factorization. All Achilles numbers are powerful. However, not all powerful numbers are Achilles numbers: only those that cannot be represented as m^k, where m, k are positive integers greater than 1.

Achilles numbers were named by Henry Bottomley after Achilles, a hero of the Trojan war, who was also powerful but imperfect. *Strong Achilles numbers* are Achilles numbers whose Euler totients are also Achilles numbers." – Wikipedia/Achilles number

"A number $n = \prod_{i=1}^{k} p_i^{a_i} = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$ is powerful if $\min\{a_1, a_2, \ldots, a_k\} \geq 2$. If in addition $\gcd(a_1, a_2, \ldots, a_k) = 1$ the number is an Achilles number.

Example 6. *The Achilles numbers up to 5000 are:* 72, 108, 200, 288, 392, 432, 500, 648, 675, 800, 864, 968, 972, 1125, 1152, 1323, 1352, 1372, 1568, 1800, 1944, 2000, 2312, 2592, 2700, 2888, 3087, 3200, 3267, 3456, 3528, 3872, 3888, 4000, 4232, 4500, 4563, 4608, 5000 *(sequence A052486 in the OEIS).*

The smallest pair of consecutive Achilles numbers is: $5425069447 = 7^3 \cdot 41^2 \cdot 97^2$, $5425069448 = 2^3 \cdot 26041^2$." – Wikipedia/Achilles number/sequence of Achilles numbers

Example 7. *"*108 *is a powerful number. Its prime factorization is* $2^2 \cdot 3^3$*, & thus its prime factors are* 2 *&* 3*. Both* $2^2 = 4$ *&* $3^2 = 9$ *are divisors of* 108*. However,* 108 *cannot be represented as* m^k*, where* m *&* k *are positive integers greater than* 1*, so* 108 *is an Achilles number.*

Example 8. 360 *is not an Achilles number because it is not powerful. 1 of its prime factors is* 5 *but* 360 *is not divisible by* $5^2 = 25$*.*

Example 9. 784 *is not an Achilles number. It is a powerful number, because not only are* 2 *&* 7 *its only prime factors, but also* $2^2 = 4$ *&* $7^2 = 49$ *are divisors of it. Nonetheless, it is a perfect power:* $784 = 2^4 \cdot 7^2 = (2^2)^2 \cdot 7^2 = (2^2 \cdot 7)^2 = 28^2$*. So it is not an Achilles number.*

Example 10. $500 = 2^2 \cdot 5^3$ *is a strong Achilles number as its Euler totient of* $200 = 2^3 \cdot 5^2$ *is also an Achilles number."* – *Wikipedia/Achilles number/examples*

Problem 21 (Project Euler, Problem 302: Strong Achilles Number)**.** *A positive integer* n *is* powerful *if* p^2 *is a divisor of* n *for every prime factor* p *in* n*. A positive integer* n *is a* perfect power *if* n *can be expressed as a power of another positive integer. A positive integer* n *is an* Achilles number *if* n *is powerful but not a perfect power. E.g.,* 864 *&* 1800 *are Achilles numbers:* $864 = 2^5 \cdot 3^3$*, &* $1800 = 2^3 \cdot 3^2 \cdot 5^2$*. We will call a positive integer* S *a* strong Achilles number *if both* S *&* $\varphi(S)$ *are Achilles numbers. E.g.,* 864 *is a strong Achilles number:* $\varphi(864) = 288 = 2^5 \cdot 3^2$*. However,* 1800 *isn't a strong Achilles number because:* $\varphi(1800) = 480 = 2^5 \cdot 3 \cdot 5$*. There are* 7 *strong Achilles numbers below* 10^4 *&* 656 *below* 10^8*. How may strong Achilles numbers are there below* 10^{18}*?*

5.2.9 Perfect power

Definition 6 (Perfect power)**.** *"In mathematics, a* perfect power *is a natural number that is a product of equal natural factors, or, in other words, an integer that can be expressed as a square or a higher integer power of another integer greater than* 1*. More formally,* n *is a* perfect power *if there exist natural numbers* $m > 1$*, &* $k > 1$ *such that* $m^k = n$*. In this case,* n *may be called a* perfect kth power*. If* $k = 2$ *or* $k = 3$*, then* n *is called a perfect square or perfect cube, respectively.*

Sometimes, 0 & 1 are also considered perfect power: $0^k = 0$, $\forall k \in (0, \infty)$, $1^k = 1$, $\forall k \in \mathbb{R}$." – Wikipedia/perfect power

"A sequence of perfect powers can be generated by iterating through the possible values for m & k. The 1st few ascending perfect powers in numerical order (showing duplicate powers) are (sequence A072103 in the OEIS): $2^2 = 4$, $2^3 = 8$, $3^2 = 9$, $2^4 = 16$, $4^2 = 16$, $5^2 = 25$, $3^3 = 27$, $2^5 = 32$, $6^2 = 36$, $7^2 = 49$, $2^6 = 64$, $4^3 = 64$, $8^2 = 64$, ...

The sum of the reciprocals of the perfect powers (including duplicates such as 3^4 & 9^2, both of which equal 81) is 1: $\sum_{m=2}^{\infty} \sum_{k=2}^{\infty} \frac{1}{m^k} = 1$, which can be proved as follows:

$$\sum_{m=2}^{\infty} \sum_{k=2}^{\infty} \frac{1}{m^k} = \sum_{m=2}^{\infty} \frac{1}{m^2} \sum_{k=0}^{\infty} \frac{1}{m^k} = \sum_{m=2}^{\infty} \frac{1}{m^2} \cdot \frac{m}{m-1} = \sum_{m=2}^{\infty} \frac{1}{m(m-1)} = \sum_{m=2}^{\infty} \left(\frac{1}{m-1} - \frac{1}{m} \right) = 1.$$

Example 11. *The 1st perfect powers without duplicates are: (sometimes* 0 *&* 1*),* 4, 8, 9, 16, 25, 27, 32, 36, 49, 64, 81, 100, 121, 125, 128, 144, 169, 196, 216, 225, 243, 256, 289, 324, 343, 361, 400, 441, 484, 512, 529, 576, 625, 676, 729, 784, 841, 900, 961, 1000, 1024, ... *(sequence A001597 in the OEIS).*

6 Character & String – Xâu & Chuỗi

147 ([Vie21], 1., p. 28, Tây Ninh 2019, Số đảo ngược – Reversed number)**.** *Tìm số đảo ngược* y *của 1 số* $x \in \mathbb{Z}$ *biết* y *gồm các chữ số của* x *& viết theo thứ tự ngược lại. Xuất ra kết quả là số* $y \mod 19$. Dữ liệu: $x \in \mathbb{N}^\star$. Kết quả: $y \mod 19$ *với* y *là số đảo ngược của* x.

reversed_number.inp	reversed_number.out	Giải thích
123	17	Đảo ngược của 123 là 321 & $321 \mod 19 = 17$

Pascal:

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/reversed_number.py.

```
x = input()
y = x[::-1]
print(int(y) % 19)
```

C:

C++:

148 ([Vie21], 2., pp. 28–29, Bắc Giang 2020, Nén xâu – String compression)**.** *Viết chương trình nhập vào 1 xâu ký tự chỉ gồm các chữ cái Tiếng Anh, chữ số, dấu cách, & dấu gạch nối.*

- Yêu cầu: *Nén các ký tự liên tiếp giống nhau thành số lượng ký tự & ký tự đó, rồi đưa ra xâu sau khi nén.*
- Input. *Đọc từ file văn bản* string_compression.inp *gồm 1 xâu* S *có số lượng ký tự không quá* 255 *ký tự.*
- Output. *Đưa ra file văn bản* string_compression.out *gồm xâu* S *sau khi nén.*

string_compression.inp	lstring_compression.out
aaaababb cc	4a1b1a2b4 2c

Pascal:

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/string_compression.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/string_compression_io.py.

```
s = input()
i = 0
ans = ""
while i < len(s):
temp = s[i]
count = 0
while temp == s[i]:
i += 1
count += 1
if i == len(s):
break
ans += str(count) + temp
print(ans)
```

C:

C++:

149 ([Vie21], 3., p. 30, Hậu Giang 2020, Tính nhân – Multiplication)**.** *Viết chương trình nhập vào 2 số nguyên dương* $a, b \in \mathbb{N}^\star$. *Sau đó thực hiện nhân* $a \times b$ *như cách nhân bằng tay thông thường ở tiểu học. E.g., nhập vào thừa số thứ 1:* 125*, nhập vào thừa số thứ 2:* 15. Output.

```
  125
  x
  15
  -----
  625
  125
  -----
  1875
```

Pascal:

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/multiplication.py.

```
a = input()
b = input()
prod = int(a) * int(b)
max_len = len(str(prod)) + 1
s = a
while len(s) < max_len:
s = ' ' + s
print(s)
print("x")
s = b
while len(s) < max_len:
s = ' ' + s
print(s)
sep = "-" * max_len # separator
print(sep)
space = 0
for i in range (len(b), 0, -1):
s = int(b[i-1]) * int(a)
if space > 0:
tmp = ' ' * space
s = str(s) + tmp
s = str(s)
while len(s) < max_len:
s = ' ' + s
print(s)
space += 1
if len(b) > 1:
print(sep)
print(' ' + str(prod))
```

Problem 22 ([DV21], pp. 42–43, Anagrams)**.** *A word* `w` *is an* anagram *of a word* `v` *if a permutation of the letters transforming* `w` *into* `v` *exists. Given a set of* n *words of length at most* k*, we would like to detect all possible anagrams.*

- Sample Input. *below the car is a rat drinking cider and bending its elbow while this thing is an arc that can act like a cat which cried during the night caused by pain in its bowel.*
- Sample Output. `{bowel below elbow}, {arc car}, {night thing}, {cried cider}, {act act}`

7 1D Array & List – Mảng 1 Chiều & Danh Sách

8 Matrix – Ma Trận

9 Arrangement – Sắp Xếp

See, e.g., [Knu98, Chap. 5: Sorting], [Vie21, Chap. II, Sect. Dạng bài sắp xếp].

150 ([Vie21], 1., p. 83, Bắc Giang 2019, Dãy số – Sequence)**.** *Sử dụng hàm Randomize để khởi tạo dãy số ngẫu nhiên từ* 0 *đến* 9 *gồm* $n \in \mathbb{N}^\star$ *phần tử,* $0 < n \leq 100$*, kết quả ghi ra tệp* `random.out`*, mỗi phần tử cách nhau* 1 *dấu cách.*

- Yêu cầu: *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.*

- Output. *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* $0, 1, \ldots, 9$. *E.g.:*

`random_sequence.inp`	`random_sequence.out`
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

151 ([Vie21], 2., p. 85, Vị Thanh, Hậu Giang 2019, Dãy số không giảm – Nondecreasing sequence)**.** *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, e.g.,* $a_1 = 234$, $a_2 = 123$, $a_3 = 345$ *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 & xuất ra màn hình, các số cách nhau 1 khoảng trắng. E.g.:*

`nondecreasing_sequence.inp`	`nondecreasing_sequence.out`
$a_1 = 234$ $a_2 = 123$ $a_3 = 345$	123234 123345 234123 234345 345123 345234

152 ([Vie21], 1., p. 87, Sorting ascending)**.** *Cho vào* m *dãy số nguyên* $(a_{i,j})_{j=1}^{n_i}$, $n_i \in \mathbb{N}^\star$, $n_i \leq 100$, $\forall i = 1, 2, \ldots, m$, $|a_{ij}| < 32000$, $\forall i = 1, 2, \ldots, m$, $\forall j = 1, 2, \ldots, \max\{n_i | i = 1, 2, \ldots, m\}$.

- Yêu cầu: *Sắp xếp từng dãy số trên theo thứ tự tăng dần.*
- Input. *Trong* m *dòng, mỗi dòng là dãy số, bắt đầu là* 1 *số nguyên* n *là số lượng các phần tử của dãy số,* $1 \leq n \leq 100$, n *số nguyên tiếp theo là giá trị các phần tử của dãy.*
- Output. *Ghi ra* m *dòng là* m *dãy số đã được sắp xếp theo thứ tự tăng dần. E.g.:*

`sorting_ascending.inp`	`sorting_ascending.out`
2 2 1 3 4 3 1 4 1 4 5 2	1 2 1 3 4 1 2 4 5

10 Algorithm – Thuật Toán

10.1 Recursion algorithm – Thuật toán đệ quy

Đệ quy (recursion) là phương pháp dùng trong các chương trình máy tính trong đó có 1 hàm tự gọi chính nó.

153 ([Vie21], p. 91, Giai thừa – Factorial $n!$)**.** *Tính* $n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdots (n-1)n$.

- Input. *Dòng đầu là số lượng test. Mỗi dòng tiếp theo gồm* 1 *số* $n \in \mathbb{N}$.
- Output. *Với mỗi test, in ra* $n!$ *theo mẫu:*

`factorial.inp`	`factorial.out`
2 3 4	3! = 6 4! = 24

154 ([Vie21], 1., p. 92, Hàm $f91$ của McCarthy – McCarthy's $f91$ function)**.** *McCarthy là* 1 *nhà khoa học máy tính nổi tiếng, ông đã định nghĩa hàm đệ quy* $f91 : \mathbb{Z} \to \mathbb{Z}$ *như sau:*

$$f_{91}(n) = \begin{cases} f_{91}(f_{91}(n+11)), & \text{if } n \leq 100, \\ n - 1, & \text{if } n \geq 101. \end{cases}$$

Viết 1 *chương trình tính toán hàm McCarthy's f91.*

- Input. *File input chứa 1 dãy các số nguyên dương, mỗi số không quá* 1000. *Mỗi số trên 1 dòng.*
- Output. *Hiện ra theo định dạng trong ví dụ sau:*

McCarthy_f91_function.inp	McCarthy_f91_function.out
500 91	f91(500) = 490 f91(91) = 91

See, e.g., Wikipedia/McCarthy 91 function.

155 ([Vie21], 2., p. 93, Chỉnh hợp – Arrangement A_n^k)**.** *Tìm tất cả các chỉnh hợp chập k của n phần tử từ* 1 *đến n,* $0 < k \leq n < 10$.

- Input. *File input chứa 1 dòng gồm* $n, k \in \mathbb{N}$.
- Output. *In ra số chỉnh hợp tìm được & liệt kê các chỉnh hợp, giữa các test là 1 dòng trắng, e.g.,*

arrangement.inp	arrangement.out	arrangement.inp	arrangement.out
1 3	3 11 13 33	2 3	6 1 2 1 3 2 1 2 3 3 1 3 2

156 ([Vie21], 3., p. 93, Hoán vị – Permutation $P_n = n!$)**.** *Viết chương trình in ra tất cả các hoán vị của* $n \in \mathbb{N}^\star$ *được nhập từ bàn phím, e.g.,*

permutation.inp	permutation.out
2	1 2 2 1
3	1 2 3 1 3 2 2 1 3 2 3 1 3 2 1 3 1 2

157 ([Vie21], 5., p. 94, Tổ hợp – Combinatoric C_n^k)**.** *Tìm tất cả các tổ hợp chập k của n phần tử từ* 1 *đến n,* $0 < k \leq n < 10$.

- Input. *File input có dòng đầu gồm 1 số tự nhiên* `ntest` *là số lượng test của file* `combinatoric.inp`. *Mỗi test 1 dòng gồm* 2 *số* $n, k \in \mathbb{Z}$.
- Output. *Với mỗi test, in ra số tổ hợp tìm được & liệt kê các tổ hợp, giữa các test là 1 dòng trắng. E.g.:*

combinatoric.inp	combinatoric.out
2 1 3 2 3	3 1 2 3 3 1 2 1 3 2 3

158 ([Vie21], 4., pp. 93–94, Trò chơi số học – Number theory game)**.** *1 trò chơi phổ biến của trẻ em với bảng* $n \times n$ *ô* $2 \leq n \leq 5$. *Trong mỗi ô chứa 1 số có* 1 *chữ số từ* 1 *tới* 9. *Với 1 số số cho trước, điền các số còn lại vào ô sao cho tổng các hàng ngang bằng nhau & bằng tổng các hàng dọc.*

- Input. `number_theory_game.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* $n \times n$ *trong đó số* 0 *là ô trắng cần điền.*

- Output. `number_theory_game.out`: *Với mỗi test, nếu có thể tìm ra đáp án thỏa mãn quy luật của bảng thì in ra ma trận $n \times n$ 1 cách điền bất kỳ. Nếu không có kết quả nào in ra 1 chuỗi:* `"cannot find."`. *Giữa các test cách nhau 1 dòng trắng. E.g.:*

number_theory_game.inp	number_theory_game.out
1 3 1 0 5 0 3 0 3 0 4	1 4 5 6 3 1 3 3 4

10.2 Search algorithm – Thuật toán tìm kiếm

Tìm kiếm & sắp xếp là 2 hoạt động hằng ngày ta thường sử dụng trong các ứng dụng tin học.

See, e.g., [Knu98, Chap. 6: Searching].

11 Problem in Elementary Physics – Bài Toán Tin Học Trong Vật Lý Sơ Cấp

12 Problem in Elementary Chemistry – Bài Toán Tin Học Trong Hóa Học Sơ Cấp

13 Tuyển Sinh THPT Chuyên Tin

13.1 Tuyển Sinh THPT Chuyên Bến Tre 2023–2024

13.2 Tuyển Sinh THPT Chuyên Tiền Giang 2023–2024

159 (TS THPT Chuyên Tiền Giang 2023–2024, Fences – Hàng rào)**.** *Bác nông dân John có 1 khu đất trồng rau hình chữ nhật kích thước $m \times n$, gồm m hàng, mỗi hàng gồm n ô vuông độ dài cạnh là 1. Bác John cần làm hàng rào để ngăn cách từng ô vuông riêng biệt. Đường biên xung quanh khu đất cũng cần được rào lại. Giúp bác John tính tổng độ dài cần rào.*

- Input. *Dọc từ file* `fences.inp` *gồm 1 dòng chứa 2 số nguyên dương lần lượt là m, n, giữa m, n được cách nhau bởi 1 dấu cách.*
- Output. *Ghi ra file* `fences.out` *1 số nguyên dương là tổng độ dài cần rào.*
- Constraints. $0 < m, n \leq 10^9$.
- Sample.

fences.inp	fences.out
2 3	17
5 3	38

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/fences.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/fences.py.

1st solution:

```
m, n = map(int, input().split())
print((n + 1)*m + (m + 1)*n)
```

2nd solution:

```
t = list(map(int, input().split()))[:2]
m = t[0]
n = t[1]
S = m*(n + 1) + n*(m + 1)
print(S)
```

160 (TS THPT Chuyên Tiền Giang 2023–2024, Odd even – Lẻ chẵn)**.** *Bé Bo đang học về tính chẵn lẻ. Hôm nay, cô giáo dạy Toán cho Bo bài toán như sau: Cho 2 số nguyên dương x, y. Dãy số A được xây dựng theo quy tắc: $A_1 = x$, $A_2 = y$,*

$$A_i = \begin{cases} (A_{i-1} + A_{i-2})\%k, & \text{nếu } i \geq 3, i \text{ là chỉ số chẵn}, \\ |A_{i-1} - A_{i-2}|\%k, & \text{nếu } i \geq 3, i \text{ là chỉ số lẻ}, \end{cases}$$

Trong đó $k := 10^9 + 7$ & % là phép chia lấy phần dư, $|A_{i-1} - A_{i-2}|$ là giá trị tuyệt đối của $A_{i-1} - A_{i-2}$. E.g., với $x = 5$, $y = 7$ thì 1 vài phần tử đầu tiên của dãy số A là: 5 7 2 9 7 16 9 25 16 41 ... *Cho trước số nguyên dương $n \in \mathbb{N}^\star$, tính A_n.*

- Input. *Đọc từ file* `oddeven.inp` *gồm 1 dòng chứa 1 số nguyên dương lần lượt là* x, y, n *giữa các số cách nhau bởi 1 dấu cách.*
- Output. *Ghi ra file* `oddeven.out` *1 số nguyên là giá trị của* A_n.
- Constraint. $1 \le x, y \le 10^9$, $3 \le n \le 10^6$.
- Sample.

oddeven.inp	oddeven.out	Giải thích
5 7 8	25	$x = 5$, $y = 7$, $n = 8$ thì $A_8 = 25$
10 10 3	0	$x = 10$, $y = 10$, $n = 3$ thì $A_3 = 0$

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/oddeven.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/circle.py.

1st solution:

```
k = 10**9 + 7
x, y, n = map(int, input().split())
mem = [x, y]
for i in range(3, n + 1):
if i % 2 == 0:
mem[1] = (mem[0] + mem[1]) % k
else:
mem[0] = abs(mem[0] - mem[1]) % k
print(mem[(n + 1) % 2])
```

2nd solution:

```
A = list(map(int, input().split()))
n = A[2]
A.pop(2)
k = 10**9 + 7
for i in range(3, n + 1):
if i % 2 == 0:
A += [(A[i-2] + A[i-3]) % k]
else:
A += [abs(A[i-2] - A[i-3]) % k]
print(A[n-1])
```

161 (TS THPT Chuyên Tiền Giang 2023–2024, Circle – Vòng tròn)**.** *Bo đang chơi 1 game. Ban đầu, Bo có 1 vòng tròn bán kính* k. *Trên màn hình game còn có* n *vòng tròn khác, vòng thứ* i *có bán kính là* A_i. *Bo có thể điều khiển vòng tròn của mình để chạm vào các vòng tròn khác. Mỗi khi vòng tròn của Bo chạm vào 1 vòng tròn khác thì 1 trong 2 khả năng có thể xảy ra:*

- *Nếu bán kính vòng tròn do Bo điều khiển lớn hơn hay bằng bán kính vòng tròn bị chạm vào thì vòng tròn bị chạm sẽ tan vỡ, còn vòng tròn của Bo trở nên lớn hơn & bán kính của nó sẽ bằng tổng bán kính vòng tròn hiện tại của Bo với bán kính vòng tròn bị chạm đó.*
- *Nếu bán kính vòng tròn do Bo điều khiển nhỏ hơn bán kính vòng tròn bị chạm vào thì kết thúc game.*

Bo chơi game rất giỏi. Bo luôn tìm cách chơi sao cho khi kết thúc game thì bán kính vòng tròn của Bo là lớn nhất. Tính bán kính lớn nhất đó.

- Input. *Đọc từ file* `circle.inp` *gồm 2 dòng:*
  - *Dòng thứ nhất gồm 2 số nguyên dương lần lượt là* n *&* k. *Trong đó,* n *là số lượng vòng tròn,* k *là bán kính vòng tròn của Bo, giữa* n *&* k *được cách nhau bởi 1 dấu cách.*
  - *Dòng thứ 2 gồm* n *số nguyên dương* A_i *lần lượt là bán kính của* n *vòng tròn, giữa các số được cách nhau bởi 1 dấu cách.*
- Output. *Ghi ra file* `circle.out` *1 số nguyên dương là bán kính lớn nhất cần tìm.*
- Constraints. $1 \le n \le 10^3$, $1 \le k \le 10^6$, $1 \le A_i \le 10^6$, $\forall i = 1, 2, \ldots, n$.
- Sample.

circle.inp	circle.out
5 4	16
7 68 3 2 20	

- Explanation. *Ban đầu, có* 5 *vòng tròn* ($n = 5$) *với bán kính lần lượt là* $7, 68, 3, 2, 20$ *&* *vòng tròn của Bo có bán kính là* $k = 4$. *Lần 1: Vòng tròn của Bo chạm vòng tròn có bán kính là* $2 \to$ *vòng tròn của Bo có bán kính mới là* $4 + 2 = 6$. *Lần 2: Vòn tròn của Bo chạm vòng tròn có bán kính là* $3 \to$ *vòng tròn của Bo có bán kính mới là* $6 + 3 = 9$. *Lần 3: Vòng tròn của Bo chạm vòng tròn có bán kính là* $7 \to$ *vòng tròn của Bo có bán kính mới là* $9 + 7 = 16$. *Đến đây, còn 2 vòng tròn có bán kính là* 20 *&* 68. *2 bán kính này đều lớn hơn bán kính vòng tròn của Bo nên kết thúc game. Vậy bán kính lớn nhất cần tìm là* 16.

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/circle.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/circle.py.

1st solution:

```
n, k = map(int, input().split())
A = list(map(int, input().split()))
A.sort()
for i in range(n):
if k >= A[i]:
k += A[i]
else:
break
print(k)
```

2nd solution:

```
T = list(map(int, input().split()))[:2]
n, k = T[0], T[1]
A = list(map(int, input().split()))[:n]
A.sort()
i = 0
while k >= A[i]:
k += A[i]
i += 1
print(k)
```

162 (TS THPT Chuyên Tiền Giang 2023–2024, Wifi password – Mật khẩu wifi). *Hôm nay, thầy giáo Tin học của lớp Bo thấy các bạn học chăm chỉ nên sau giờ thực hành, thầy cho các em truy cập wifi để giải trí. Để vào wifi, các em cần có 1 mật khẩu. Thầy cho 1 xâu* `s` *gồm các ký tự in thường trong bảng chữ cái tiếng Anh & ký tự* `*`*. Xâu mật khẩu có được bằng cách: đi theo thứ tự từ trái qua phải của xâu* `s`*, các em thay mỗi ký tự* `*` *trong xâu* `s` *bằng 1 ký tự in thường (ưu tiên chọn ký tự đứng trước trong bảng chữ cái để thay), nhưng đảm bảo số lần xuất hiện của mỗi ký tự* `c` *trong xâu* `s` *đúng bằng f_c (với* `c` *là 1 ký tự trong bảng chữ cái từ 'a' đến 'z'). Tìm xâu mật khẩu đó.*

- Input. *Đọc từ file* `pass.inp` *gồm 2 dòng:*
  - *Dòng thứ nhất chứa xâu* `s` *chỉ gồm các ký tự in thường & ký tự* `*`*.*
  - *Dòng thứ 2 gồm 26 số nguyên không âm $f_a, f_b, \ldots, f_z$ tương ứng với số lần xuất hiện của các ký tự từ 'a' đến 'z' trong xâu* `s`*, giữa các số được cách nhau bởi 1 dấu cách. Dữ liệu vào đảm bảo xâu* `s` *khác rỗng, luôn có ký tự* `*` *trong xâu* `s` *& tổng $\sum_{c='a'}^{'z'} = f_a + f_b + \cdots + f_z$ đúng bằng số lượng ký tự* `*` *có trong xâu* `s`*.*
- Output. *Ghi ra file* `pass.out` *xâu mật khẩu tìm được.*
- Constraint. *Độ dài xâu* `s` *không quá 10^5, $0 \le f_c \le 10^5$, 'a'≤c≤'z'.*
- Sample.

pass.inp	pass.out
bc*m**a 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	bcamdda
y*e**z* 0 0 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	ydedfzz

- Explanation. *Ở bộ test đầu: Có 3 dấu* `*` *trong xâu* `s`*. Dấu* `*` *thứ nhất được thay bởi ký tự* `a` *(vì $f_a = 1$). Dấu* `*` *thứ 2 & thứ 3 được thay bởi ký tự* `d` *(vì $f_d = 2$).*

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/pass.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/pass.py.

1st solution:

```
s = input()
f = list(map(int, input().split()))
s = list(s)
j = 0
for i in range(len(s)):
if s[i] == '*':
while f[j] == 0:
j += 1
s[i] = chr(j + 97)
f[j] -= 1
s = "".join(s)
print(s)
```

2nd solution:

```
st = input()
F = list(map(int, input().split()))
password = ''
i, j = 0, 0
while i != len(st):
if st[i] != '*':
password += st[i]
else:
while F[j] == 0:
j +=1
password += chr(97 + j)
F[j] -=1
i += 1
print(password)
```

163 (TS THPT Chuyên Tiền Giang 2023–2024, Shape – Cắt hình). *Bé Bo có 1 mảnh giấy hình chữ nhật gồm $m \times n$ ô vuông bằng nhau, với m là chiều dài & n là chiều rộng mảnh giấy. Bo tìm cách cắt từ mảnh giấy này để mỗi lần cắt được hình vuông có diện tích lớn nhất. Thao tác này được thực hiện nhiều lần như thế đối với phần giấy thừa còn lại cho đến khi hết giấy thừa. Đếm số hình vuông mà Bo có thể cắt được.*

- Input. *Đọc từ file* `shape.inp` *gồm 1 dòng chứa 2 số nguyên dương lần lượt là* m *&* n*, giữa* m *&* n *được cách nhau bởi 1 dấu cách.*
- Output. *Ghi ra file* `shape.out` *1 số nguyên dương là số hình vuông Bo cắt được.*
- Constraint. $1 \leq m, n \leq 10^9$, $m \geq n$.
- Sample.

shape.inp	shape.out
8 3	5
21 4	9

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/shape.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/HTP/SHAPE.py.

1st solution:

```
m, n = map(int, input().split())
ans = 0
while m*n != 0:
ans += m // n
m, n = n, m % n
print(ans)
```

2nd solution:

```
m, n = map(int, input().split())
count = 0
while m != 0 and n != 0:
if m > n:
count += m // n
m %= n
else:
count += n // m
n %= m
print(count)
```

13.3 Tuyển Sinh THPT Chuyên Tp. Hồ Chí Minh 2023–2024

164 (TS THPT Chuyên Tp. Hồ Chí Minh 2023–2024, Exponentiation – Lũy thừa)**.** *An vừa được học phép tính lũy thừa & biết được* $a^n = \underbrace{a \cdot a \cdots a}_{n}$*. Cô giáo đã giao cho An 1 số bài tập trên lớp học trực tuyến để luyện tập tính lũy thừa. Các bài tập có dạng tính giá trị* $y = \sum_{i=1}^{n} a_i^{x_i} = a_1^{x_1} + a_2^{x_2} + \cdots + a_n^{x_n}$ *với* $a_1, a_2, \ldots, a_n$ *là các số nguyên dương &* $x_1, x_2, \ldots, x_n$ *là các số nguyên không âm có 1 chữ số (i.e.,* $a_i \in \mathbb{N}^\star$*,* $x_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$*,* $\forall i = 1, 2, \ldots, n$*). An đã thực hiện xong các bài tập & muốn kiểm tra lại đáp án bằng 1 chương trình tính toán. Tuy nhiên khi An nhập dữ liệu cho chương trình thì không nhập được số mũ có định dạng chỉ số trên (superscript) nên chỉ có thể nhập* $y = \sum_{i=1}^{n} p_i = p_1 + p_2 + \cdots + p_n$*, trong đó* p_i *có dạng* $a_i x_i$*,* $\forall i = 1, 2, \ldots, n$*. Ví dụ bài tập khi xem trên lớp học trực tuyến thì biểu thức có dạng* $y = 2^5 + 3^5 + 10^3 + 215^2$ *nhưng khi nhập vào chương trình thì có dạng* $y = 25 + 35 + 103 + 2152$*.*

- Requirement. *Viết chương trình tính giá trị biểu thức là tổng các lũy thừa nhưng biểu thức được nhập như mô tả ở trên.*
- Input. *Vào từ file văn bản* `exponentiation.inp`*, dòng đầu là 1 số nguyên* n *cho biết số lượng số hạng của biểu thức cần tính. Dòng thứ* i *trong* n *dòng tiếp theo cho biết số nguyên* p_i*.*
- Output. *Ghi ra file văn bản* `exponentiation.out` *cho biết giá trị của biểu thức cần tính. Có thể giả sử giá trị các biểu thức luôn nhỏ hơn* 10^9*.*
- Constraint.
  - 40% *test ứng với* 40% *số điểm của bài có* $1 \leq n \leq 3$ *&* $10 \leq p_i < 100$*.*
  - 60% *test ứng với* 60% *số điểm của bài có* $1 \leq n \leq 20$ *&* $10 \leq p_i < 10000$*.*
- Sample.

exponentiation.inp	exponentiation.out
4 25 35 103 2152	47500

Python:

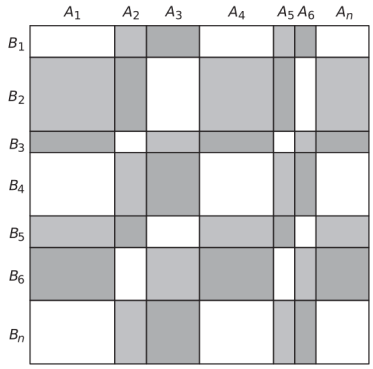
14 Olympic 30.4

165 ([BTC10], 1., p. 5, Connect)**.** *Cho n số nguyên dương $a_1, a_2, \ldots, a_n$, $n \in \mathbb{N}$, $1 < n \leq 100$, $0 < a_i \leq 10^9$, $\forall i = 1, 2, \ldots, n$. Từ các số nguyên này người ta tạo ra 1 số nguyên mới bằng cách kết nối tất cả các số đã cho viết liên tiếp nhau. E.g., với $n = 4$ & các số $12, 34, 567, 890$ ta có thể tạo ra các số mới như sau:* 1234567890, 3456789012, 8905673412, ... *Dễ thấy có $4! = 24$ cách tạo mới như vậy. Trong trường hợp này, số lớn nhất có thể tạo thành là* 8905673412.

- Yêu cầu: *Cho n & các số $a_1, a_2, \ldots, a_n$. Xác định số lớn nhất có thể kết nối được theo quy tắc trên.*
- Input. *Cho trong file văn bản* `connect.inp` *gồm $n + 1$ dòng. Dòng đầu tiên ghi số nguyên n. Trong các dòng còn lại, dòng thứ $i + 1$ ghi số a_i.*
- Dữ liệu ra: *Ghi vào file văn bản* `connect.out` *số lớn nhất được kết nối thành từ các số ban đầu. E.g.:*

connect.inp	connect.out
4	8905673412
12	
34	
567	
890	

Problem 23 (Frosting on the Cake)**.** *Iskander the Baker is decorating a huge cake by covering the rectangular surface of the cake with frosting. For this purpose, he mixes frosting sugar with lemon juice & beetle juice, in order to produce 3 kinds of frosting: yellow, pink, & white. These colors are identified by the number* 0 *for yellow,* 1 *for pink, & 2 for white. To obtain a nice pattern, he partitions the cake surface into vertical stripes of width $A_1, A_2, \ldots, A_n$ centimeters, & horizontal stripes of height $B_1, B_2, \ldots, B_n$ centimeters, for some positive integer $n \in \mathbb{N}^\star$. These stripes split the cake surface into $n \times n$ rectangles. The intersection of vertical stripe i & horizontal stripe j has color number $(i + j) \mod 3$ for all $1 \leq i, j \leq n$, e.g.:*



Hình 1: An instance of the problem *Frosting on the Cake.*

To prepare the frosting, Iskander wants to know the total surface in square centimeters to be colored for each of the 3 colors, & asks for your help.

- Input. *The input consists of the following integers: on the 1st line: the integer $n \in \mathbb{N}^\star$, on the 2nd line: the values of $A_1, A_2, \ldots, A_n, n$ integers separated by single spaces, on the 3rd line: the values of $B_1, B_2, \ldots, B_n, n$ integers separated by single spaces.*
- Limits. *The input satisfies $3 \leq n \leq 100000$ & $1 \leq A_i, B_i \leq 10000$, $i = 1, 2, \ldots, n$.*
- Output. *The output should consist of 3 integers separated by single spaces, representing the total area for each color $0, 1, 2$.*

Solution. Followed [DV21, Sect. 1.8, pp. 39–41], we want to find a solution that runs in time $O(n)$ or possibly $O(n \log n)$, which rules out the naive solution which loops over all n^2 grid cells & accumulates their areas in variables corresponding to each color. Permuting columns or rows preserves the total area of each color. Hence, we can reduce to the $n = 3$ case, by simply summing the values of each color class $A_{3k}, A_{3k+1}, A_{3k+2}$. Then the answer per color class is just the sum of the areas of 3 rectangles. The tricky part relies in not mixing up the colors. □

```
def read_ints(): return [int(x) for x in input().split()]
def cat(l): return tuple(sum(l[n::3]) for n in [1, 2, 0])
input() # n
A = cat(read_ints())
B = cat(read_ints())
print("{} {} {}".format(B[2] * A[0] + B[0] * A[2] + B[1] * A[1],
B[2] * A[1] + B[0] * A[0] + B[1] * A[2],
B[2] * A[2] + B[0] * A[1] + B[1] * A[0]))
```

166 ([Tru23], 1., p. 13, HSG Lớp 10 Vĩnh Phúc 2020–2021, Square – Hình vuông)**.** *Cho n điểm có tọa độ là các số nguyên trên hệ trục tọa độ Oxy. Tìm diện tích hình vuông nhỏ nhất có các cạnh song song với các trục tọa độ sao cho tất cả các điểm đã cho đều thuộc hình vuông đó (điểm nằm trên cạnh hình vuông cũng được coi là thuộc hình vuông đó).*

- Input. *Dòng 1: chứa số nguyên dương $n \in \mathbb{N}^\star$, $2 \leq n \leq 20$, là số lượng điểm có tọa độ là các số nguyên. n dòng tiếp theo, mỗi dòng ghi 2 số nguyên $x, y \in \mathbb{Z}$, $1 \leq x, y \leq 100$, là tọa độ của mỗi điểm.*
- Output. *Ghi diện tích hình vuông nhỏ nhất tìm được.*
- Sample.

square.inp	square.out
3 3 4 5 7 4 3	16

Pascal:

Python: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/square.py.

```
# 1st solution. Author: NQBH
def square():
n = int(input())
a = [0]*n
b = [0]*n
for i in range(n):
tmp = input()
tmp = tmp.split()
a[i] = int(tmp[0])
b[i] = int(tmp[1])
max_diff = 0
for i in range(n):
for j in range(i + 1, n):
if abs(a[i] - a[j]) > max_diff:
max_diff = abs(a[i] - a[j])
if abs(b[i] - b[j]) > max_diff:
max_diff = abs(b[i] - b[j])
print(max_diff**2)
square()
# 2nd solution. Author: Vuong Thanh Trung
inf = 1000
xmin = inf
xmax = -inf
ymin = inf
ymax = -inf
file = open("square.inp", "r")
n = int(file.readline())
for i in range(n):
x, y = map(int, file.readline().split())
xmin = min(xmin, x)
xmax = max(xmax, x)
ymin = min(ymin, y)
ymax = max(ymax, y)
file.close()
file = open("square.out", "w")
c = max(xmax - xmin, ymax - ymin)
file.write(str(c*c))
file.close()
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/square.cpp.

```
#include <bits/stdc++.h>
#define inf 1000
using namespace std;
int xmin = inf, ymin = inf, xmax = -inf, ymax = -inf;
```

```
int n;
int main() {
        freopen("square.inp", "r", stdin);
        freopen("square.out", "w", stdout);
        cin >> n;
        for (int i = 1; i <= n; ++i) {
                int x, y;
                cin >> x >> y;
                xmin = min(xmin, x);
                xmax = max(xmax, x);
                ymin = min(ymin, y);
                ymax = max(ymax, y);
        }
        int c = max(xmax - xmin, ymax - ymin);
        cout << c*c;
        return 0;
}
// Terminal: g++ -std=c++11 -O2 -Wall square.cpp -o square
```

Lưu ý 2. *Nếu yêu cầu của bài toán trên đổi từ 'hình vuông' sang 'hình chữ nhật' thì:*

```
cout << (xmax - xmin)*(ymax - ymin);
```

Bài toán này có thể mở rộng từ 2D sang 3D:

167 (Mở rộng [Tru23], 1., p. 13, HSG Lớp 10 Vĩnh Phúc 2020–2021, Square – Hình vuông)**.** *Cho n điểm có tọa độ là các số nguyên trên hệ trục tọa độ $Oxyz$. Tìm diện tích hình chữ nhật & lập phương nhỏ nhất có các cạnh song song với các trục tọa độ sao cho tất cả các điểm đã cho đều thuộc hình vuông đó (điểm nằm trên cạnh hình chữ nhật hoặc hình vuông cũng được coi là thuộc hình vuông đó).*

- Input. *Dòng 1: chứa số nguyên dương $n \in \mathbb{N}^\star$, $2 \leq n \leq 20$, là số lượng điểm có tọa độ là các số nguyên. n dòng tiếp theo, mỗi dòng ghi 3 số nguyên $x, y, z \in \mathbb{Z}$, $1 \leq x, y, z \leq 100$, là tọa độ của mỗi điểm.*
- Output. *Ghi diện tích hình vuông nhỏ nhất tìm được.*
- Sample.

square.inp	square.out
3 3 4 1 5 7 -8 4 3 9	

168 ([Tru23], 2., pp. 13–14, HSG Lớp 10 Vĩnh Phúc 2020–2021, Divisible by 3 – Chia hết cho 3)**.** *Cho dãy a gồm n số nguyên dương. Cho biết có bao nhiêu cặp số trong dãy có tổng chia hết cho 3, i.e., đếm xem có bao nhiêu cặp chỉ số i, j, $1 \leq i < j \leq n$, sao cho $a_i + a_j \vdots 3$.*

- Input. *Dòng 1: 1 số nguyên duy nhất n, $1 \leq n \leq 10^5$. Dòng 2: Ghi n số nguyên dương $a_1, a_2, \ldots, a_n$, $1 \leq a_i \leq 10^5$, $\forall i = 1, 2, \ldots, n$, là các phần tử của dãy.*
- Output. *1 dòng duy nhất ghi số lượng cặp số của dãy a có tổng chia hết cho 3.*
- Sample.

div3.inp	div3.out	Giải thích
5 3 6 9 12	3	3 cặp số tìm được có chỉ số: $(1,4), (2,3), (3,5)$.
4 3 6 9 12	6	6 cặp số tìm được có chỉ số: $(1,2), (1,3), (1,4), (2,3), (2,4), (3,4)$.

Python:

- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/div3.py.
- https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Python/div3_io.py.

```
file = open("div3.inp", "r")
n = int(file.readline())
a = list(map(int, file.readline().split()))
c = [0, 0, 0]
for x in a:
c[x % 3] += 1
file.close()
res = (c[0]*(c[0] - 1)) // 2 + c[1]*c[2]
file = open("div3.out", "w")
file.write(str(res))
file.close()
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/div3.cpp.

```
#include <bits/stdc++.h>
using namespace std;
int n, x, c[3];
long long res;
int main() {
	freopen("div3.inp", "r", stdin);
	freopen("div3.out", "w", stdout);
	cin >> n;
	for (int i = 1; i <= n; ++i) {
		cin >> x;
		++c[x % 3];
	}
	res = (c[0]*(c[0] - 1))/2 + c[1]*c[2];
	cout << res;
	return 0;
}
// Terminal: g++ -std=c++11 -O2 -Wall div3.cpp -o div3
```

14.1 Google Kickstart Round A 2020

Watch YouTube/William Lin/Winning Google Kickstart Round A 2020.

Problem 24 (Google Kickstart Round A 2020, Allocation)**.** *There are n houses for sale. The ith house costs a_i dollars to buy. You have a budget of b dollars to spend. What is the maximum number of houses you can buy?*

- Input. *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a single line containing the 2 integers n, b. The 2nd line contains n integers. The ith integer is a_i, the cost of the ith house.*
- Output. *For each test case, output 1 line containing* `Case #x: y`*, where* `x` *is the test case number (starting from 1) &* `y` *is the maximum number of houses you can buy.*
- Limits: *Time limit:* 15 s/test set. *Memory limit:* 1GB. $1 \le t \le 100$, $1 \le b \le 10^5$, $1 \le a_i \le 1000$, $\forall i = 1, 2, \ldots, n$. *Test set 1:* $1 \le n \le 100$. *Test set 2:* $1 \le n \le 10^5$.
- Sample.

allocation.inp	allocation.out
3 4 100 20 90 40 90 4 50 30 30 10 10 3 300 999 999 999	Case #1: 2 Case #2: 3 Case #3: 0

```
#include <bits/stdc++.h>
using namespace std;
int n, b, a[100000];

void solve() {
	cout << "Enter n: ";
	cin >>  n;
```

```
                cout << "Enter b: ";
                cin >>  b;
                cout << "Enter array a: ";
                for(int i = 0; i < n; ++i)
                cin >> a[i];
                sort(a, a + n);
                int ans = 0;
                for (int i = 0; i < n; ++i) {
                        if (b >= a[i]) {
                                b -= a[i];
                                ++ ans;
                        }
                }
                cout << ans << "\n";
        }

        int main() {
                int t, i = 1;
                cout << "Enter t: ";
                cin >> t;
                while (t--) {
                        cout << "Case #" << i << ": ";
                        solve();
                        ++i;
                }
        }
```

Problem 25 (Google Kickstart Round A 2020, Plates)**.** *Dr. Patel has n stacks of plates. Each stack contains k plates. Each plate has a positive beauty value, describing how beautiful it looks. Dr. Patel would like to take exactly p plates to use for dinner tonight. If he would like to take a plate in a stack, he must also take all of the plates above it in that stack as well. Help Dr. Patel pick the p plates that would maximize the total sum of beauty values.*

- Input. *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a line containing the 3 integers n, k, p. Then, n lines follow. The ith line contains k integers, describing the beauty values of each stack of plates from top to bottom.*
- Output. *For each test case, output 1 line containing* `Case #x: y`*, where* `x` *is the test case number (starting from* 1*) &* `y` *is the maximum total sum of beauty values that Dr. Patel could pick.*
- Limits: *Time limit:* 20 s/test set*. Memory limit:* 1GB*.* $\leq t \leq 100$*,* $1 \leq k \leq 30$*,* $1 \leq p \leq nk$*. The beauty values are between* 1 *&* 100*, inclusive. Test set 1:* $1 \leq n \leq 3$*. Test set 2:* $1 \leq n \leq 50$*.*
- Sample.

plate.inp	plate.out
2	Case #1: 250
2 4 5	Case #2: 180
10 10 100 30	
80 50 10 50	
3 2 3	
80 80	
15 50	
20 10	

15 CSES Problem Set

Problem 26 (Weird Algorithm)**.** *Consider an algorithm that takes as input a positive integer $n \in \mathbb{N}^\star$. If n is even, the algorithm divides it by* 2*, & if n is odd, the algorithm multiplies it by* 3 *& adds* 1*. The algorithms repeats this, until n is* 1*. E.g., the sequence for $n = 3$ is as follows:* $3 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$*. Your task is to simulate the execution of the algorithm for a given value of n.*

- Input. *The only input line contains an integer $n \in \mathbb{Z}$.*
- Output. *Print a line that contains all values of n during the algorithm.*
- Constraints. $1 \leq n \leq 10^6$.

- Sample.

weird_algorithm.inp	weird_algorithm.out
3	3 10 5 16 8 4 2 1

Source: CSES Problem Set/weird algorithm, & [Laa20, Sect. 1.3, pp. 5–7].

```
#include <iostream>
using namespace std;

int main() {
        long long n;
        cin >> n;
        while (1) {
                cout << n << " ";
                if (n == 1) break;
                if (n%2 == 0) n /= 2;
                else n = n*3 + 1;
        }
        cout << "\n";
}
```

Remark 1 (Collatz conjecture)**.** *The above algorithm terminates* $\forall n \in \mathbb{N}^\star$.

Problem 27 (Missing Number)**.** *You are given all numbers between* $1, 2, \ldots, n$ *except one. Your task is to find the missing number.*

- Input. *The 1st input line contains a positive integer* $n \in \mathbb{N}^\star$*. The 2nd line contains* $n-1$ *numbers. Each number is distinct & between* 1 *&* n *(inclusive).*
- Output. *Print the missing number.*
- Constraints. $2 \leq n \leq 2 \cdot 10^5$.
- Sample.

missing_number.inp	missing_number.out
5 2 3 1 5	4

Source: CSES Problem Set/missing number.

```
file_in = open("missing_number.inp")
file_out = open("missing_number.out", "w")
n = int(file_in.readline())
data = file_in.readline()
A = data.split()
A = [int(i) for i in A]
A.sort()
for i in range(n-1):
if i + 1 != A[i]:
file_out.write(str(i + 1))
break
file_in.close()
file_out.close()
```

Problem 28 (Repetitions)**.** *You are given a DNA sequence: a string consisting of characters A, C, G, & T. Your task is to find the longest repetition in the sequence. This is a maximum-length substring containing only 1 type of character.*

- Input. *The only input line contains a string of* $n \in \mathbb{N}^\star$ *characters.*
- Output. *Print 1 integer: the length of the longest repetition.*
- Constraints. $1 \leq n \leq 10^6$.

- Sample.

repetition.inp	repetition.out
ATTCGGGA	3

Source: CSES problem Set/repetition.

```
DNA = input()
DNA_count = []
count = 1
char = DNA[0]
for i in range(1, len(DNA)):
if DNA[i] == DNA[i-1]:
count = count + 1
else:
DNA_count.append(count)
count = 1
DNA_count.append(count)
print(max(DNA_count))
```

Problem 29 (Non-decreasing Array)**.** *You are given an array of n integers. You want to modify the array so that it is non-decreasing, i.e., every element is at least as large as the previous element. On each move, you may increase the value of any element by 1. What is the minimum number of moves required?*

- Input. *The 1st input line contains an integer n: the size of the array. The 2nd line contains n integers $x_1, x_2, \ldots, x_n$: the contents of the array.*
- Output. *Print the minimum number of moves.*
- Constraints. $1 \leq n \leq 2 \cdot 10^5$, $1 \leq x_i \leq 10^9$.
- Sample.

nondecreasing_array.inp	nondecreasing_array.out
5 3 2 5 1 7	5

Source: CSES problem Set/increasing array.

```
n = int(input())
A = input()
A = A.split()
A = [int(i) for i in A]
ans = 0
for i in range(1, len(A)):
if A[i] < A[i - 1]:
ans = ans + A[i-1] - A[i]
A[i] = A[i - 1]
print(ans)
```

Problem 30 (Permutations)**.** *A permutation of integers $1, 2, \ldots, n$ is called* beautiful *if there are no adjacent elements whose difference is 1. Given n, construct a beautiful permutation if such a permutation exists.*

- Input. *The only input line contains an integer n.*
- Output. *Print a beautiful permutation of integers $1, 2, \ldots, n$. If there are several solutions, you may print any of them. If there are no solutions, print* `"NO SOLUTION"`.
- Constraints. $1 \leq n \leq 10^6$.
- Sample.

permutations.inp	permutations.out
5	4 2 5 3 1
3	NO SOLUTION

```
n = int(input())
A = [1]
if n == 1:
print(1)
elif n == 2 or n == 3:
print("NO SOLUTION")
elif n == 4:
print("2 4 1 3")
else:
for i in range(1, n):
tmp = A[i-1] + 2
if tmp <= n:
A.append(tmp)
else:
A.append(2)
print(*A)
```

Problem 31 (Number Spiral). *A* number spiral *is an infinite grid whose upper-left square has number* 1. *Here are the 1st 5 layers of the spiral:*

1	2	9	10	25
4	3	8	11	24
5	6	7	12	23
16	15	14	13	22
17	18	19	20	21

Your task is to find out the number in row y & column x.

- Input. *The 1st input line contains an integer t: the number of tests. After this, there are t lines, each containing integers y, x.*
- Output. *For each test, print the number in row y and column x.*
- Constraints. $1 \le t \le 10^5$, $1 \le x, y \le 10^9$.
- Sample.

number_spiral.inp	number_spiral.out
3 2 3 1 1 4 2	8 1 15

16 Problems in Elementary Mathematics – Bài Toán Tin Học Trong Toán Học Sơ Cấp

169 (Even vs. odd). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xét tính chẵn lẻ của $n \in \mathbb{Z}$ được nhập từ bàn phím.*

170 (Divisible by). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để kiểm tra liệu $a \vdots b$ hay không, với $a, b \in \mathbb{Z}$ được nhập từ bàn phím.*

171 (Triangle). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để liệu a, b, c có phải là độ dài của: (a) 1 tam giác. (b) 1 tam giác nhọn. (c) 1 tam giác vuông. (d) 1 tam giác tù.*

172 (Polynomial equation). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc nhất, bậc 2, bậc 3, & bậc 4 với các hệ số thực được nhập từ bàn phím.*

173 (Fibonacci sequence). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để xuất ra màn hình, với $n \in \mathbb{N}$ được nhập từ bàn phím: (a) Số Fibonacci thứ n. (b) n số Fibonacci đầu tiên.*

174 (1st n square roots). *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 2 của n số tự nhiên đầu tiên với $n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

175 (Số chính phương – Square number). *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số $n \in \mathbb{N}^\star$ được nhập từ bàn phím có phải là số chính phương hay không.*

176 (1st n cube roots)**.** *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc 3 của n số tự nhiên đầu tiên với $n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

177. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số $n \in \mathbb{N}^\star$ được nhập từ bàn phím có phải là lập phương của 1 số tự nhiên hay không.*

178 (1st n nth roots)**.** *Viết chương trình* Pascal, C/C++, Python *xuất ra căn bậc n của m số tự nhiên đầu tiên với $m, n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

179. *Viết chương trình* Pascal, C/C++, Python *để kiểm tra 1 số m được nhập từ bàn phím có phải là lũy thừa bậc n của 1 số tự nhiên hay không với $m, n \in \mathbb{N}^\star$ được nhập từ bàn phím.*

16.1 Algebraic Expression – Biểu Thức Đại Số

180 ([Vie21], 1., p. 15, Vũng Tàu 2020)**.** *Cho $a, b, c \in \mathbb{N}^\star$.* Yêu cầu: *Tính giá trị của biểu thức $S = \dfrac{a^2+b^2+c^2}{abc} + \sqrt{abc}$.*

- Dữ liệu vào: *File* `root.inp` *chứa 3 số nguyên dương a, b, c. Mỗi số trên 1 dòng.*
- Kết quả: *Ghi vào File* `root.out` *kết quả S tính được (làm tròn lấy 2 chữ số sau phần thập phân). E.g.:*

`root.inp`	`root.out`
2 1 2	4.25

181 ([Vie21], 2., p. 19, Bắc Giang 2020)**.** *Nhà An có 1 trang trại rộng lớn. Do sở thích của An nên bố An chỉ nuôi gà & chó. 1 hôm bố An đố con gái nhà mình nuôi bao nhiêu gà, bao nhiêu chó? Bố An cho biết nhà có tổng số gà & chó là x con. Do số lượng nhiều & khó đếm từng loại nên An chỉ đếm được tổng số chân của gà & chó là y chân. Giúp An trả lời câu đố.*

- Dữ liệu vào: *Đọc từ file văn bản* `toanco.inp` *gồm 2 số nguyên dương x, y trên 1 dòng. 2 số cách nhau 1 khoảng trống ($x \leq 10^5$, $y \leq 4 \cdot 10^5$).*
- Kết quả: *Ghi ra file văn bản* `toanco.out` *gồm 2 số tương ứng là số gà & số chó tìm được. 2 số cách nhau 1 khoảng trống. Giả sử bài toán luôn có nghiệm.*

`toanco.inp`	`toanco.out`
36 100	22 14

182 ([Vie21], 4., p. 26, Quãng Ngãi 2020, Lãi suất– Interest rate)**.** *1 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).*

- Yêu cầu: *Tính số tiền người đó nhận được sau t tháng. Biết 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 1 ký tự trắng.*
- Dữ liệu ra: *Tệp văn bản* `bl2.out` *ghi 1 số là số tiền nhận được (làm tròn đến 1 số lẻ sau dấu chấm thập phân). E.g.:*

`bl2.inp`	`bl2.out`
12 13 100 1.0 0.2	112.9
0 10 100 1.0 0.2	102.0

16.2 Number Theory – Số Học

183 ([Vie21], 3., p. 20, Yên Bái 2020, Tổng nguyên tố)**.** *Viết chương trình nhập vào 2 số nguyên $a, b \in \mathbb{Z}$, $0 < a < b$. (a) Tìm & tính tổng các số nguyên tố của dãy số từ a đến b. (b) Xuất ra màn hình các số chia hết cho 5 của dãy số từ a đến b. (c)* (Bội của $n \in \mathbb{N}^\star$) *Xuất ra màn hình các số chia hết cho n của dãy số từ a đến b với $n \in \mathbb{N}^\star$ được nhập từ bàn phím. E.g., nhập $a = 6$, $b = 22$. Kết quả tổng các số nguyên tố trong dãy số từ 6 đến 22: $7 + 11 + 13 + 17 + 19 = 67$. Các số chia hết cho 5 của dãy số từ 6 đến 22: $10, 15, 20$.*

184 ([Vie21], 4., p. 22, Hải Dương 2020, Số mạnh mẽ)**.** Số mạnh mẽ *là số khi nó chia hết cho 1 số nguyên tố thì cũng chia hết cho cả bình phương của số nguyên tố đó, i.e., $a \in \mathbb{N}^\star$ là số mạnh mẽ $\Leftrightarrow (a \vdots p \Rightarrow a \vdots p^2, \forall p$: prime). E.g., 25 là số mạnh mẽ, vì nó chia hết cho số nguyên tố 5 & chia hết cho cả $5^2 = 25$. Viết chương trình liệt kê các số mạnh mẽ không vượt quá 1000.*

See, e.g., Wikipedia/powerful number, MathWorld/powerful number.

185 ([Vie21], 5., p. 23, Việt Nam 2020, Bội chính phương). *Cho 1 dãy số A có n phần tử. Tìm số nguyên dương P nhỏ nhất thỏa mãn: a là số chính phương & a chia hết cho tất cả các phần tử của dãy số A.*

- Yêu cầu: *In ra phần dư của phép chia khi chia a cho $10^9 + 7$.*
- Dữ liệu vào: *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 1 dòng được ghi cách nhau bởi dấu cách.*
- Kết quả: *Ghi ra thiết bị ra gồm 1 số nguyên duy nhất là kết quả của bài toán. E.g.:*

Dữ liệu vào	Dữ liệu ra
3 2 1 3	36

186 ([Vie21], 1., p. 25, Hải Dương 2020, Số hạnh phúc & số buồn bã – Happy- & sad numbers). *Với 1 số nguyên dương bất kỳ, thay thế số đó bằng tổng bình phương các chữ số của nó & cứ lặp lại quá trình đó sẽ có các trường hợp sau xảy ra: Kết thúc bằng 1 – ta gọi số đó là* số hạnh phúc/happy number. *Kết thúc bằng 0 – ta gọi số đó là* số buồn bã/sad number. *Lặp lại vô hạn lần – số đó không hạnh phúc cũng không buồn bã. E.g., số 44: lần 1: $4^2 + 4^2 = 32$, lần 2: $3^2 + 2^2 = 13$, lần 3: $1^2 + 3^2 = 10$, lần 4: $1^2 + 0^2 = 1$, nên 44 là số hạnh phúc. Viết chương trình để kiểm tra xem ngày sinh của 1 người bất kỳ có phải là số hạnh phúc không?*

187 ([Vie21], 2., p. 25, Gia Lai 2019, Phân số tối giản – Irreducible fraction). *1 chuỗi được gọi là có dạng phân số nếu nó có dạng* `'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. E.g., Chuỗi* `'12/15'` *biểu diễn cho phân số. Dạng tối giản của phân số đó là* `'3/5'`.

188 (Tổng tất cả, tổng phần tử chẵn, lẻ, bình phương, lập phương, lũy thừa bậc n, căn bậc 2, 3, & căn bậc n, nghịch đảo, nghịch đảo bình phương, nghịch đảo lập phương, nghịch đảo lũy thừa bậc n, nghịch đảo căn bậc 2, 3, & nghịch đảo căn bậc n – Sums of all, odds, evens, squares, cubes, nth powers, square roots, cube roots, nth roots, reciprocals of square, of cubes, of nth powers, of square roots, of cube roots, of nth roots). *Cho 1 dãy gồm n số nguyên: $(a_i)_{i=1}^m = a_1, a_2, \ldots, a_m$, $m \in \mathbb{N}^\star$, $a_i \in \mathbb{Z}$, $\forall i = 1, 2, \ldots, m$, mỗi số có giá trị không vượt quá 10^9.*

- Yêu cầu: *Tính tổng S tất cả các phần tử, tổng S_{even} các số chẵn, tổng S_{odd} các số lẻ, tổng S_{sqr} bình phương, tổng $S_{\text{sqr,even}}$ bình phương các số chẵn, tổng $S_{\text{sqr,odd}}$ bình phương các số lẻ, tổng S_{cb} lập phương, tổng $S_{\text{cb,even}}$ lập phương các số chẵn, tổng $S_{\text{cb,odd}}$ lập phương các số lẻ, tổng $S_{\text{pwr},n}$ lũy thừa bậc n, tổng $S_{\text{pwr,even},n}$ lũy thừa bậc n các số chẵn, tổng $S_{\text{pwr,odd},n}$ lũy thừa bậc n các số lẻ, tổng S_{sqrt} căn bậc 2, tổng $S_{\text{sqrt,even}}$ căn bậc 2 các số chẵn, tổng $S_{\text{sqrt,odd}}$ căn bậc 2 các số lẻ, tổng S_{cbrt} căn bậc 3, tổng $S_{\text{cbrt,even}}$ căn bậc 3 các số chẵn, tổng $S_{\text{cbrt,odd}}$ căn bậc 3 các số lẻ, tổng $S_{\text{rt},n}$ căn bậc n của các số, tổng $S_{\text{rt,even},n}$ căn bậc n của các số chẵn, tổng $S_{\text{rt,odd},n}$ căn bậc n của các số lẻ trong dãy $(a_i)_{i=1}^m$.*
- Dữ liệu: *Dòng đầu tiên chứa $m \in \mathbb{N}^\star$, $1 \le m \le 10^9$. Dòng thứ 2 chứa $n \in \mathbb{N}^\star$. m dòng tiếp theo, dòng thứ $i + 2$ chứa a_i, $\forall i = 1, 2, \ldots, m - 1$.*

Hint. Công thức toán học tính các tổng:

$$S \coloneqq \sum_{i=1}^{m} a_i = a_1 + a_2 + \cdots + a_m,\ S_{\text{even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i,$$

$$S_{\text{sqr}} \coloneqq \sum_{i=1}^{m} a_i^2 = a_1^2 + a_2^2 + \cdots + a_m^2,\ S_{\text{sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^2,\ S_{\text{sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^2,$$

$$S_{\text{cb}} \coloneqq \sum_{i=1}^{m} a_i^3 = a_1^3 + a_2^3 + \cdots + a_m^3,\ S_{\text{cb,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^3,\ S_{\text{odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^3,$$

$$S_{\text{pwr},n} \coloneqq \sum_{i=1}^{m} a_i^n = a_1^n + a_2^n + \cdots + a_m^n,\ S_{\text{pwr,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} a_i^n,\ S_{\text{pwr,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} a_i^n,\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{sqrt}} \coloneqq \sum_{i=1}^{m} \sqrt{a_i} = \sqrt{a_1} + \sqrt{a_2} + \cdots + \sqrt{a_m},\ S_{\text{sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt{a_i},\ S_{\text{sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt{a_i},$$

$$S_{\text{cbrt}} \coloneqq \sum_{i=1}^{m} \sqrt[3]{a_i} = \sqrt[3]{a_1} + \sqrt[3]{a_2} + \cdots + \sqrt[3]{a_m},\ S_{\text{cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[3]{a_i},\ S_{\text{cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[3]{a_i},$$

$$S_{\text{rt},n} \coloneqq \sum_{i=1}^{m} \sqrt[n]{a_i} = \sqrt[n]{a_1} + \sqrt[n]{a_2} + \cdots + \sqrt[n]{a_m},\ S_{\text{rt,even},n} \coloneqq \sum_{i=1,\, 2|a_i}^{m} \sqrt[n]{a_i},\ S_{\text{rt,odd},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \sqrt[n]{a_i},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i} = \frac{1}{a_1} + \frac{1}{a_2} + \cdots + \frac{1}{a_m},\ S_{\text{rcpc,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{a_i},\ S_{\text{rcpc,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i},$$

$$S_{\text{rcpc,sqr}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^2} = \frac{1}{a_1^2} + \frac{1}{a_2^2} + \cdots + \frac{1}{a_m^2},\ S_{\text{rcpc,sqr,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{a_i^2},\ S_{\text{rcpc,sqr,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^2},$$

$$S_{\text{rcpc,cb}} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^3} = \frac{1}{a_1^3} + \frac{1}{a_2^3} + \cdots + \frac{1}{a_m^3},\ S_{\text{rcpc,cb,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{a_i^3},\ S_{\text{rcpc,cb,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^3},$$

$$S_{\text{rcpc,pwr},n} \coloneqq \sum_{i=1}^{m} \frac{1}{a_i^n} = \frac{1}{a_1^n} + \frac{1}{a_2^n} + \cdots + \frac{1}{a_m^n},\ S_{\text{rcpc,even,pwr},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{a_i^n},\ S_{\text{rcpc,odd,pwr},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{a_i^n},\ \forall n \in \mathbb{N}^\star,$$

$$S_{\text{rcpc,sqrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt{a_i}} = \frac{1}{\sqrt{a_1}} + \frac{1}{\sqrt{a_2}} + \cdots + \frac{1}{\sqrt{a_m}},\ S_{\text{rcpc,sqrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt{a_i}},\ S_{\text{rcpc,sqrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt{a_i}},$$

$$S_{\text{rcpc,cbrt}} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[3]{a_i}} = \frac{1}{\sqrt[3]{a_1}} + \frac{1}{\sqrt[3]{a_2}} + \cdots + \frac{1}{\sqrt[3]{a_m}},\ S_{\text{rcpc,cbrt,even}} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt[3]{a_i}},\ S_{\text{rcpc,cbrt,odd}} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[3]{a_i}},$$

$$S_{\text{rcpc,rt},n} \coloneqq \sum_{i=1}^{m} \frac{1}{\sqrt[n]{a_i}} = \frac{1}{\sqrt[n]{a_1}} + \frac{1}{\sqrt[n]{a_2}} + \cdots + \frac{1}{\sqrt[n]{a_m}},\ S_{\text{rcpc,even,rt},n} \coloneqq \sum_{i=1,\, 2|a_i,\, a_i\neq 0}^{m} \frac{1}{\sqrt[n]{a_i}},\ S_{\text{rcpc,odd,rt},n} \coloneqq \sum_{i=1,\, 2\nmid a_i}^{m} \frac{1}{\sqrt[n]{a_i}},\ \forall n \in \mathbb{N}^\star.$$

Dựa vào các công thức này, sử dụng vòng lặp `for` hoặc `while` để tính các tổng này. □

189 ([Vie21], 5., p. 26, Nghệ An 2019, Giả thuyết Goldbach cho số nguyên tố – Goldbach conjecture for primes). *Minh đố An: Cho 1 số chẵn $k \in \mathbb{N}$, $2 \leq k \leq 1000$, tìm 2 số nguyên tố sao cho tổng của chúng bằng số chẵn k đã cho.*

- Yêu cầu: *Viết chương trình* Pascal, Python, C/C++ *giúp An trả lời câu hỏi của Minh.*
- Dữ liệu vào: *Tệp văn bản* `prime.inp`*: Dòng đầu tiên chứa $n \in \mathbb{N}^\star$ tương ứng số test. n dòng tiếp theo, mỗi dòng chứa 1 số k, i.e., k_i, $i = 1, 2, \ldots, n$.*
- Dữ liệu ra: *Tệp văn bản* `prime.out` *gồm n dòng tương ứng n kết quả. Mỗi kết quả hiển thị tổng 2 số nguyên tố bằng số k nhập vào. E.g.:*

`prime.inp`	`prime.out`
2	$8 = 5 + 3$
8	$24 = 19 + 5$
24	

190 ([Vie21], 6., p. 27, Tây Ninh 2019, Số hoàn hảo – Perfect number). Số hoàn hảo *là 1 số tự nhiên mà tổng tất cả các ước tự nhiên thực sự của nó bằng chính nó. Trong đó ước thực sự của 1 số là các ước dương không bằng số đó. Lập trình nhập vào 1 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.*

191 ([Vie21], 7., p. 27, Đồng Nai 2020, Số may mắn – Lucky number). *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 1 dãy số phần thưởng. Mỗi bạn sẽ bốc thăm 1 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 & k, $1 \leq k \leq n \leq 1000$.*
- Dữ liệu ra: *In số của món quà học sinh đạt được:*

`lucky_number.inp`	`lucky_number.out`
10 6	2

192 ([Vie21], 8., p. 27, Ninh Bình 2019, Ước chung lớn nhất ƯCLN – greatest common divisor gcd). *Nhập vào 3 số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *của 3 số này. E.g., nhập vào 3 số: $4, 6, 12$ thì kết quả* ƯCLN *là 2.*

193 (Bội chung nhỏ nhất BCNN – least common multiplier lcd). *Nhập vào $n \in \mathbb{N}^\star$ số từ bàn phím, kiểm soát dữ liệu nhập vào là số nguyên dương. Lập trình tìm* ƯCLN *&* BCNN *của n số này.*

17 Character & String – Xâu & Chuỗi

194 ([Vie21], 1., p. 28, Tây Ninh 2019, Số đảo ngược – Reversed number). *Tìm số đảo ngược y của 1 số $x \in \mathbb{Z}$ biết y gồm các chữ số của x & viết theo thứ tự ngược lại. Xuất ra kết quả là số $y \mod 19$.* Dữ liệu: $x \in \mathbb{N}^\star$. Kết quả: $y \mod 19$ *với y là số đảo ngược của x.*

`reversed_number.inp`	`reversed_number.out`	Giải thích
123	17	Đảo ngược của 123 là 321 & $321 \mod 19 = 17$

195 ([Vie21], 2., pp. 28–29, Bắc Giang 2020, Nén xâu – String compression). *Viết chương trình nhập vào 1 xâu ký tự chỉ gồm các chữ cái Tiếng Anh, chữ số, dấu cách, & dấu gạch nối.*

- Yêu cầu: *Nén các ký tự liên tiếp giống nhau thành số lượng ký tự & ký tự đó, rồi đưa ra xâu sau khi nén.*
- Dữ liệu vào: *Đọc từ file văn bản* `string_compression.inp` *gồm 1 xâu S có số lượng ký tự không quá* 255 *ký tự.*
- Dữ liệu ra: *Đưa ra file văn bản* `string_compression.out` *gồm xâu S sau khi nén.*

`string_compression.inp`	`lstring_compression.out`
aaaababb cc	4a1b1a2b4 2c

196 ([Vie21], 3., p. 30, Hậu Giang 2020, Tính nhân – Multiplication). *Viết chương trình nhập vào 2 số nguyên dương $a, b \in \mathbb{N}^\star$. Sau đó thực hiện nhân $a \times b$ như cách nhân bằng tay thông thường ở tiểu học. E.g., nhập vào thừa số thứ 1:* 125*, nhập vào thừa số thứ 2:* 15. Dữ liệu ra:

```
125
x
15
-----
625
125
-----
1875
```

18 1D Array & List – Mảng 1 Chiều & Danh Sách

19 Matrix – Ma Trận

20 Arrangement – Sắp Xếp

See, e.g., [Knu98, Chap. 5: Sorting], [Vie21, Chap. II, Sect. Dạng bài sắp xếp].

197 ([Vie21], 1., p. 83, Bắc Giang 2019, Dãy số – Sequence). *Sử dụng hàm Randomize để khởi tạo dãy số ngẫu nhiên từ* 0 *đến* 9 *gồm $n \in \mathbb{N}^\star$ phần tử, $0 < n \leq 100$, kết quả ghi ra tệp* `random.out`*, mỗi phần tử cách nhau* 1 *dấu cách.*

- Yêu cầu: *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.*
- Dữ liệu ra: *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* $0, 1, \ldots, 9$*. E.g.:*

`random_sequence.inp`	`random_sequence.out`
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

198 ([Vie21], 2., p. 85, Vị Thanh, Hậu Giang 2019, Dãy số không giảm – Nondecreasing sequence)**.** *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, e.g.,* $a_1 = 234$, $a_2 = 123$, $a_3 = 345$ *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 & xuất ra màn hình, các số cách nhau 1 khoảng trắng. E.g.:*

nondecreasing_sequence.inp	nondecreasing_sequence.out
$a_1 = 234$ $a_2 = 123$ $a_3 = 345$	123234 123345 234123 234345 345123 345234

199 ([Vie21], 1., p. 87, Sorting ascending)**.** *Cho vào m dãy số nguyên* $(a_{i,j})_{j=1}^{n_i}$, $n_i \in \mathbb{N}^\star$, $n_i \leq 100$, $\forall i = 1, 2, \ldots, m$, $|a_{ij}| < 32000$, $\forall i = 1, 2, \ldots, m$, $\forall j = 1, 2, \ldots, \max\{n_i | i = 1, 2, \ldots, m\}$.

- Yêu cầu: *Sắp xếp từng dãy số trên theo thứ tự tăng dần.*
- Dữ liệu vào: *Trong m dòng, mỗi dòng là dãy số, bắt đầu là 1 số nguyên n là số lượng các phần tử của dãy số,* $1 \leq n \leq 100$, *n số nguyên tiếp theo là giá trị các phần tử của dãy.*
- Dữ liệu ra: *Ghi ra m dòng là m dãy số đã được sắp xếp theo thứ tự tăng dần. E.g.:*

sorting_ascending.inp	sorting_ascending.out
2 2 1	1 2
3 4 3 1	1 3 4
4 1 4 5 2	1 2 4 5

21 Algorithm – Thuật Toán

21.1 Recursion algorithm – Thuật toán đệ quy

Đệ quy (recursion) là phương pháp dùng trong các chương trình máy tính trong đó có 1 hàm tự gọi chính nó.

200 ([Vie21], p. 91, Giai thừa – Factorial $n!$)**.** *Tính* $n! = \prod_{i=1}^{n} i = 1 \cdot 2 \cdots (n-1)n$.

- Input: *Dòng đầu là số lượng test. Mỗi dòng tiếp theo gồm 1 số* $n \in \mathbb{N}$.
- Output: *Với mỗi test, in ra n! theo mẫu:*

factorial.inp	factorial.out
2	3! = 6
3	4! = 24
4	

201 ([Vie21], 1., p. 92, Hàm $f91$ của McCarthy – McCarthy's $f91$ function)**.** *McCarthy là 1 nhà khoa học máy tính nổi tiếng, ông đã định nghĩa hàm đệ quy* $f91 : \mathbb{Z} \to \mathbb{Z}$ *như sau:*

$$f_{91}(n) = \begin{cases} f_{91}(f_{91}(n+11)), & \text{if } n \leq 100, \\ n - 1, & \text{if } n \geq 101. \end{cases}$$

Viết 1 chương trình tính toán hàm McCarthy's f91.

- Input: *File input chứa 1 dãy các số nguyên dương, mỗi số không quá* 1000. *Mỗi số trên 1 dòng.*
- Output: *Hiện ra theo định dạng trong ví dụ sau:*

McCarthy_f91_function.inp	McCarthy_f91_function.out
500	f91(500) = 490
91	f91(91) = 91

See, e.g., Wikipedia/McCarthy 91 function.

202 ([Vie21], 2., p. 93, Chỉnh hợp – Arrangement A_n^k)**.** *Tìm tất cả các chỉnh hợp chập k của n phần tử từ* 1 *đến n,* $0 < k \leq n < 10$.

- Input: *File input chứa 1 dòng gồm* $n, k \in \mathbb{N}$.

- Output: *In ra số chỉnh hợp tìm được & liệt kê các chỉnh hợp, giữa các test là 1 dòng trắng, e.g.,*

arrangement.inp	arrangement.out	arrangement.inp	arrangement.out
1 3	3 11 13 33	2 3	6 1 2 1 3 2 1 2 3 3 1 3 2

203 ([Vie21], 3., p. 93, Hoán vị – Permutation $P_n = n!$). *Viết chương trình in ra tất cả các hoán vị của $n \in \mathbb{N}^\star$ được nhập từ bàn phím, e.g.,*

permutation.inp	permutation.out
2	1 2 2 1
3	1 2 3 1 3 2 2 1 3 2 3 1 3 2 1 3 1 2

204 ([Vie21], 5., p. 94, Tổ hợp – Combinatoric C_n^k). *Tìm tất cả các tổ hợp chập k của n phần tử từ 1 đến n, $0 < k \leq n < 10$.*

- Input: *File input có dòng đầu gồm 1 số tự nhiên* `ntest` *là số lượng test của file* `combinatoric.inp`. *Mỗi test 1 dòng gồm 2 số $n, k \in \mathbb{Z}$.*
- Output: *Với mỗi test, in ra số tổ hợp tìm được & liệt kê các tổ hợp, giữa các test là 1 dòng trắng. E.g.:*

combinatoric.inp	combinatoric.out
2 1 3 2 3	3 1 2 3 3 1 2 1 3 2 3

205 ([Vie21], 4., pp. 93–94, Trò chơi số học – Number theory game). *1 trò chơi phổ biến của trẻ em với bảng $n \times n$ ô $2 \leq n \leq 5$. Trong mỗi ô chứa 1 số có 1 chữ số từ 1 tới 9. Với 1 số số cho trước, điền các số còn lại vào ô sao cho tổng các hàng ngang bằng nhau & bằng tổng các hàng dọc.*

- Input: `number_theory_game.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 $n \times n$ trong đó số 0 là ô trắng cần điền.*
- Output: `number_theory_game.out`*: Với mỗi test, nếu có thể tìm ra đáp án thỏa mãn quy luật của bảng thì in ra ma trận $n \times n$ 1 cách điền bất kỳ. Nếu không có kết quả nào in ra 1 chuỗi:* `"cannot find."`. *Giữa các test cách nhau 1 dòng trắng. E.g.:*

number_theory_game.inp	number_theory_game.out
1 3 1 0 5 0 3 0 3 0 4	1 4 5 6 3 1 3 3 4

21.2 Search algorithm – Thuật toán tìm kiếm

Tìm kiếm & sắp xếp là 2 hoạt động hằng ngày ta thường sử dụng trong các ứng dụng tin học.
See, e.g., [Knu98, Chap. 6: Searching].

22 Problem in Elementary Physics – Bài Toán Tin Học Trong Vật Lý Sơ Cấp

23 Problem in Elementary Chemistry – Bài Toán Tin Học Trong Hóa Học Sơ Cấp

24 Miscellaneous

206 ([BTC10], 1., p. 5, Connect). *Cho n số nguyên dương $a_1, a_2, \ldots, a_n$, $n \in \mathbb{N}$, $1 < n \leq 100$, $0 < a_i \leq 10^9$, $\forall i = 1, 2, \ldots, n$. Từ các số nguyên này người ta tạo ra 1 số nguyên mới bằng cách kết nối tất cả các số đã cho viết liên tiếp nhau. E.g., với $n = 4$ & các số $12, 34, 567, 890$ ta có thể tạo ra các số mới như sau:* 1234567890, 3456789012, 8905673412, ... *Dễ thấy có $4! = 24$ cách tạo mới như vậy. Trong trường hợp này, số lớn nhất có thể tạo thành là* 8905673412.

- Yêu cầu: *Cho n & các số $a_1, a_2, \ldots, a_n$. Xác định số lớn nhất có thể kết nối được theo quy tắc trên.*
- Dữ liệu vào: *Cho trong file văn bản* `connect.inp` *gồm $n + 1$ dòng. Dòng đầu tiên ghi số nguyên n. Trong các dòng còn lại, dòng thứ $i + 1$ ghi số a_i.*
- Dữ liệu ra: *Ghi vào file văn bản* `connect.out` *số lớn nhất được kết nối thành từ các số ban đầu. E.g.:*

`connect.inp`	`connect.out`
4 12 34 567 890	8905673412

24.1 Google Kickstart Round A 2020

Watch YouTube/William Lin/Winning Google Kickstart Round A 2020.

Problem 32 (Google Kickstart Round A 2020, Allocation). *There are n houses for sale. The ith house costs a_i dollars to buy. You have a budget of b dollars to spend. What is the maximum number of houses you can buy?*

- Input: *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a single line containing the 2 integers n, b. The 2nd line contains n integers. The ith integer is a_i, the cost of the ith house.*
- Output: *For each test case, output 1 line containing* `Case #x: y`, *where* **`x`** *is the test case number (starting from 1) &* **`y`** *is the maximum number of houses you can buy.*
- Limits: *Time limit:* 15 s/test set. *Memory limit:* 1GB. $1 \leq t \leq 100$, $1 \leq b \leq 10^5$, $1 \leq a_i \leq 1000$, $\forall i = 1, 2, \ldots, n$. *Test set 1:* $1 \leq n \leq 100$. *Test set 2:* $1 \leq n \leq 10^5$.
- Sample:

`allocation.inp`	`allocation.out`
3 4 100 20 90 40 90 4 50 30 30 10 10 3 300 999 999 999	Case #1: 2 Case #2: 3 Case #3: 0

Problem 33 (Google Kickstart Round A 2020, Plates). *Dr. Patel has n stacks of plates. Each stack contains k plates. Each plate has a positive beauty value, describing how beautiful it looks. Dr. Patel would like to take exactly p plates to use for dinner tonight. If he would like to take a plate in a stack, he must also take all of the plates above it in that stack as well. Help Dr. Patel pick the p plates that would maximize the total sum of beauty values.*

- Input: *The 1st line of the input gives the number of test cases, t. t test cases follow. Each test case begins with a line containing the 3 integers n, k, p. Then, n lines follow. The ith line contains k integers, describing the beauty values of each stack of plates from top to bottom.*
- Output: *For each test case, output 1 line containing* `Case #x: y`, *where* **`x`** *is the test case number (starting from 1) &* **`y`** *is the maximum total sum of beauty values that Dr. Patel could pick.*

- Limits: *Time limit:* 20 s/test set. *Memory limit:* 1GB. $\leq t \leq 100$, $1 \leq k \leq 30$, $1 \leq p \leq nk$. *The beauty values are between* 1 *&* 100, *inclusive. Test set 1:* $1 \leq n \leq 3$. *Test set 2:* $1 \leq n \leq 50$.
- Sample:

plate.inp	plate.out
2 2 4 5 10 10 100 30 80 50 10 50 3 2 3 80 80 15 50 20 10	Case #1: 250 Case #2: 180

207 (Polynomial equation – Phương trình đa thức). *Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc nhất, bậc 2, bậc 3, & bậc 4 với các hệ số thực được nhập từ bàn phím.*

208 (A special cubic equation – 1 dạng phương trình bậc 3 đặc biệt). *(a) Viết thuật toán & các chương trình bằng các ngôn ngữ lập trình* Pascal, Python, C/C++ *để giải phương trình bậc 3 có dạng đặc biệt* $x^3 + 3a^2x + 2b = 0$ *(khuyết số hạng* x^2*), với* $p, q \in \mathbb{R}$ *được nhập từ bàn phím. Biết phương trình này có 1 nghiệm duy nhất là*[17]*:*

$$x = \sqrt[3]{\sqrt{a^6 + b^2} - b} - \sqrt[3]{\sqrt{a^6 + b^2} + b}.$$

(b) Chứng minh chặt chẽ bằng toán học công thức nghiệm đã cho.

C: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/C/special_cubic_eqn.c.

```c
#include <math.h>
#include <stdio.h>
int main() {
        double a, b, x, delta, test;
        printf("a = ");
        scanf("%lf", &a);
        printf("b = ");
        scanf("%lf", &b);
        delta = pow(a, 6.0) + pow(b, 2.0);
        delta = sqrt(delta);
        x = pow(delta - b, 1.0/3) - pow(delta + b, 1.0/3);
        printf("Root x = %lf.\n", x);
        test = x*x*x + 3*a*a*x + 2*b;
        printf("x^3 + 3a^2x + 2b = %lf.\n", test);
}
```

C++: https://github.com/NQBH/hobby/blob/master/elementary_computer_science/Cpp/special_cubic_eqn.cpp.

```cpp
#include <cmath>
#include <iostream>
using namespace std;
int main() {
        double a, b, x, delta;
        cout << "a = ";
        cin >> a;
        cout << "b = ";
        cin >> b;
        delta = pow(a, 6.0) + pow(b, 2.0);
        delta = sqrt(delta);
        x = pow(delta - b, 1.0/3) - pow(delta + b, 1.0/3);
        cout << "Root x = " << x << ".\n";
        double test = x*x*x + 3*a*a*x + 2*b;
        cout << "x^3 + 3a^2x + 2b = " << test << ".\n";
}
```

209 (Tổng quát ⋆). *Cho* ΔABC *vuông tại* A. *(a) Cho trước 2 trong 6 số* a, b, c, b', c', h. *Tính 4 số còn lại theo 2 số đã cho. (c) Cho trước 2 trong 8 số* a, b, c, b', c', h, p, S. *Tính 6 số còn lại theo 2 số đã cho. (b) Cho trước 2 trong 14 số* $a, b, c, b', c', h, m_a, m_b, m_c, d_a, d_b, d_c, p,$ *với* d_a, d_b, d_c *lần lượt là 3 đường phân giác ứng với* BC, CA, AB. *Tính 12 số còn lại theo 2 số đã cho. Viết các chương trình* Pascal, Python, C/C++ *để mô phỏng.*

[17] See. e.g., [Thư+21, pp. 68–69].

References

[BTC10] BTC. *Tuyển Tập Đề Thi Olympic 30 Tháng 4, Lần Thứ XVI – 2010 Tin học*. Nhà Xuất Bản Đại Học Sư Phạm, 2010, p. 285.

[DT06] Lê Văn Doanh and Trần Khắc Tuấn. *101 Thuật Toán & Chương Trình Bài Toán Khoa Học Kỹ Thuật & Kinh Tế Bằng Ngôn Ngữ Turbo-Pascal*. In lần thứ 10. Nhà Xuất Bản Khoa Học & Kỹ Thuật, 2006, p. 268.

[Đức22] Nguyễn Tiến Đức. *Tuyển Tập 200 Bài Tập Lập Trình Bằng Ngôn Ngữ Python*. Nhà Xuất Bản Đại Học Thái Nguyên, 2022, p. 327.

[DV21] Christoph Dürr and Jill-Jênn Vie. *Competitive Programming in Python: 128 Algorithms to Develop Your Coding Skills*. Translated by Greg Gibbons & Danièle Gibbons. Cambridge University Press, 2021, pp. x+254.

[Huy24] Nguyễn Xuân Huy. *Sáng Tạo Trong Thuật Toán & Lập Trình. Tập 1*. Tái bản lần 10. Nhà Xuất Bản Thông Tin & Truyền Thông, 2024, p. 371.

[Knu98] Donald Ervin Knuth. *The Art of Computer Programming. Volume 3: Sorting and Searching*. 2nd edition. Addison-Wesley Professional, 1998, pp. xiii+782.

[Laa20] Antti Laaksonen. *Guide to Competitive Programming: Learning & Improving Algorithms Through Contests*. 2nd edition. Undergraduate Topics in Computer Science. Springer, 2020, pp. xv+309.

[Thư+21] Trần Đan Thư, Nguyễn Thanh Phương, Đinh Bá Tiến, and Trần Minh Triết. *Nhập Môn Lập Trình*. Nhà Xuất Bản Khoa Học & Kỹ Thuật, 2021, p. 427.

[Tru23] Vương Thành Trung. *Tuyển Tập Dề Thi Học Sinh Giỏi Cấp Tỉnh Trung Học Phổ Thông Tin Học*. Tài liệu lưu hành nội bộ, 2023, p. 235.

[Vie21] Học Viện VietSTEM. *Sách Luyện Thi Hội Thi Tin Học Trẻ với Python Bảng B: Thi Kỹ Năng Lập Trình Cấp Trung Học Cơ Sở*. Nhà Xuất Bản Đại Học Quốc Gia Hà Nội, 2021, p. 190.