ĐẶNG TUẤN THÀNH
(Sưu tầm và biên soạn)

21 GIỜ HỌC LẬP TRÌNH

Quyển 1

Tháng 01 năm 2021

LỜI NÓI ĐẦU

Nhằm cung cấp tài liệu, trang bị kĩ năng tự học, tự nghiên cứu cho học sinh có hứng thú với công nghệ thông tin, hình thành và phát triển kỹ năng cho người học; Xây dựng hệ thống bài tập, đề kiểm tra củng cố kiến thức bộ môn theo chương trình Sách giáo khoa; Bổ sung kiến thức nâng cao, bồi dưỡng học sinh giỏi môn Tin học THCS và THPT, tác giả biên soạn cuốn tài liệu: "21 giờ học lập trình".

Code chương trình trong "**21 giờ học lập trình**" được viết bởi ngôn ngữ lập trình Pascal. Nội dung gồm **56** bài tập của các chuyên đề được biên soạn dạng bài kiểm tra, đề thi tương ứng với 21 giờ tự học. Hệ thống kiến thức được trình bày từ dễ đến khó, tạo điều kiện thuận lợi cho việc tự học, tự nghiên cứu. Với mỗi đơn vị kiến thức, có các bài tập, hướng dẫn thuật toán và code chương trình tham khảo giúp người học nhanh chóng làm quen định dạng, quy cách làm bài thi; Người học được hướng dẫn thuật toán có mức độ hoàn thiện bài khác nhau, phát triển kỹ năng "làm mịn chương trình".

Trong thời gian tới, tác giả viết tiếp tài liệu mới "21 giờ học lập trình – Quyển 2" với các chuyên đề mới, hay hơn nữa. Rất mong các bạn đón đọc.

Tài liệu này được biên soạn rất tỉ mỉ nhưng không tránh được những thiếu sót, tác giả rất mong được sự đóng góp nội dung, phản hồi của bạn đọc vào mail: dtthanh.c3ntt@yenbai.edu.vn.

Tác giả

MỤC LỤC

PHẦN I. ĐỀ BÀI

Ngày 1/7

NỘI DUNG CẦN ĐẠT

- Làm quen với phần mềm Free Pascal, phần mềm chấm bài Themis.
- Làm việc với kiểu dữ liệu tệp. Biết quy cách đặt tên file chương trình, file dữ liệu vào/ra.
- Cách tách các chữ số của một số nguyên.
- Cách tìm ước nguyên dương của một số và các bài toán về ước với thuật toán có độ phức tạp $O(\sqrt{N})$.
- Tổ chức hàm tính a^x mod L chi phí tốt nhất và tránh tràn bộ nhớ.
- Kỹ thuật chặt nhị phân.
- Tổ chức hàm kiểm tra tính nguyên tố của N với chi phí $O(\sqrt{N})$. Cài đặt được giải thuật sàng nguyên tố Eratosthenes.
- Tìm hiểu về số Fibonacci.

Hãy sử dụng chương trình Free Pascal bản mới nhất (freepascal.org)

Bài 1. Cho a, b nguyên dương. Hãy tính diện tích hình chữ nhật có cạnh a, b.

Ví dụ: a = 2, b = 3; → Kết quả: 6

Dữ liệu: Vào từ tệp 'RECT.INP' gồm số nguyên dương a, b (a, b ≤ 10^9).

Kết quả: Ghi ra tệp 'RECT.OUT' diện tích hình chữ nhật tính được.

RECT.INP	RECT.OUT
2 3	6

Bài 2. Nhập số nguyên dương a là số có hai chữ số. Tính tổng các chữ số của a.

Dữ liệu: Vào từ tệp 'PERM.INP' gồm số nguyên dương a là số có hai chữ số.

Kết quả: Ghi ra tệp 'PERM.OUT' tổng các chữ số của a.

PERM.INP	PERM.OUT
23	5

Chuyên đề 1. Số học (làm 5 phút/bài)

1. NCOUNT.*

Tính tổng các chữ số của số nguyên dương N (N ≤ 10^{18}).

Ví dụ: N = 214 → Kết quả: 7.

2. DCOUNT.*

Jame viết các số nguyên dương liên tiếp từ 1 tới N lên bảng. Hãy cho biết Jame đã viết được bao nhiêu chữ số. Biết N ≤ 10^6.

Ví dụ: N = 10 → Kết quả: 11.

3. XCHANGE.*

Jame cần rút X đồng tiền từ cây ATM. Biết hiện tại cây ATM chỉ còn các tờ tiền mệnh giá 50, 20 và 10. Cho biết cây ATM có đáp ứng được lệnh rút tiền của Jame không? Nếu có đáp ứng được, ghi ra số lượng tờ tiền ít nhất Jame nhận được, ngược lại ghi ra -1.

Biết X nguyên dương, $X \leq 10^9$.

Ví dụ: X = 130 → Kết quả: 4;

X = 131 → Kết quả: -1.

4. XMAX.*

Jame viết lên bảng số nguyên dương N. Hãy cho biết giá trị của chữ số lớn nhất trong N. Biết $N \leq 10^{18}$.

Ví dụ: N = 12345678910 → Kết quả: 9.

5. SDIV.*

Cho số nguyên dương N ($N \leq 10^9$). Hãy tính tổng các ước nguyên dương của N.

Ví dụ: N = 12 → tong = 1 + 2 + 3 + 4 + 6 + 12 = 28.

6. AMUX.*

Cho số nguyên dương a và x. Hãy tính $P = a^x$. Vì P có thể rất lớn, nên ta chỉ cần lưu kết quả chia dư cho 10^9+7. Biết a, $x \leq 100$.

Ví dụ: a = 100, x = 2 → Kết quả: 10000.

7. SQUA.*

Cho số nguyên X ($|X| < 10^{18}$). Hãy tính căn bậc 3 của X.

Kết quả lấy chính xác đến 0.001. Ví dụ: X = 27 → Kết quả: 3.000

8. CKPRIME.*

Số nguyên tố là số chỉ có hai ước, 1 và chính nó. Số 1 không là số nguyên tố.

Cho số nguyên dương N ($N \leq 10^9$). Kiểm tra N có là số nguyên tố, nếu có ghi ra TRUE ngược lại ghi ra FALSE.

Ví dụ: N = 10 → FALSE; N = 11 → TRUE.

9. FPRIME.*

Dãy số Fibonacci là dãy số có tính chất:

$F_1 = 1$, $F_2 = 1$, F[N] = F[N-1] + F[N-2] $(\forall N \geq 3)$

Như vậy, các số Fibonacci đầu tiên là: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…

Cho số nguyên dương P. Hãy đếm số lượng các số Fibonacci nhỏ hơn hoặc bằng P và là số nguyên tố.

Ví dụ: P = 10 → Kết quả: 3

10. ERATOS.*

Cho số nguyên dương N ($N \leq 10^7$). Hãy liệt kê các số nguyên tố nhỏ hơn hoặc bằng N. Mỗi số cách nhau một dấu cách.

Ví dụ: N = 10 → Kết quả: 2 3 5 7.

Care and diligence bring luck!

Ngày 2/7

NỘI DUNG CẦN ĐẠT

- Cài đặt được cách phân tích số N thành thừa số nguyên tố với chi phí O($\sqrt{N}$).
- Vận dụng được hàm kiểm tra một số là số nguyên tố vào các bài tập mở rộng.
- Nắm được kĩ thuật duyệt trên mảng một chiều.
- Cài đặt và hiểu được thuật toán sắp xếp nhanh.
- Hiểu và vận dụng được kĩ thuật tính tổng tích lũy.
- Cài đặt chương trình chuẩn tìm giá trị lớn nhất, nhỏ nhất và code được bài tập vận dụng.
- Nắm được kĩ thuật sắp xếp, nén mảng, tìm giá trị lớn nhì, nhỏ nhì.

Bài 1. TSNT.*

Cho số nguyên dương N. Phân tích N thành thừa số nguyên tố.

Ví dụ: N = 28 = 2 * 2 * 7

Dữ liệu: Vào từ tệp 'TSNT.INP' gồm số nguyên dương N ($1 < N \leq 10^9$).

Kết quả: Ghi ra tệp 'TSNT.OUT' các thừa số nguyên tố của N, mỗi số cách nhau một dấu cách.

TSNT.INP	TSNT.OUT
28	2 2 7

Bài 2. NFIND.*

Cho số nguyên dương M ($M \leq 10^9$). Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M. Nếu không tồn tại K, đưa ra -1.

Dữ liệu: Vào từ tệp 'NFIND.INP' gồm số nguyên dương M ($M \leq 10^6$).

Kết quả: Ghi ra tệp 'NFIND.OUT' K nhỏ nhất tìm được hoặc ghi ra -1 nếu không tồn tại K.

NFIND.INP	NFIND.OUT
10	25
13	-1

Bài 3. NUMBERC.*

Cho số nguyên dương N. Người ta có thể cắt lần lượt các chữ số của N từ phải sang trái để thu được số N mới.

Hãy tìm số N mới là số nguyên tố lớn nhất. Biết phải cắt theo yêu cầu ít nhất 1 số. Nếu không có ghi ra -1. Ví dụ: N = 1124 → N = 11.

Dữ liệu: Vào từ tệp 'NUMBERC.INP' gồm số nguyên dương N ($10 \leq N \leq 10^{12}$).

Kết quả: Ghi ra tệp 'NUMBERC.OUT' số nguyên tố lớn nhất tìm được. Nếu không có ghi ra -1.

NUMBERC.INP	NUMBERC.OUT
1124	11
468	-1

Bài 4. ARRDEL.*

Xóa các phần tử giống nhau đứng cạnh nhau trong dãy số, với những số này chỉ giữ lại đại diện một số. Đưa ra dãy sau khi xóa, giữ nguyên thứ tự đầu vào.

Ví dụ: N = 9; dãy 1 3 3 4 2 2 2 7 3 → Kết quả: 1 3 4 2 7 3

Dữ liệu: Vào từ tệp 'ARRDEL.INP' gồm:

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên $a_1, a_2,\ldots, a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'ARRDEL.OUT' dãy số còn lại sau khi xoá.

Mỗi số cách nhau một dấu cách.

ARRDEL.INP	ARRDEL.OUT
9 1 3 3 4 2 2 2 7 3	1 3 4 2 7 3
4 2 2 2 2	2

Bài 5. QSORT.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Hãy sắp xếp dãy này thành dãy không giảm bằng thuật toán sắp xếp nhanh (Quick Sort). Đưa dãy sau khi sắp xếp ra.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → 1 2 3 3 4 7

Dữ liệu: Vào từ tệp 'QSORT.INP' gồm

- Dòng 1: Ghi số nguyên dương N ($N \le 10^5$).
- Dòng 2: Ghi N số nguyên $a_1, a_2,\ldots,a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'QSORT.OUT' dãy số sau khi sắp xếp ra. Mỗi số cách nhau một dấu cách.

QSORT.INP	QSORT.OUT
6 1 3 4 2 7 3	1 2 3 3 4 7

Bài 6. ILUCKY.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Tìm các vị trí i (nếu có) chia dãy số thành 2 phần có tổng bằng nhau. Nếu không có thì ghi ra -1.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → i = 4 (Giải thích: Vị trí thứ 4 chia dãy thành hai phần có tổng 1+3+4+2=7+3).

Dữ liệu: Vào từ tệp 'ILUCKY.INP' gồm

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên $a_1, a_2,\ldots,a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'ILUCKY.OUT' các vị trí đẹp tìm được, mỗi số cách nhau một dấu cách. Nếu không có vị trí đẹp thì ghi ra -1.

ILUCKY.INP	ILUCKY.OUT
6 1 3 4 2 7 3	4
4 1 2 3 7	-1

Bài 7. PMIN.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Đưa ra các vị trí phần tử đạt giá trị nhỏ nhất.

Ví dụ: N = 6; dãy 2 1 4 2 1 4 → 2 và 5.

Dữ liệu: Vào từ tệp 'PMIN.INP' gồm:

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên $a_1, a_2,\ldots,a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'PMIN.OUT' đưa ra các vị trí phần tử đạt giá trị nhỏ nhất. Mỗi số cách nhau một dấu cách.

PMIN.INP	PMIN.OUT
6 2 1 4 2 1 4	2 5

Bài 8. PSECOND.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Tìm giá trị nhỏ nhì (nếu có) của dãy số và đưa ra các vị trí đạt giá trị đó. Nếu không có thì ghi ra -1.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → min2 = 2; vitri = 4

Dữ liệu: Vào từ tệp 'PSECOND.INP' gồm

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên dương $a_1, a_2,\ldots, a_N$ ($a_i \le 10^9$).

Kết quả: Ghi ra tệp 'PSECOND.OUT' giá trị nhỏ nhì của dãy số và đưa ra các vị trí đạt giá trị đó. Mỗi số cách nhau một dấu cách. Nếu không có thì ghi ra -1.

PSECOND.INP	PSECOND.OUT
6 1 3 4 2 7 3	2 5
4 2 2 2 2	-1

Subtask1: $N \le 10^5$

Subtask2: $N \le 10^6$, $a_i \le 10^6$

Subtask3: $N \le 10^6$, $a_i \le 10^9$

Little by little does the trick!

Ngày 3/7

NỘI DUNG CẦN ĐẠT

- Nắm được giải thuật Ơ'clit tìm ước chung lớn nhất.
- Cài đặt được kĩ thuật con trỏ hai đầu.
- Vận dụng được kĩ thuật sắp xếp mảng.
- Nắm được kĩ thuật đếm phân phối và vận dụng được vào các bài tập mở rộng.
- Cài đặt được thuật toán tìm kiếm nhị phân.
- Cài đặt lớp bài toán tìm dãy con liên tiếp thỏa mãn điều kiện.
- Làm việc với mảng hai chiều.
- Nắm được kĩ thuật cài đặt lớp bài toán trên mảng hai chiều: tìm vị trị hoặc số lượng các phần tử thỏa mãn điều kiện.
- Kĩ thuật tính tổng tích lũy trên mảng hai chiều và vận dụng vào các bài tập về hình vuông con kích thước K, hình chữ nhật con thỏa mãn điều kiện.

Bài 1. GCD.*

Cho a, b, c nguyên dương. Hãy tìm ước chung lớn nhất của a, b và c.

Ví dụ: a = 4, b = 6; c = 8 → Kết quả: 2

Dữ liệu: Vào từ tệp 'GCD.INP' gồm số nguyên dương a, b, c (a, b, c ≤ 10^9).

Kết quả: Ghi ra tệp 'GCD.OUT' ước chung lớn nhất của a, b và c.

GCD.INP	GCD.OUT
4 6 8	2

Bài 2. ARRADD2.*

Cho dãy a (N phần tử) và dãy b (M phần tử) là dãy không giảm. Tạo dãy c có M+N phần tử cũng là dãy không giảm. Đưa dãy C ra.

Ví dụ: N = 3; dãy a: 1 3 4; M = 3; dãy b: 2 3 7; → dãy C: 1 2 3 3 4 7

Dữ liệu: Vào từ tệp 'ARRADD2.INP' gồm:

- Dòng 1: Ghi số nguyên dương N và M (M, N ≤ 10^5).
- Dòng 2: Ghi N số nguyên $a_1, a_2, \ldots, a_N$ ($|a_i| \leq 10^9$).
- Dòng 3: Ghi M số nguyên $b_1, b_2, \ldots, b_M$ ($|b_i| \leq 10^9$).

Kết quả: Ghi ra tệp 'ARRADD2.OUT' dãy c tìm được. Mỗi số cách nhau một dấu cách.

ARRADD2.INP	ARRADD2.OUT
5 6 1 2 2 5 7 3 3 7 8 9 9	1 2 2 3 3 5 7 7 8 9 9

Bài 3. ARRCOUNT.*

Cho N và dãy số nguyên dương $a_1, a_2, \ldots, a_N$. Đếm số lần xuất hiện của mỗi phần tử trong dãy a.

Dữ liệu: Vào từ tệp 'ARRCOUNT.INP' gồm:

- Dòng 1: Ghi số nguyên dương N là số phần tử của dãy a (N ≤ 10^5).

- Dòng 2: Ghi N số nguyên dương a_1, a_2, …, a_N ($a_i \le 10^9$).

Kết quả: Ghi ra tệp 'ARRCOUNT.OUT' dạng số a[i] xuất hiện p lần, mỗi bộ số trên một dòng. Mỗi số cách nhau một dấu cách. Các số a[i] là dãy tăng ngặt.

ARRCOUNT.INP	ARRCOUNT.OUT	Giải thích
8 1 3 3 4 2 2 2 7	1 1 2 3 3 2 4 1 7 1	Số 1 xuất hiện 1 lần Số 2 xuất hiện 3 lần ……

Subtask1: a[i] ≤ 10^6. Subtask2: a[i] ≤ 10^9.

Bài 4. BINS.* (Binary Search – Tìm kiếm nhị phân)

Cho N, X và dãy số nguyên a_1, a_2, …, a_N là dãy đơn điệu tăng. Hãy tìm vị trí của X trong dãy a. Nếu không có X trong dãy a thì ghi ra -1.

Dữ liệu: Vào từ tệp 'BINS.INP' gồm:

- Dòng 1: Ghi số nguyên dương N, X ($N \le 10^6$, $|X| \le 10^9$).
- Dòng 2: Ghi N số nguyên a_1, a_2, …, a_N ($|a_i| \le 10^9$).

Kết quả:

Ghi ra tệp 'BINS.OUT' vị trí của X trong dãy a. Nếu không có X trong dãy a, ghi ra -1.

BINS.INP	BINS.OUT
5 6 2 3 5 6 9	4
5 4 2 3 5 6 9	-1

Bài 5. UPSEQ.*

Cho N và dãy số nguyên a_1, a_2, …, a_N. Tìm dãy con liên tiếp không giảm dài nhất của dãy. Đưa ra độ dài lớn nhất tìm được.

VD: N = 6; dãy 1 3 4 2 7 3 → 3

Giải thích: dãy không giảm dài nhất là dãy 1 3 4.

Dữ liệu: Vào từ tệp 'UPSEQ.INP' gồm:

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên a_1, a_2, …, a_N ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'UPSEQ.OUT' độ dài dãy con liên tiếp không giảm dài nhất.

UPSEQ.INP	UPSEQ.OUT
6 1 3 4 2 7 3	3
4 2 2 2 2	4

Bài 6. ARR2D.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy tính và đưa ra tổng của từng hàng, tổng của từng cột.

Dữ liệu: Vào từ tệp 'ARR2D.INP' gồm:

- Dòng 1: Ghi số nguyên dương M và N là kích thước mảng a (M, N ≤ 3000).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \le 10^5$).

Kết quả: Ghi ra tệp 'ARR2D.OUT' dạng:

- Dòng 1: Ghi M số nguyên là tổng của hàng 1, hàng 2,…. hàng M.
- Dòng 2: Ghi N số nguyên là tổng của cột 1, cột 2, …, cột N.

Mỗi số cách nhau một dấu cách.

ARR2D.INP	ARR2D.OUT
2 3 1 2 -2 -1 2 1	1 2 0 4 -1

Bài 7. ARR2DPRI.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy đưa ra vị trí các phần tử là số nguyên tố.

Dữ liệu: Vào từ tệp 'ARR2DPRI.INP' gồm:

- Dòng 1: Ghi số nguyên dương M và N là kích thước mảng a (M, N ≤ 300).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên dương a[i,j] ($a_{i,j} \le 10^4$).

Kết quả: Ghi ra tệp 'ARR2DPRI.OUT' mỗi vị trí thoả mãn ghi trên một dòng. Nếu không có vị trí thoả mãn thì ghi ra -1.

Mỗi số cách nhau một dấu cách.

ARR2DPRI.INP	ARR2DPRI.OUT	Giải thích
2 3 1 4 5 1 2 1	1 3 2 2	Tại ô (1,3) có số 5 là số nguyên tố.
2 3 1 4 4 1 1 1	-1	

Bài 8. ARR2DSQU.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy tìm hình vuông có cạnh bằng K có tổng lớn nhất. Đưa ra các góc trái trên của các hình vuông và tổng lớn nhất tìm được.

Dữ liệu: Vào từ tệp 'ARR2DSQU.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N và K (M, N, K ≤ 300, K ≤ min(M,N)).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \le 10^4$).

Kết quả: Ghi ra tệp 'ARR2DSQU.OUT' dạng:

- Dòng 1: Ghi số tổng lớn nhất tìm được.
- Các dòng tiếp theo ghi toạ độ góc trái trên của hình vuông cạnh K có tổng lớn nhất.

Mỗi số cách nhau một dấu cách.

ARR2DSQU.INP	ARR2DSQU.OUT	Giải thích
2 3 2 1 -4 2 -1 **2** 1	1 1 2	Tổng lớn nhất: -4+2+2+1=1 Toạ độ góc trái trên của hình vuông cạnh 2 là: (1,2).

Success is achieved and maintained by those who try and keep trying.

W. Clement Stone

Ngày 4/7

NỘI DUNG CẦN ĐẠT

- Vận dụng kĩ thuật tính tổng tích lũy để giải lớp bài toán tìm hình vuông con, hình chữ nhật con thỏa mãn điều kiện.
- Có kĩ năng xử lí với các đối tượng có kích thước dữ liệu lớn.
- Nắm được kĩ thuật làm mịn chương trình.

Bài 1. ARR2HVS1**.***

Cho mảng a là mảng hai chiều kích thước M x N. ***Hãy tìm tất cả các hình vuông có tổng bằng S***.

Dữ liệu: Vào từ tệp 'ARR2HVS1.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N và S (M, N ≤ 100, S ≤ 10^9).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \leq 10^4$).

Kết quả: Ghi ra tệp 'ARR2HVS1.OUT' gồm nhiều dòng, mỗi dòng ghi: toạ độ góc trái trên và độ dài cạnh của hình vuông có tổng bằng S. Nếu không có hình vuông nào thoả mãn, ghi ra -1.

Mỗi số cách nhau một dấu cách.

ARR2HVS1.INP	ARR2HVS1.OUT	Giải thích
2 3 2 1 4 2 -1 -2 1	1 1 2 1 3 1	Hình vuông có góc trái trên (1,1) và cạnh 2 có tổng = 1+4-1-2 = 2 = S …

Bài 2. ARR2HVS2.*

Cho mảng a là mảng hai chiều kích thước M x N. **Hãy tìm hình vuông lớn nhất có tổng bằng S**.

Dữ liệu: Vào từ tệp 'ARR2HVS2.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N và S (M, N ≤ 100, S ≤ 10^9).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \leq 10^4$).

Kết quả: Ghi ra tệp 'ARR2HVS2.OUT' kích thước của hình vuông tìm được.

Nếu không có hình vuông nào thoả mãn, ghi ra -1.

Mỗi số cách nhau một dấu cách.

ARR2HVS2.INP	ARR2HVS2.OUT	Giải thích
2 3 2 1 4 2 -1 -2 1	2	Hình vuông có góc trái trên (1,1) và cạnh 2 có tổng = 1+4-1-2 = 2 = S

Bài 3. ARR2HCN1.*

Cho mảng a là mảng hai chiều kích thước M x N. ***Hãy tìm tất cả các hình chữ nhật có tổng bằng S***.

Dữ liệu: Vào từ tệp 'ARR2HCN1.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N và S (M, N ≤ 50, S ≤ 10).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \leq 10^4$).

Kết quả: Ghi ra tệp 'ARR2HCN1.OUT' gồm nhiều dòng, mỗi dòng ghi: toạ độ góc trái trên và góc phải dưới của hình chữ nhật có tổng bằng S.

Nếu không có hình chữ nhật nào thoả mãn, ghi ra -1.

Mỗi số cách nhau một dấu cách.

ARR2HCN1.INP	ARR2HCN1.OUT	Giải thích
2 3 2 1 4 2 -1 -2 1	1 1 2 2 1 2 2 2 1 3 1 3	Hình chữ nhật có góc trái trên (1,1) và góc phải dưới (2,2) có tổng = 1+4-1-2 = 2 = S …

Bài 4. ARR2HCN2.*

Cho mảng a là mảng hai chiều kích thước M x N. **Hãy đếm số lượng hình chữ nhật có tổng chia hết cho K.**

Dữ liệu: Vào từ tệp 'ARR2HCN2.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N và K (M, N ≤ 50, K ≤ 10^9).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên dương a[i,j] ($|a_{i,j}| \leq 10^4$).

Kết quả: Ghi ra tệp 'ARR2HCN2.OUT' số lượng hình chữ nhật có tổng chia hết cho K.

Nếu không có hình chữ nhật nào thoả mãn, ghi ra -1.

ARR2HCN2.INP	ARR2HCN2.OUT
2 3 2 1 4 2 1 2 1	8

Khổ luyện thành tài, miệt mài thành giỏi!

Ngày 5/7

NỘI DUNG CẦN ĐẠT

- Vận dụng kĩ năng tính tổng tích lũy trên mảng hai chiều để làm các bài tập mới.
- Hình thành tư tưởng quy hoạch động như: Gọi H[i] là tổng của hàng thứ i, c[j] là tổng của cột thứ j.
- Làm việc thành thạo với kiểu dữ liệu xâu. Như nén xâu, giải nén xâu, tác số trong xâu…

Bài 1. ARR2REC1.*

Cho ma trận kích thước MxN mà các phần tử có giá trị bằng 0 hoặc 1. Hãy tìm hình chữ nhật chứa toàn số 1 có diện tích lớn nhất. Đưa ra diện tích lớn nhất đó. Dữ liệu đảm bảo có nghiệm.

Dữ liệu: Vào từ tệp '**ARR2REC1**.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N (M, N ≤ 100).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] $a_{i,j}$ = 0 hoặc $a_{i,j}$ = 1.

Kết quả: Ghi ra tệp '**ARR2REC1**.OUT' diện tích lớn nhất tìm được.

Mỗi số cách nhau một dấu cách.

ARR2REC1.INP	**ARR2REC1.OUT**	**Giải thích**
3 4 0 0 0 0 1 1 1 0 1 1 1 0	6	Hình chữ nhật chứa toàn số 1 có kích thước 2x3.

Bài 2. ARR2REC2.*

Cho ma trận MxN. Quân xe trong cờ vua hoặc cờ tướng có đặc điểm di chuyển và ăn quân theo hình dấu (+).

Người ta đặt quân xe ở ô (i, j) thì số điểm thu được bằng tổng giá trị các phần tử ở hàng i và tổng giá trị các phần tử ở cột j và a[i,j].

Hãy tìm vị trí đặt quân xe để đạt tổng điểm lớn nhất. Đưa ra tổng điểm lớn nhất đó.

Dữ liệu: Vào từ tệp '**ARR2REC2.INP**' gồm:

- Dòng 1: Ghi số nguyên dương M, N (M, N ≤ 100).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \le 10^4$).

Kết quả: Ghi ra tệp '**ARR2REC2.OUT**' tổng điểm lớn nhất tìm được.

Mỗi số cách nhau một dấu cách.

ARR2REC2.INP	**ARR2REC2.OUT**	**Giải thích**
3 4 0 0 0 0 1 -1 1 0 **1** 1 1 0	4	Đặt quân xe ở ô (3,1) có tổng bằng 4 là lớn nhất.

Bài 3. ARR2REC3*. Quân mã

Cho ma trận kích thước M, N. Tìm vị trí đặt quân Mã để kiểm soát được các ô có tổng lớn nhất (bao gồm cả ô con Mã đang đứng).

Dữ liệu: Vào từ tệp '**ARR2REC3**.INP' gồm:

- Dòng 1: Ghi số nguyên dương M, N (M, N ≤ 100).
- M dòng tiếp theo, dòng thứ i ghi N số nguyên a[i,j] ($|a_{i,j}| \le 10^4$).
  Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp '**ARR2REC3**.OUT' tổng điểm lớn nhất tìm được.

ARR2REC3.INP	ARR2REC3.OUT	Giải thích
3 4 0 **0** 0 0 1 -1 1 0 **1** 1 **1** 0	2	Đặt quân mã ở ô (1,2) có tổng bằng 2 là lớn nhất.

Bài 4. STR1NUM.*

Nhập xâu S. Tách các số trong xâu ra.

Dữ liệu: Vào từ tệp '**STR1NUM**.INP' xâu S có độ dài nhỏ hơn 250 kí tự chỉ bao gồm chữ số, chữ cái thường và chữ cái in hoa.

Kết quả: Ghi ra tệp '**STR1NUM**.OUT' dãy số trong xâu ra. Mỗi số cách nhau một dấu cách.

STR1NUM.INP	STR1NUM.OUT
Abc234abd2dfg56	234 2 56

Bài 5. STR2NUM.*

Nhập xâu S. Tìm số lớn nhất trong xâu.

Dữ liệu: Vào từ tệp '**STR2NUM**.INP' xâu S có độ dài nhỏ hơn 250 kí tự chỉ bao gồm chữ số, chữ cái thường và chữ cái in hoa.

Kết quả: Ghi ra tệp '**STR2NUM**.OUT' số lớn nhất trong xâu.

STR2NUM.INP	STR2NUM.OUT
Abc234abd2dfg56	234

Bài 6. STR2NUM .* Nén xâu

Cho xâu S chỉ gồm các chữ cái từ 'a' đến 'z'. Hãy nén xâu S. Đưa xâu nén ra.

Ví dụ: S= 'aaabbccccd' → Snen ='a3b3c4d1'.

Dữ liệu: Vào từ tệp '**STR2NUM**.INP' xâu S có độ dài nhỏ hơn 1000 kí tự.

Kết quả: Ghi ra tệp '**STR2NUM**.OUT' số lớn nhất trong xâu.

STR2NUM.INP	STR2NUM.OUT
aaabbccccd	a3b3c4d1

Bài 7. STR3NUM.* Giải nén xâu.

Nhập xâu Snen. Hãy giải nén xâu.

Ví dụ: Snen ='a3b3c4d1' → S = 'aaabbccccd'

Dữ liệu: Vào từ tệp '**STR3NUM**.INP' xâu S có độ dài nhỏ hơn 250 kí tự gồm chữ cái và chữ số.

Kết quả: Ghi ra tệp '**STR3NUM**.OUT' số lớn nhất trong xâu.

STR3NUM.INP	STR3NUM.OUT
a3b3c4d1	aaabbccccd

Ngày 6/7

NỘI DUNG CẦN ĐẠT

- Nắm được khái niệm: cấu hình, hoán vị, dãy nhị phân, chỉnh hợp không lặp, tổ hợp chập k, đệ quy, quay lui, nhánh cận…
- Biết được cấu hình và thứ tự cấu hình.
- Vận dụng phương pháp sinh làm các bài tập có tính chất: chọn/không chọn.
- Có kĩ thuật tổ chức chương trình thành các chương trình con.
- Kỹ thuật tạo bộ test và sử dụng ứng dụng Themis để chấm bài.

---

1. DQBIT01.*

Cho N. Hãy liệt kê các dãy nhị phân có độ dài N.

Dữ liệu: Vào từ tệp 'DQBIT01.INP' gồm số nguyên dương N ($N \leq 20$).

Kết quả: Ghi ra tệp 'DQBIT01.OUT' gồm nhiều dòng, mỗi dòng ghi một dãy nhị phân tìm được. Các dãy nhị phân được sắp xếp theo thứ tự từ điển.

Mỗi số cách nhau một dấu cách.

DQBIT01.INP	DQBIT01.OUT
3	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1

2. DQBIT02.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có số lượng số 0 bằng số lượng số 1. Nếu không có ghi ra -1.

Dữ liệu: Vào từ tệp 'DQBIT02.INP' gồm số nguyên dương N ($N \leq 20$).

Kết quả: Ghi ra tệp 'DQBIT02.OUT' gồm nhiều dòng, mỗi dòng ghi một dãy nhị phân tìm được. Các dãy nhị phân được sắp xếp theo thứ tự từ điển.

Mỗi số cách nhau một dấu cách.

DQBIT02.INP	DQBIT02.OUT
2	0 1 1 0

3. DQBIT03.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N không có 3 số 1 đứng cạnh nhau. Nếu không có ghi ra -1.

Dữ liệu: Vào từ tệp 'DQBIT03.INP' gồm số nguyên dương N ($N \leq 20$).

Kết quả: Ghi ra tệp 'DQBIT03.OUT' gồm nhiều dòng, mỗi dòng ghi một dãy nhị phân tìm được. Các dãy nhị phân được sắp xếp theo thứ tự từ điển, mỗi cấu hình trên một dòng.

Mỗi số cách nhau một dấu cách.

DQBIT03.INP	DQBIT03.OUT
3	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0

4. DQBIT04.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có đúng K số 1. Nếu không có ghi ra -1.

Dữ liệu: Vào từ tệp 'DQBIT04.INP' gồm số nguyên dương N và K ($K \leq N \leq 20$).

Kết quả: Ghi ra tệp 'DQBIT04.OUT' gồm nhiều dòng, mỗi dòng ghi một dãy nhị phân tìm được. Các dãy nhị phân được sắp xếp theo thứ tự từ điển.

Mỗi số cách nhau một dấu cách.

DQBIT04.INP	DQBIT04OUT
3 2	0 1 1 1 0 1 1 1 0

5. DQBIT05.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có không quá K số 1.

Dữ liệu: Vào từ tệp 'DQBIT05.INP' gồm số nguyên dương N và K ($K \leq N \leq 20$).

Kết quả: Ghi ra tệp 'DQBIT05.OUT' gồm nhiều dòng, mỗi dòng ghi một dãy nhị phân tìm được. Các dãy nhị phân được sắp xếp theo thứ tự từ điển. Nếu không có ghi ra -1.

Mỗi số cách nhau một dấu cách.

DQBIT05.INP	DQBIT05OUT
3 2	0 0 0 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0

Ngày 7/7

NỘI DUNG CẦN ĐẠT

- Kỹ thuật tổ chức chương trình thành các hàm và thủ tục.
- Từ bài toán thực tế, tìm công thức Toán học và cài đặt chương trình giải bài toán đó.
- Biết được điều kiện để các điểm thỏa mãn: nằm trong hay ngoài đường tròn, trong ngoài tam giác….
- Biết điều kiện vị trí tương đối của các đường tròn…
- Vận dụng các thuật toán đã học về ước số để giải quyết các bài toán mới với chi phí tốt nhất $O(\sqrt{N})$.
- Làm quen với lớp bài: Cho tập hợp, tìm vị trí của phần tử X. Cho vị trí, tìm giá trị của X.
- Vận dụng hàm a^x để giải quyết bài tập mới. Lưu ý tránh để tràn dữ liệu khi input chứa số có giá trị lớn.
- Rèn luyện kĩ năng xử lí xâu.

A. TRIANGLE.*

Cho toạ độ các điểm A(x1,y1), B(x2,y2), C(x3,y3). Kiểm tra A, B, C có là ba đỉnh của một tam giác? Nếu có hãy diện tích tam giác ABC, ngược lại ghi ra -1.

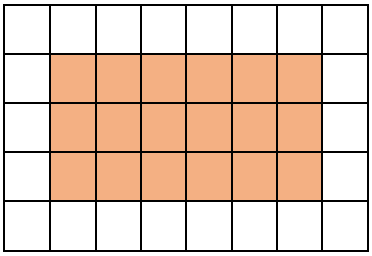
Dữ liệu: Vào từ tệp 'TRIANGLE.INP' ghi số nguyên x1, y1, x2, y2, x3, y3. Các số có giá trị không quá 10^9. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp 'TRIANGLE.OUT' diện tích tam giác ABC nếu ABC là 3 đỉnh của một tam giác (lấy chính xác đến 3 chữ số thập phân), ngược lại ghi ra -1.

TRIANGLE.INP	TRIANGLE.OUT
0 0 0 1 1 0	0.500
0 0 0 1 0 2	-1

A. RECTAB1.*

Cho hình chữ nhật có kích thước a x b, được chia thành lưới ô vuông đơn vị. Gọi X là số ô đường viền, Y là số ô trong lõi. Hãy tìm X, Y.



Dữ liệu: Vào từ tệp 'RECTAB1.INP' ghi số nguyên dương a và b. Các số có giá trị không quá 10^9. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp 'RECTAB1.OUT' giá trị X, Y tìm được.

RECTAB1.INP	RECTAB1.OUT
5 8	22 18

B. CIRCLE.*

Cho điểm M (x1, y1) và đường tròn tâm I(x0, y0) bán kính R. Hãy xét vị trí tương đối của điểm M với đường tròn. Nếu M nằm trong hoặc trên đường tròn ghi ra 1, ngược lại ghi ra 0.

Dữ liệu: Vào từ tệp 'CIRCLE.INP' ghi số nguyên dương x1, y1, x0, y0 và R. Các số có giá trị không quá 10^9. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp 'CIRCLE.OUT' giá trị 1 nếu M nằm trong hoặc trên đường tròn ngược lại ghi ra 0.

CIRCLE.INP	CIRCLE.OUT
1 0 0 0 2	1
3 0 0 0 2	0

C. RECTM2.*

Cho điểm M(x1, y1) và hình chữ nhật có góc trái trên (x2, y2) và góc phải dưới (x3, y3). Kiểm tra M nằm trong hay ngoài hình chữ nhật. Nếu nằm trong và trên cạnh thì ghi ra 1, ngược lại, M nằm ngoài hình chữ nhật ghi ra 0.

Dữ liệu: Vào từ tệp 'RECTM2.INP' ghi số nguyên x1, y1, x2, y2, x3, y3. Các số có giá trị không quá 10^9. Mỗi số cách nhau một dấu cách.

Kết quả: Ghi ra tệp 'RECTM2.OUT' nếu nằm trong và trên cạnh thì ghi ra 1, ngược lại, M nằm ngoài hình chữ nhật ghi ra 0.

RECTM2.INP	RECTM2.OUT
0 2 3 0 0 3	0
11 0 2 3 0	1

D. PERFECTN.*

Số H được gọi là *số hoàn thiện* nếu tổng tất cả các ước của H gấp đôi H.

Ví dụ: Số 6 là số hoàn thiện, 6 chia hết cho 1, 2, 3, 6 và tổng ước = 12 = 2x6

Cho N, hãy tìm các số hoàn thiện nhỏ hơn hoặc bằng N.

Dữ liệu: Vào từ tệp 'PERFECTN.INP' ghi số nguyên dương N ($N \leq 10^4$).

Kết quả: Ghi ra tệp 'PERFECTN.OUT' các số hoàn thiện tìm được, mỗi số cách nhau một dấu cách. Nếu không có ghi ra -1.

PERFECTN.INP	PERFECTN.OUT
10	6
5	-1

E. FIBPRIME.*

Dãy số Fibonacci có đặc điểm: F[1] = 1, F[2] = 1, F[i] = F[i-1] + F[i-2] với $i \geq 3$.

Cho số nguyên dương M và N. Hãy tìm các số vừa là số Fibonacci, vừa là số nguyên tố nằm có giá trị thuộc đoạn [M, N].

Dữ liệu: Vào từ tệp 'FIBPRIME.INP' ghi số nguyên dương M và N ($1 \leq M \leq N \leq 10^6$).

Kết quả: Ghi ra tệp 'FIBPRIME.OUT' các số Fibonacci nguyên tố tìm được, mỗi số cách nhau một dấu cách. Nếu không có ghi ra -1.

FIBPRIME.INP	FIBPRIME.OUT
1 10	2 3 5
14 15	-1

F. NBEAUTY.*

Số đẹp là số có tổng các chữ số chia hết cho 9. Như vậy, các số 9, 18, 27, 36,... là các số đẹp. Cho số nguyên dương N. Tìm số đẹp thứ N.

Dữ liệu: Vào từ tệp 'NBEAUTY.INP' ghi số nguyên dương N ($N \leq 10^6$).

Kết quả: Ghi ra tệp 'NBEAUTY.OUT' số đẹp thứ N.

NBEAUTY.INP	NBEAUTY.OUT
2	18
3	27

G. NPALIN.*

Số Palidrom là số đối xứng, nghĩa là đọc từ trái sang phải hay từ phải sang trái ta đều được một số. Ví dụ: Các số palidrom 11, 121, 1331, 2222, 1556551,... Các số có một chữ số cũng là số Palidrom.

Cho số nguyên dương N, hãy tìm số Palidrom thứ N.

Dữ liệu: Vào từ tệp 'NPALIN.INP' ghi số nguyên dương N ($N \leq 10^6$).

Kết quả: Ghi ra tệp 'NPALIN.OUT' số đẹp thứ N.

NPALIN.INP	NPALIN.OUT
3	3
10	11

H. LASTNUM.*

Cho số nguyên dương a và N. Hãy tìm chữ số tận cùng của a^N. ().

Dữ liệu: Vào từ tệp 'LASTNUM.INP' ghi số nguyên dương a và N ($a \leq 100$; $N \leq 10^6$).

Kết quả: Ghi ra tệp 'LASTNUM.OUT' chữ số tận cùng của a^N.

LASTNUM.INP	LASTNUM.OUT
3 3	7
10 1	0

PHẦN II. HƯỚNG DẪN THUẬT TOÁN

Ngày 1/7

Nên sử dụng chương trình Free Pascal bản mới nhất (freepascal.org) để hỗ trợ lập trình tốt hơn. Tương lai gần, bạn nên sử dụng C++.

1) RECT.*

Vì a, b ≤ 10^9, nên tích = a*b lớn nhất có thể lên tới 10^{18}. Kiểu dữ liệu *longint* và các kiểu dữ liệu số nguyên khác bé hơn không lưu trữ được. Nên ta phải dùng kiểu nguyên *int64*.

Note:

- *Khi làm bài lưu ý đến giới hạn của các biến đầu vào, biến kết quả và biến trung gian để chọn kiểu dữ liệu hợp lý.*
- *Tên file.PAS, file.INP và file. OUT phải đặt giống nhau theo yêu cầu của đề bài. Khi bị sai tên file, bài làm sẽ bị 0 điểm do chấm bài bằng phần mềm Themis.*
- *Trước khi nộp bài, hãy chạy lại test ví dụ để kiểm tra tên file inp, file out và kết quả ghi ra file out đã đúng định dạng chưa, có bị ghi thừa kết quả không.*

```
program caux_010120;
var
	a, b: int64;
	res: int64;
const
	fi = 'RECT.INP';
	fo = 'RECT.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(a, b);
	res := a*b;
	writeln(res);
	close(input); close(output);
end.
```

2) PERM .*

Nhập số nguyên dương a là số có hai chữ số. Tính tổng các chữ số của a.

Dùng phép toán chia nguyên (div) và chia dư (mod) để tách các chữ số của a.

Hangchuc = a div 10;

Hangdv = a mod 10;

Tong = hangchuc + hangdv;

Note:

- *Dùng div và mod ta có thể tách các chữ số của một số ra, sau đó xử lí theo yêu cầu của đề.*
- *Bài tập luyện tập thêm kĩ năng làm việc với tệp.*

```
program caux_010120;
var
	n, res: longint;
const
	fi = 'PERM.INP';
	fo = 'PERM.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	while n <> 0 do
		begin
			res := res + n mod 10;
			n := n div 10;
		end;
	write(res);
	close(input); close(output);
end.
```


3) NCOUNT.*

Tính tổng các chữ số của số nguyên dương N ($N \leq 10^{18}$).

Ví dụ: N = 214 → KQ: 7.

Thuật toán:

Ta lần lượt tách chữ số ở hàng đơn vị của N bằng phép toán chia dư cho 10 và đẩy nó vào tổng. Phần còn lại của N là N div 10.

Thuật toán dừng khi N = 0. Ta đưa giá trị của tong ra.

```
Tong := 0;
While n > 0 do
Begin
  Tong := tong + (N mod 10);
  N := N div 10;
End;
Write(tong);
```

NOTE:

- *Giá trị của $N \leq 10^{18}$, nên kiểu dữ liệu của N là int64. Nếu chọn kiểu dữ liệu nhỏ hơn, bài làm đúng chỉ đạt dưới 50% điểm của bài.*
- *Dự vào cách tách các chữ số của N như trên, ta sẽ tìm được số đảo của số N bằng cách thay:*
  *Tong := tong + (N mod 10); bằng sodao := sodao*10+(N mod 10);*

```
program caux_010120;
var
	n, res: int64;
const
	fi = 'PERM.INP';
	fo = 'PERM.OUT';

begin
	assign(input,fi); reset(input);
	   assign(output,fo); rewrite(output);
	readln(n);
	while n <> 0 do
		begin
			res := res + n mod 10;
			n := n div 10;
		end;
	writeln(res);
	close(input); close(output);
end.
```

4) DCOUNT.*

Ví dụ: N = 10 → KQ: 11

Giải thích: Jame đã viết các chữ số là: 12345678910, có 11 chữ số.

Thuật toán:

- **Sub1** (cách 1): Cho i chạy từ 1 tới N, đến số thứ i ta đếm số lượng chữ số của i và đẩy vào tổng.

Tong := tong + demso(i); trong đó hàm demso(i) trả về số lượng chữ số của i.

Để đếm số lượng chữ số của số i, ta tổ chức thành hàm demso(u: longint): longint;

- **Sub2**: Kĩ thuật đếm phân phối

Ta sử dụng một mảng c: array[0..9] of byte;

Cho i chạy từ 1 tới N, đến số thứ i ta phantich(i);

Thủ tục phân tích i lần lượt cắt các chữ số ở hàng đơn vị của i, và tăng giá trị mảng đếm phân phối lên.

```
Procedure phantich(u: longint);
Var j: longint;
Begin
  While u > 0 do
  Begin
    P:= u mod 10;
    C[p] := c[p]+1;
 End;
```

Ta tính tổng tất cả các c[i] sẽ là kết quả của bài.

```
Res := 0;
For i:=0 to 9 do
  Res := res + c[i];
Write(res);
```

Note:

- *Các biến tổng, tích, kết quả phải khởi tạo giá trị ban đầu.*
- *Nên tổ chức chương trình thành các chương trình con dạng hàm hoặc thủ tục.*

```
var
      n, i, res: longint;
      st: string;
const
      fi = 'DCOUNT.INP';
      fo = 'DCOUNT.OUT';
begin
      assign(input,fi); reset(input);
         assign(output,fo); rewrite(output);
      readln(n);
      for i:=1 to n do
            begin
                  str(i,st);
                  res := res + length(st);
            end;
      write(res);
      close(input); close(output);

end.
```

5) XCHANGE.*

Thuật toán: Để rút được số lượng tờ tiền ít nhất, máy sẽ trả tiền từ mệnh giá to tới mệnh giá nhỏ.

Ví dụ: X = 130, máy sẽ rút 2 tờ 50, còn dư 30, máy sẽ rút mệnh giá nhỏ hơn là 20, được 1 tờ, còn dư 10, máy sẽ rút nốt tờ 10đ. Đã rút xong.

Ta sử dụng phép toán div, mod để làm bài này.

Xử lí thêm trường hợp không rút hết: Sau khi rút xong tờ mệnh giá 10đ, nếu phần tiền còn lại X > 0, ta kết luận không rút được và ghi ra -1.

```
If x > 0 then
 Write(-1)
Else write(res); // res là tổng số tờ tiền nhận được.
```

NOTE:

Hãy luôn nhớ xử lí trường hợp vô nghiệm để không bị mất trường hợp vô nghiệm.

```
program caux_010120;
var
	n, i, res: longint;

const
	fi = 'XCHANGE.INP';
	fo = 'XCHANGE.OUT';

begin
	assign(input,fi); reset(input);
	   assign(output,fo); rewrite(output);
	readln(n);
	if n mod 50 mod 20 mod 10 <> 0 then write(-1)
	else
		begin
			res := n div 50;
			n := n mod 50;
			res := res + n div 20;
			n := n mod 20;
			res := res + n div 10;
			write(res);
		end;
	close(input); close(output);
end.
```

6) XMAX.*

Jame viết lên bảng số nguyên dương N. Hãy cho biết giá trị của chữ số lớn nhất trong N. Biết $N \leq 10^{18}$.

Ví dụ: N = 12345678910 → Kết quả: 9.

Thuật toán:

Lần lượt tách các chữ số hàng đơn vị của N và cập nhật resmax.

```
Resmax := low(int64);
While n>0 do
Begin
   Resmax := max(resmax, n mod 10);
   N:= N div 10;
End;
Write(resmax);
```

NOTE:

- *Lưu ý giới hạn của $N \leq 10^{18}$.*
- *Khi tìm max thì khởi tạo bằng low(kiểu dữ liệu); khi tìm min thì khởi tạo bằng high(kiểu dữ liệu). Nhớ sử dụng thư viện toán học math.*

```
program caux_010120;
var
	n: string;
	i: integer;
	res: char;
const
	fi = 'XMAX.INP';
	fo = 'XMAX.OUT';
begin
	assign(input,fi); reset(input);
	   assign(output,fo); rewrite(output);
	readln(n);
	res := '0';
	for i:=1 to length(n) do
		begin
			if n[i] = '9'then begin write(9); exit; end
			else if n[i] > res then res := n[i];
		end;
	write(res);
	close(input); close(output);
end.
```

7) SDIV.*

Cho số nguyên dương N ($N \leq 10^9$). Hãy tính tổng các ước nguyên dương của N.

Ví dụ: N = 12 → tong = 1 + 2 + 3 + 4 + 6 + 12 = 28.

Thuật toán:

- **Sub1:** Duyệt i từ 1 tới N, đến số thứ i ta kiểm tra N có chia hết cho i, nếu có ta đẩy vào tổng.

Độ phức tạp thuật toán O(N). Chạy được với $N \leq 10^6$. Làm theo cách này sẽ đạt không quá 50% điểm của bài.

Tuy nhiên, khi N đủ lớn, ta không thể For tới N vì thế ta phải cải tiến chương trình.

- **Sub2:** Nếu N = 1 thì tổng = 1;

Với N > 1, nếu i là ước của N thì $i \in [1, n\ div\ 2]$.

Tất nhiên, nếu N chia hết cho i thì N cũng chia hết cho N div i.

Ví dụ: 10 chia hết cho **2** thì 10 cũng chia hết cho **5** (10 div 2).

Tuy nhiên, 9 chia hết cho 3 và cũng chia hết cho phần còn lại là 3 (9 div 3).

Do đó ta chỉ đẩy ước *N div i* vào tổng khi *N div i* khác *i*.

Độ phức tạp thuật toán O(N/2) ~ 70% điểm của bài.

```
Function tonguoc(u: longint): int64;
Var j: longint;
 Temp: int64=0;
Begin
   If u <= 1 then exit(1);
   For j:=1 to u div 2 do
     If u mod j = 0 then
          Begin
              Inc(temp,j);
             If j<> n div j then inc(temp,n div j);
          End;
   Exit(temp);
End;
```

- **Sub3: Bổ đề: Nếu i là ước của N thì i bé nhất khác 1 (nếu có) sẽ thuộc đoạn [1, $\sqrt{N}$]**

Ta sẽ tổ chức chương trình tương tự Sub2 nhưng thay ***For (j,1,u div 2)*** thành ***for (j,1, trunc(sqrt(u))).***

Độ phức tạp thuật toán O($\sqrt{N}$) ~ 100% điểm của bài.

NOTE:

- *Để tính tổng các ước của N, khi tổ chức thành chương trình con ta đặt tên hàm và biến như sau: Function tonguoc(u: longint):* ***int64****; Tức là dùng u đại diện cho N. Trong chương trình con này dùng biến chạy j.*
- *Từ giờ trở đi, các bài toán nào liên quan đến ước như: đếm ước, tổng ước, kiểm tra tính chất của một ước của N ta đề For (i,1,*[$\sqrt{N}$]*).*
- *Tổng các ước của N có thể vượt kiểu dữ liệu của N.*

```
program caux_010120;
var
      n, i: longint;
      res: int64;
const
      fi = 'SDIV.INP';
      fo = 'SDIV.OUT';
begin
      assign(input,fi); reset(input);
         assign(output,fo); rewrite(output);
      readln(n);
      for i:=1 to trunc(sqrt(n)) do
            begin
                  if n mod i = 0 then
```

```
                    begin
                        res := res + i;
                        if i <> n div i then res := res + (n div i);
                    end;
            end;
        write(res);
        close(input); close(output);
end.
```

8) AMUX.*

Cho số nguyên dương a và x. Hãy tính $P = a^x$.

Thuật toán:

- **Sub1:** Ta for(i,1,x) p = p* a; với p ban đầu bằng 1. Sau đó write(p mod 1000000007);

Làm theo cách này đạt 50% số điểm của bài. Do $\mathbf{a^x}$ có thể rất lớn, nên tràn kiểu dữ liệu của P và bị exit code do khi tính a^x ta không lưu được vào biến kiểu longint hoặc int64.

- **Sub2:** Dựa vào tính chất chia đồng dư.

Ta for(i,1,x) p = p* a mod 1000000007; Tức là tính đến đâu ta chia dư tới đó luôn do tính chất của phép chia đồng dư.

- **Sub3:** Ta nhận thấy rằng: $10^{25} = 10^{12}.10^{12}.10^{1}$; $10^{12} = 10^{6}.10^{6}.10^{0}$;……

Ta tổ chức chương trình tính u mũ v như sau: L = trunc(1e9)+7;

```
Function umuv(u,v: longint): int64;
Var temp: int64;;
Begin
   If v=0 then exit(1);
   If v=1 then exit(u);
   Temp := umuv(u,v div 2);
   Exit((((temp*temp) mod L)*umuv(u,v mod 2) mod L) mod L);
End;
```

NOTE:

- Khi nào cần tính a^b ta luôn tổ chức thành hàm như trên.

```
program caux_010120;
var
      a, x, i: longint;
const
      fi = 'AMUX.INP';
      fo = 'AMUX.OUT';
      l = trunc(1e9)+7;
function amux(u,v: longint): int64;
var
      temp: int64;
```

```
begin
      if v = 0 then exit(1);
      if v = 1 then exit(u);
      temp := amux(u,v div 2);
      exit(((sqr(temp) mod l)*amux(u,v mod 2) mod l) mod l);
end;
begin
      assign(input,fi); reset(input);
         assign(output,fo); rewrite(output);
      readln(a,x);
      write(amux(a,x));
      close(input); close(output);
end.
```

9) SQUA.*

Cho số nguyên X ($|X| < 10^{18}$). Hãy tính căn bậc 3 của X. Kết quả lấy chính xác đến 0.001.

Ví dụ: X = 27 → KQ: 3.000

Thuật toán:

- **Sub1:** Dùng hàm có sẵn, tuy nhiên phải sử dụng đến kiến thức logarit, học sinh chưa học đến.
- **Sub2:** Ta tiếp cận bài toán theo kỹ thuật: ***Chặt nhị phân***.

Vì căn bậc ba của một số nguyên cho kết quả là một số thực. Nếu lưu dấu của X vào biến dau, thì ta tính tổng quát được là tính căn bậc 3 của số nguyên dương X.

Ta thử các giá trị trong đoạn [1,X], nếu số nào mũ 3 bằng X hoặc mũ 3 chênh lệch với X không quá 0.001 thì đưa nó ra.

Nhận xét $\sqrt[3]{x} \leq \sqrt{x}$. Do đó ta sẽ thử các giá trị trong đoạn $[1,\sqrt{x}]$

```
Function canbac3(u: int64): real;
Var dau,giua,cuoi:real;
Begin
  Dau:=1;
  Cuoi:=sqrt(x);
   While cuoi-dau>0.001 do
   Begin
      Giua:= (dau+cuoi)/2;
      If giua*giua*giua < X then dau:=giua
      Else cuoi:=giua;
    End;
  Exit(giua);

End;
```

NOTE:

- Luôn nhớ tư tưởng ***chặt nhị phân*** là thử các giá trị trong khoảng nào đó, nếu giá trị đó chưa phải nghiệm cần tìm, thì thử tìm đoạn trên hoặc đoạn dưới xem có giá trị cần tìm hay không.
- Tức là sau mỗi vòng lặp, không gian tìm kiếm sẽ giảm đi một nửa ~ độ phức tạp thuật toán O(logN).

```
program sub1;
var
	n: int64;
	c3: real;
const
	fi = 'SQUA.INP';
	fo = 'SQUA.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	c3 := exp(ln(n)/3); //han che dung ham nay
	write(c3:0:3);
	close(input); close(output);
end.
```

10) CKPRIME.*

Số nguyên tố là số chỉ có hai ước, 1 và chính nó. Số 1 không là số nguyên tố.

Cho số nguyên dương N ($N \leq 10^9$). Kiểm tra N có là số nguyên tố, nếu có ghi ra TRUE ngược lại ghi ra FALSE.

N = 10 → FALSE; N = 11 → TRUE.

Thuật toán: Để kiểm ra số N có là số nguyên tố, ta đi đếm số lượng ước của nó, nếu nó chỉ có 2 ước, nó là số nguyên tố ngược lại, nó không là số nguyên tố.

- **Sub1:** For(i,1,N) nếu N chia hết cho i thì tăng đếm lên. Nếu đếm =2 thì write(true) ngược lại write(false); Độ phức tạp thuật toán O(N) ~ $N \leq 10^6$.
- **Sub2:** Tương tự Sub1 nhưng for(i,1,N/2). Độ phức tạp thuật toán O(N/2) ~ $N \leq 2.10^6$.
- **Sub3:** Nhận xét: Nếu N≤1 thì N không nguyên tố. Nếu N=2 hoặc 3 thì N nguyên tố. N > 3 thì ta kiểm tra xem nó có tồn tại ước i bé nhất thuộc đoạn $[2,[\sqrt{N}]]$. Nếu có, thì thời điểm này N có 3 ước: 1, i và N → kết luận N không nguyên tố.

Sau khi for xong các i thuộc đoạn trên, nếu không bị exit(false) lần nào thì chứng tỏ N là số nguyên tố, ta exit(true);

```
Function nguyento(u: longint): boolean;
Var j: longint;
```

```
Begin
   If u <=1 then exit(false);
  If (u=2) or (u=3) then exit(true);
  For j:=2 to trunc(sqrt(u)) do
    If u mod j = 0 then exit(false);
  Exit(true);
End;
```

Độ phức tạp thuật toán: $O(\sqrt{N}) \sim N \leq 10^{12}$

NOTE:

- Khi cần kiểm tra tính nguyên tố của một số, ta nên tổ chức thành hàm trên.

```
program caux_010120;
var
	n, i: longint;
const
	fi = 'CKPRIME.INP';
	fo = 'CKPRIME.OUT';
function snt(n: longint): boolean;
begin
	if (n = 2) or (n = 3) then exit(TRUE);
	if (n mod 2 = 0) or (n mod 3 = 0) then exit(FALSE);
	i := 5;
	while i * i <= n do
		begin
			if (n mod i = 0) or (n mod (i+2) = 0) then exit(FALSE);
			inc(i,6);
		end;
	exit(TRUE);
end;
begin
	assign(input,fi); reset(input);
	  assign(output,fo); rewrite(output);
	readln(n);
	write(snt(n));
	close(input); close(output);
end.
```

11) FPRIME.*

Dãy số Fibonacci là dãy số có tính chất:

$F_1 = 1$, $F_2 = 1$, F[N] = F[N-1] + F[N-2].

Như vậy, các số Fibonacci đầu tiên là: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…

Cho số nguyên dương P. Hãy đếm số lượng các số Fibonacci nhỏ hơn hoặc bằng P và là số nguyên tố.

Ví dụ: P = 10 → KQ: 3

Thuật toán:

- **Sub1**: Lần lượt tạo ra các số Fibonacci, đến số Fibonacci thứ i ta kiểm tra tính nguyên tố của nó, nếu thoả mãn tăng biến đếm lên.

```
F[1]:=1; F[2]:=1; res:=0;
i:=3; F[i]:=f[i-1]+f[i-2];
while f[i] <= p do
begin
   if nguyento(f[i]) then inc(res);
   inc(i);
   F[i]:=F[i-1]+F[i-2];
 end;
Write(res);
```

Độ phức tạp thuật toán: O(res.$\sqrt{f[imax]}$)

Làm theo cách này, ta chỉ đạt không quá 50% điểm của bài.

- **Sub2:** Ta dùng Sàng nguyên tố Eratosthenes tới P để lưu được mảng trạng thái c[i]=1 nếu i nguyên tố, c[i]=0 với i không nguyên tố. Sau đó ta vẫn tổ chức chương trình như **sub1** chỉ thay đổi ở chỗ kiểm tra tính nguyên tố của F[i].

```
F[1]:=1; F[2]:=1; res:=0;
i:=3; F[i]:=f[i-1]+f[i-2];
while f[i] <= p do
begin
   if C[f[i]] then inc(res);
   inc(i);
   F[i]:=F[i-1]+F[i-2];
 end;
Write(res);
```

NOTE:

- Thủ tục inc(tenbien); tương đương với lệnh tenbien:=tenbien+1;
- inc(tenbien,giatri); tương đương với lệnh tenbien:=tenbien+giatri;

```
program caux_010120;
var
      n, i, k, d: longint;
      f: array[1..10000] of int64;
const
      fi = 'FPRIME.INP';
      fo = 'FPRIME.OUT';
```

```
function snt(n: longint): boolean;
begin
	if (n = 2) or (n = 3) then exit(TRUE);
	if (n mod 2 = 0) or (n mod 3 = 0) then exit(FALSE);
	i := 5;
	while i * i <= n do
		begin
			if (n mod i = 0) or (n mod (i+2) = 0) then exit(FALSE);
			inc(i,6);
		end;
	exit(TRUE);
end;
begin
	assign(input,fi); reset(input);
	  assign(output,fo); rewrite(output);
	readln(k);
	if n <= 2 then
	f[1] := 1;
	f[2] := 1;
	i:=3;
	repeat
		f[i] := f[i-1] + f[i-2];
		if snt(f[i]) then inc(d);
		inc(i);
	until f[i-1] + f[i-2] >= k;
	write(d);
	readln;
	close(input); close(output);
end.
```

12) ERATOS.*

Cho số nguyên dương N ($N \le 10^7$). Hãy liệt kê các số nguyên tố nhỏ hơn hoặc bằng N. Mỗi số cách nhau một dấu cách.

Ví dụ: N = 10 → 2 3 5 7

Thuật toán:

- **Sub1:** For(i,1,N) nếu i nguyên tố thì đưa i ra. Độ phức tạp thuật toán $O(N.\sqrt{N}) \sim N \le 10^4$.
- **Sub2:** Sử dụng giải thuật Sàng nguyên tố Eratosthenes có độ phức tạp $O(N) \sim N \le 10^{6\to7}$.

```
program caux_010120;
var
```

```
        n, i, j, maxC: longint;
        e: array[1..10000000] of boolean;
const
        fi = 'ERATOS.INP';
        fo = 'ERATOS.OUT';
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n);
        fillchar(e,sizeof(e),TRUE);
        maxC := trunc(sqrt(n));
        i := 2;
        while i <= maxC do
                begin
                        while e[i] = FALSE do inc(i);
                        for j:=2 to n div i do
                                e[i*j] := FALSE;
                        inc(i);
                end;
        for i:=2 to n do
                if e[i] then write(i,#32);
        close(input); close(output);
end.
```

Care and diligence bring luck!

Ngày 2/7 SOLVE

1) TSNT.*

Cho số nguyên dương N. Phân tích N thành thừa số nguyên tố.

Thuật toán:

- **Sub1:** Cho i chạy từ 2 tới N, nếu i là ước của N và i nguyên tố thì: đưa i ra và chừng nào N còn chia hết cho i thì tiếp tục chia cho i. Do giá trị của N thay đổi sau mỗi vòng lặp, ta nên dùng While..do.

  Độ phức tạp thuật toán O(N).

  Làm theo cách này ta đạt không quá 50% điểm của bài.

- **Sub2:** Cho i chạy từ 2 tới $[\sqrt{N}]$, nếu N chia hết cho i thì, chừng nào N còn chia hết cho i thì đưa i ra và giảm N. Nếu N không chia hết cho i thì tăng i lên.

Sau vòng while..do, nếu N>1 thì chứng tỏ phần còn lại của N là số nguyên tố, ta đưa N ra.

Ví dụ: N = 26 → 2 13 (ở đây i chỉ chạy tới 5)

Độ phức tạp thuật toán O($\sqrt{N}$)

NOTE:

- Nên tổ chức chương trình thành thủ tục phantich(u: longint);
- Sử dụng mảng C để lưu nghiệm. Mảng C có K phần tử.

```
//K khai báo biến toàn cục
Procedure phantich(u: longint);
Var j: longint;
Begin
    J:=2;
    K:=0;
    While j <= trunc(sqrt(n)) do
    begin
        While u mod j = 0 do
         Begin
              K:= K +1;
              C[k]:=j;
              U := u div j;
        End;
        J:= j+1;
    End;
    For j:=1 to k do write(c[k],#32);
End;
```

```
uses math;
var
      n, i: longint;
```

```
const
      fi = 'TSNT.INP';
      fo = 'TSNT.OUT';
begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
      readln(n);
      i := 2;
         repeat
                  while n mod i <> 0 do inc(i);
                  write(i,#32);
                  n := n div i;
         until n = 1;
      close(input); close(output);
end.
```

2) NFIND.*

Cho số nguyên dương M ($M \leq 10^9$). Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M. Nếu không tồn tại K, đưa ra -1.

Thuật toán:

- **Sub1:** Ta tổ chức hàm tính tích các chữ số của u:

  function tich(u:longint):longint;

Khởi tạo K = 1; Chừng nào tich(K) < M thì tăng K lên.

Nếu tich(k) = M thì đưa K ra, ngược lại ghi ra -1.

Độ phức tạp thuật toán O(M).

Tuy nhiên thuật toán này này chỉ đúng một phần.

- **Sub2:** Ta nhận thấy: Ví dụ M = 26 = 2*13; Chứng tỏ M không thể là tích các chữ số của số K nào vì có số 13 > 9. Hay nói cách khác, tích các chữ số của K chỉ là tích của các số có giá trị từ '0'..'9'.

Vậy, để kiểm tra có nghiệm hay không, ta phân tích M thành thừa số nguyên tố, nếu có thừa số nào có giá trị > 9 thì ta exit(false) luôn.

Xác định được trường hợp có nghiệm rồi thì ta xử lí khéo léo như sau:

Ví dụ: N = 28 = 2.2.7 = 4.7 → KQ: 47

Tức là sau khi phân tích N thành thừa số nguyên tố ta thu được mảng C có P phần tử của N. Ta sẽ rút gọn mảng từ ***trái sang phải*** bằng cách nhân các thừa số lại với nhau sao cho tích các thừa số này nhỏ hơn hoặc bằng 9. Sau đó đẩy tích này vào mảng B. tiếp tục lại rút gọn mảng C như trên với các phần tử còn lại.

Thu được mảng B có N phần tử, ta sắp xếp mảng B thành dãy không giảm. Đưa mảng B ra là nghiệm của bài

Ví dụ: M =56700 = 2.2.3.3.3.3.5.5.7=4.9.9.5.5.7 = 4.5.5.7.9.9→ K = 455799

Độ phức tạp thuật toán: O($\sqrt{N}$) do chủ yếu chi phí phân tích N thành thừa số nguyên tố.

NOTE:

- Kĩ năng sử dụng mảng C lưu nghiệm.

```
uses math;
var
	i: longint;
	n: int64;
	st, res: string;
const
	fi = 'NFIND.INP';
	fo = 'NFIND.OUT';
function bbu(n: longint): string;
begin
	if n = 0 then exit('10')
	else if n = 1 then exit('1');
	i := 9;
	   while (n > 1) and (i > 1) do
	      begin
	                  while n mod i = 0 do
	                  begin
	                        str(i,st);
	                        res := st + res;
	                              n := n div i;
	                  end;
	            dec(i);
	         end;
	if n > 1 then exit('-1') else exit(res);
end;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	write(bbu(n));
	close(input); close(output);
end.
```

3) NUMBERC.*

Cho số nguyên dương N. Người ta có thể cắt lần lượt các chữ số của N từ phải sang trái để thu được số N mới.

Hãy tìm số N mới là số nguyên tố lớn nhất. Biết phải cắt ít nhất 1 số. Nếu không có ghi ra -1. Ví dụ: N = 1124 → N = 11.

Thuật toán:

Lần lượt cắt các chữ số ở hàng đơn vị của N đi, ta thu được N mới, nếu N mới là số nguyên tố thì dừng lại, đưa ra kết quả. Sau khi cắt và kiểm tra, nếu N = 0 chứng tỏ không có số N mới nào là số nguyên tố, ta ghi ra -1.

Độ phức tạp thuật toán: O($\sqrt{N}$.loglogN)

```
uses math;
var
	i: longint;
	n: int64;
const
	fi = 'NUMBERC.INP';
	fo = 'NUMBERC.OUT';
function snt(n: int64): boolean;
begin    if (n<=1) then exit(false);
	if (n = 2) or (n = 3) then exit(TRUE);
	if (n mod 2 = 0) or (n mod 3 = 0) then exit(FALSE);
	i := 5;
	while i * i <= n do
		begin
			if (n mod i = 0) or (n mod (i+2) = 0) then exit(FALSE);
			inc(i,6);
		end;
	exit(TRUE);
end;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	while n div 10 > 0 do
		begin
			n := n div 10;
			if snt(n) then begin write(n); exit; end;
		end;
	write('-1');
	close(input); close(output);
end.
```

4) ARRDEL.*

Xóa các phần tử giống nhau đứng cạnh nhau trong dãy số, với những số này chỉ giữ lại đại diện một số. Đưa dãy sau khi xóa ra màn hình, giữ nguyên thứ tự đầu vào.

Ví dụ: N = 9; dãy 1 3 3 4 2 2 2 7 3 ; → 1 3 4 2 7 3

Thuật toán:

Thay vì xoá các phần tử liên tiếp giống nhau ta sẽ tạo mảng C chứa các phần tử liên tiếp khác nhau của a.

Ta dùng kĩ năng sử dụng mảng C có K phần tử để lưu nghiệm.

```
K:=0;
For i:=1 to N do
  If a[i]<>a[i+1] then
Begin
   inc(k); c[k]:=a[i];
End;
For i:=1 to K do write(c[i]);
```

NOTE:

Đoạn code trên là vận dụng kỹ năng sử dụng mảng C lưu nghiệm và cách cài đặt chương trình

```
uses math;
var
      n, i: longint;
      a: array[1..1000000] of longint;
const
      fi = 'ARRDEL.INP';
      fo = 'ARRDEL.OUT';
begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
      readln(n);
      read(a[1]);
      i := 2;
      while i <= n do
            begin
                  read(a[i]);
                  if a[i] <> a[i-1] then inc(i)
                  else dec(n);
            end;
      for i:=1 to n do write(a[i],#32);
      close(input); close(output);
end.
```

5) QSORT.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Hãy sắp xếp dãy này thành dãy không giảm bằng thuật toán Sắp xếp nhanh Quick Sort. Đưa dãy sau khi sắp xếp ra.

Thuật toán:

Ta sử dụng thuật toán Sắp xếp nhanh Quicksort để sắp xếp dãy số thành dãy không giảm.
Ta nên cố định code sắp xếp nhanh như sau:

```
Procedure qsort(L, H:longint);
Var i,j,mid, temp:longint;
Begin
   If (L>H) then exit();
   I:=L;
   J:=H;
  Mid:=a[(i+j) div 2];
   While i<=j do
   Begin
       While a[i]<mid do inc(i);
       While a[j]>mid do dec(j);
       If i<=j then
       Begin
           Temp:=a[i];
           A[i]:=a[j];
           A[j]:=temp;
      End;
      Inc(i); dec(j);
   End;
   If L < j then qsort(L,j);
   If i < H then qsort(i,H);
End;
```

Trong chương trình chính gọi: qsort(1,N); để sắp xếp dãy số
Giải thuật sắp xếp nhanh phù hợp với dãy số có $N \leq 10^5$.
Độ phức tạp thuật toán là: O(N.logN).

```
uses math;
var
      n, i: longint;
      a: array[1..100000] of longint;
const
      fi = 'QSORT.INP';
      fo = 'QSORT.OUT';
procedure sort(l,r: longint);
var
      i, j, x, y: longint;
begin
      i := l;
      j := r;
```

```
        x := a[(l+r) div 2];
        repeat
            while a[i] < x do inc(i);
            while x < a[j] do dec(j);
            if i <= j then
                begin
                    y := a[i];
                    a[i] := a[j];
                    a[j] := y;
                    inc(i); dec(j);
                end;
        until i > j;
        if l < j then sort(l,j);
        if i < r then sort(i,r);
end;
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n);
        for i:=1 to n do read(a[i]);
        sort(1,n);
        for i:=1 to n do write(a[i],#32);
        close(input); close(output);
end.
```

Trong các chương trình, khi có nhu cầu sắp xếp chúng ta nên sử dụng hàm qsort này.

6) ILUCKY.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Tìm các vị trí i (nếu có) chia dãy số thành 2 phần có tổng bằng nhau. Nếu không có thì ghi ra -1.

Thuật toán:

- **Sub1:** Duyệt từ đầu đến cuối, đến vị trí thứ i, ta tính tổng đoạn bên trái từ [1,i] lưu vào S1, đoạn bên phải từ i+1 tới N lưu vào S2. Nếu S1=S2 thì đưa i ra.
  Độ phức tạp thuật toán $O(N^2)$.
  Làm theo cách này bạn đạt không quá 50% điểm của bài.
- **Sub2:** Ta gọi S[i] là tổng các phần tử từ 1 tới i. S[1] = a[1]; duyệt i từ 2 tới N ta tính S[i] = S[i-1] + a[i];

Duyệt i từ 1 tới N, nếu S[i] = S[n]/2 thì đưa i ra.

Độ phức tạp thuật toán O(N).

Lưu ý: S[i] là tổng các phần tử nên S[i] có thể rất lớn.

NOTE:

- Mảng S[i] gọi là mảng **tổng tích luỹ.**

```
uses math;

var
	n, i, j, d: longint;
	a: array[1..1000000] of longint;
	s1, s2: int64;
const
	fi = 'ILUCKY.INP';
	fo = 'ILUCKY.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	for i:=1 to n do
		begin
			read(a[i]);
			inc(s2,a[i]);
		end;
	for i:=1 to n do
		begin
			inc(s1,a[i]);
			dec(s2,a[i]);
			if s1=s2 then
				begin
					write(i,' ');
					inc(d);
				end;
		end;
	if d=0 then write(-1);
	close(input); close(output);
end.
```

7) PMIN.*

Cho N và dãy a_1, a_2,…,a_N. Đưa ra các vị trí phần tử đạt giá trị nhỏ nhất.

Ví dụ: N = 6; dãy 2 1 4 2 1 4 → 2 và 5.

Thuật toán:

Tìm giá trị nhỏ nhất của dãy số lưu vào biến Rmin.

Duyệt lại từ đầu đến cuối dãy số, nếu a[i] = Rmin thì đưa i ra.

Độ phức tạp thuật toán O(N).

```
uses math;var
    n, i: longint;
    res: longint = high(longint);
    a: array[1..1000000] of longint;
const
    fi = 'PMIN.INP';
    fo = 'PMIN.OUT';
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:=1 to n do
        begin
            read(a[i]);
            res := min(res,a[i]);
        end;
    for i:=1 to n do if a[i] = res then write(i,#32);
    close(input); close(output);
end.
```

8) PSECOND.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Tìm giá trị nhỏ nhì (nếu có) của dãy số và đưa ra các vị trí đạt giá trị đó. Nếu không có thì ghi ra -1.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → min2 = 2; vitri = 4

Subtask1: $N \le 10^5$

Subtask2: $N \le 10^6$, $a_i \le 10^6$

Subtask3: $N \le 10^6$, $a_i \le 10^9$

Thuật toán:

- **Sub1:** Vì $N \le 10^5$, ta dùng thủ tục sắp xếp nhanh Quicksort để xếp lại mảng a thành dãy không giảm.
  Duyệt từ đầu dãy đến cuối dãy, phần tử đầu tiên khác a[1] là phần tử nhỏ nhì. Nếu không có phần tử nào là giá trị nhỏ nhì (dãy bằng nhau) thì ghi ra -1.
  Lưu ý: *Sử dụng biến p = 0; mỗi lần cập nhật nghiệm ta gắn p = 1; Sau đoạn For, nếu p = 0 thì chứng tỏ vô nghiệm, ta ghi ra -1*
- **Sub2:** Vì $N \le 10^6$, $a_i \le 10^6$ nên ta dùng mảng B để đếm phân phối các phần tử a[i].
  Duyệt từ 1 đến 100.000,
  nếu a[i] <> 0 thì {tăng đếm, nếu đếm =2 thì đưa a[i] ra và break;}

Sau vòng lặp, nếu đếm=0 thì vô nghiệm, ta ghi ra -1.

- **Sub3:** Vì $N \le 10^6$, $a_i \le 10^9$ nên ta làm như sau:

Tìm giá trị nhỏ nhất của dãy số lưu vào Rmin.

Duyệt từ đầu đến cuối dãy số, nếu a[i] > Rmin và a[i] < res thì cập nhật lại res, ghi nhận P. Nếu p = 0 thì đưa ra -1 ngược lại đưa ra 0.

```
uses math;
const
    fi = 'PSECOND.INP';
    fo = 'PSECOND.OUT';
var
    n, i: longint;
    a: array[1..1000000] of longint;
    min1: longint = high(longint);
    min2: longint = low(longint);
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:=1 to n do
        begin
            read(a[i]);
            min1 := min(min1,a[i]);
            min2 := max(min2,a[i]);
        end;
    if min1 = min2 then begin write(-1); exit; end;
    for i:=1 to n do
        if (a[i] <= min2) and (a[i] > min1) then min2 := a[i];
    for i:=1 to n do if a[i] = min2 then write(i,#32);
    close(input); close(output);
end.
```

Little by little does the trick!

Ngày 3/7

1) GCD.*

Cho a, b, c nguyên dương. Hãy tìm ước chung lớn nhất của a, b và c.

Thuật toán:

- **Sub1:** gọi d là ước chung lớn nhất của a, b và c. Khi đó giá trị của d thuộc đoạn [1,min(a,b,c)].

Tìm min của a, b,c: rmin := min(a, min(b,c)); // nhớ dùng thư viện toán học

Res :=1;

For(i,1,rmin)

If (a mod i = 0) and (b mod i = 0) and (c mod i = 0) then res :=i;

Đưa res ra.

Độ phức tạp thuật toán: O(min(a,b,c));

Sử dụng thuật toán này ta chỉ đạt không quá 50% điểm của bài.

- **Sub2:** Ta tổ chức hàm tìm ước chung lớn nhất của u và v nguyên dương theo thuật toán **ơ'clit**.

```
Function GCD(u, v: longint): longint;
Var r: longint; // r là số dư của u chia cho v
Begin
    While v>0 do
      Begin
          R:= u mod v;
          U:= v;
          V:=r;
      End;
   Exit(u);
End;
```

Kết quả của bài toán: res:= gcd(a,gcd(b,c));

Độ phức tạp thuật toán: O(logloga);

NOTE:

- Tổng quát lên, để tìm GCD của dãy $a_1, a_2, \ldots, a_N$. Ta gắn res := a[1];
  For(i,2,N) res:=gdc(res,a[i]);
- Bội chung nhỏ nhất của a và b là: a*b div GCD(a,b); Tổng quát lên ta cũng có thể tìm BCNN của N phần tử.

```
uses math;
const
    fi = 'GCD.INP';
    fo = 'GCD.OUT';
var a, b, r, c: int64;
```

```
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(a,b,c);
    while b<>0 do
        begin
            r:=a mod b;
            a:=b;
            b:=r;
        end;
    while c<>0 do
        begin
            r:=a mod c;
            a:=c;
            c:=r;
        end;
    write(a);
    close(input); close(output);
end.
```

2) ARRADD2.*

Cho dãy a (N phần tử) và dãy b (M phần tử) là dãy không giảm. Tạo dãy c có M+N phần tử cũng là dãy không giảm. Đưa dãy C ra.

Ví dụ: N = 3; dãy a: 1 3 4; M = 3; dãy b: 2 3 7; → dãy C: 1 2 3 3 4 7

Thuật toán:

- **Sub1:** Ta tạo mảng C có: N phần tử từ 1 tới N là của dãy a, từ N+1 tới N+M của dãy B.

  Sắp xếp mảng C thành dãy không giảm theo thuật toán sắp xếp nhanh QuickSort M+N phần tử.

  Độ phức tạp thuật toán: O((M+N).log(M+N)).
- **Sub2:** Ta sử dụng biến chạy i trên dãy a, biến chạy j trên dãy b. Giả thiết cho dãy a và dãy b là dãy không giảm do vậy, để tạo ra dãy C là dãy không giảm ta sẽ so sánh lần lượt các phần tử a[i] với b[j], phần tử nào bé hơn ta sẽ đẩy vào mảng C.

Ta sử dụng kỹ thuật cấp phát một phần tử mới và đẩy vào C.

```
K:= K+1;
C[k]:= X; // trong đó X là giá trị cần đẩy vào mảng C.
```

Nếu mảng a đã hết, ta đẩy các phần tử còn lại của b vào C. Còn nếu b đã hết ta đẩy các phần tử còn lại của a vào C.

Như vậy, ta thu được mảng C gồm M+N phần tử là dãy không giảm.

Độ phức tạp thuật toán: O(M+N).

```
uses math;
var
    n, i, j, k, d1, m: longint;
    a, b, c: array[1..200000] of LongInt;
const
    fi = 'ARRADD2.INP';
    fo = 'ARRADD2.OUT';
begin
    assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
    readln(m,n);
    for i:=1 to m do read(a[i]);
    for i:=1 to n do read(b[i]);
    k := 1;
    j := 1;
    d1 := 1;
    while (k <= m) and (j <= n) do
        begin
            if a[k] > b[j] then
                begin
                    c[d1] := a[k];
                    inc(d1);
                    inc(k);
                end
            else
                begin
                    c[d1] := b[j];
                    inc(d1);
                    inc(j);
                end;
        end;
    if k <= m then
        for i:=k to m do
            begin
                c[d1] := a[i];
                inc(d1);
            end;
    if j <= n then
        for i:=j to n do
            begin
```

```
                    c[d1] := b[i];
                    inc(d1);
                end;
    for i:=1 to d1-1 do write(c[i],#32);
    close(input); close(output);
end.
```

3) ARRCOUNT.*

Cho N và dãy số nguyên dương a_1, a_2, …, a_N. Đếm số lần xuất hiện của mỗi phần tử trong dãy a.

Subtask1: a[i] ≤ 10^6.

Subtask2: a[i] ≤ 10^9.

- **Sub1:** Vì a[i] ≤ 10^6 nên ta sử dụng kỹ thuật ***đếm phân phối***. Ta thu được mảng b[i] lưu số lần xuất hiện của i.

Duyệt từ 1 tới 10^6, nếu b[i] khác 0 thì đưa số i xuất hiện b[i] lần ra.

```
For(i,1,N)
  B[a[i]] := B[a[i]]+1;
For(i,1,1000000)
   If b[i] <> 0 then writeln(i,#32,b[i]);
```

Sử dụng thuật toán này ta chỉ đạt không quá 50% điểm của bài.

- **Sub2**: Vì a[i] ≤ 10^9 nên ta không thể dùng phương án đếm phân phối. Vậy ta xử lý khéo léo như sau:

  Sắp xếp dãy a thành dãy không giảm bằng giải thuật Quicksort.

  Duyệt từ đầu đến cuối dãy a, nếu a[i]=a[i+1] thì tăng đếm lên, ngược lại thì đưa a[i], đếm ra và khởi tạo lại đếm = 1;

  Độ phức tạp thuật toán: O(N.logN) do chi phí chủ yếu ở phần sắp xếp Quicksort.

NOTE:

- Bạn nên nắm chắc kỹ thuật đếm phân phối.
- Khi có nhu cầu sắp xếp, luôn dùng QuickSort để sắp xếp với chi phí ít nhất.

```
uses math;
var
      n, i, d: longint;
      a: array[1..100000] of longint;
const
      fi = 'ARRCOUNT.INP';
      fo = 'ARRCOUNT.OUT';
procedure sort(l,r: longint);
var
      i, j, x, y: longint;
```

```
begin
        i := l;
        j := r;
        x := a[(l+r) div 2];
        repeat
                while a[i] < x do inc(i);
                while x < a[j] do dec(j);
                if i <= j then
                        begin
                                y := a[i];
                                a[i] := a[j];
                                a[j] := y;
                                inc(i);
                                dec(j);
                        end;
        until i > j;
        if l < j then sort(l,j);
        if i < r then sort(i,r);
end;

begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n);
        for i:=1 to n do read(a[i]);
        sort(1,n);
        i := 1;
        while i <= n do
                begin
                        d := 1;
                        write(a[i],#32);
                        while a[i] = a[i+1] do
                                begin
                                        inc(d);
                                        inc(i);
                                end;
                        inc(i); writeln(d);
                end;
        close(input); close(output);
end.
```

4) BINS.*

Cho N, X và dãy số nguyên a_1, a_2, …, a_N là dãy đơn điệu tăng. Hãy tìm vị trí của X trong dãy a. Nếu không có X trong dãy a thì ghi ra -1.

Thuật toán: Tìm kiếm nhị phân:

Ban dầu khởi tạo đầu = 1; cuối = N;

Chừng nào đầu <= cuối thì

{

Giữa = a[(dau+cuoi) div 2];

Nếu a[giữa]= X thì exit(giữa);

Nếu a[giữa] < X thì đầu = giữa + 1 ngược lại cuối = giữa -1;

}

Vòng while trên dừng khi: hoặc bị exit(giữa) – trường hợp có nghiệm; hoặc giá trị đầu > cuối – vô nghiệm.

Vậy ta xét thêm:

Nếu đầu > cuối thì exit(-1);

Trong chương trình chính: write(bins(X));

Độ phức tạp của thuật toán tìm kiếm nhị phân: O(logN).

```
Function TKNP(u: longint): longint;
Var dau,cuoi,giua: longint;
Begin
   Dau:= 1; cuoi := N;
   While dau <= cuoi do
    Begin
        Giua := (dau + cuoi) div 2;
        If a[giua] = X then exit(giua);
        If a[giua] < X then dau:=giua +1
        Else cuoi:=giua-1;
   End;
   If dau>cuoi then exit(-1);
End;
```

NOTE:

- Thuật toán TKNP chỉ áp dụng trên dãy đơn điệu tăng (tăng ngặt) hoặc giảm ngặt.
- Mở rộng bài toán này, cho dãy x[i], hãy tìm vị trí của x[i] trong dãy a[i].

```
uses math;
var
      n, i, x: longint;
      a: array[1..1000000] of longint;
const
```

```
        fi = 'BINS.INP';
        fo = 'BINS.OUT';
function TKNP(s, f, t: longint): longint;
var
        mid: longint;
begin
        while s <= f do
                begin
                        mid := (s+f) div 2;
                        if t = a[mid] then exit(mid)
                        else if t < a[mid] then f := mid - 1
                        else s := mid + 1;
                end;
        if s > f then exit(-1);
end;
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n,x);
        for i:=1 to n do read(a[i]);
        write(TKNP(1,n,x));
        close(input); close(output);
end.
```

5) UPSEQ.*

Dãy liên tiếp không giảm dài nhất

Cho N và dãy số nguyên a_1, a_2, …, a_N. Tìm dãy con liên tiếp không giảm dài nhất của dãy. Đưa ra độ dài lớn nhất tìm được.

VD: N = 6; dãy 1 3 4 2 7 3 → 3 (Giải thích: dãy không giảm dài nhất là dãy 1 3 4)

Thuật toán:

Duyệt từ đầu đến cuối, chừng nào a[i]<=a[i+1] thì tiếp tục tăng temp lên; cập nhật res = max(res, temp) và khởi tạo lại temp = 1;

```
Temp:=1;
I:=1;
While i<n do
Begin
    While (i<n) and a[i] <= a[i+1] do
        begin
        inc(i); inc(temp);
        end;
```

```
    Res := max (res, temp);
     Temp:=1;
     Inc(i);
End;
Write(res);
```

Độ phức tạp thuật toán: O(N).

```
uses math;
var
        n, i: longint;
        d: longint = 1;
        res: longint = 0;
        a: array[1..1000000] of longint;
const
        fi = 'UPSEQ.INP';
        fo = 'UPSEQ.OUT';
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n);
        read(a[1]);
        for i:=2 to n do
                begin
                        read(a[i]);
                        if a[i] >= a[i-1] then inc(d)
                        else
                                begin
                                        res := max(res,d);
                                        d := 1;
                                end;
                end;
        write(max(res,d));
        close(input); close(output);
end.
```

6) ARR2D.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy tính và đưa ra tổng của từng hàng, tổng của từng cột.

Thuật toán:

Ta gọi **hang[i]** là tổng của hàng thứ i.

Cot[j] là tổng của cột thứ j.

Duyệt qua các phần tử của mảng a, đến hàng thứ i ta đẩy a[i,j] vào tổng hang[i].
Duyệt qua các phần tử của mảng a, đến cột thứ j ta đẩy a[i,j] vào tổng cot[j].
Đưa mảng hang[i] và cot[j] ra.

```
For i:=1 to m do
Begin
        Hang[i]:=0;
        For j:=1 to n do hang[i]:=hang[i]+a[i,j];
End;
For j:=1 to n do
Begin
     Cot[j] :=0;
     For i:=1 to m do cot[j] := cot[j] + a[i,j];
End;
For i:=1 to m do write(hang[i],#32);
Writeln();
For j:=1 to n do write(cot[j],#32);
```

NOTE:

Mở rộng bài toán: Cho mảng a kích thước mxn hãy:

- Tìm hàng có tổng là số nguyên tố lớn nhất.
- Tìm K cột để tổng K cột này là lớn nhất.
- Tìm vị trí (i,j) có tổng hàng[i] + tổng cột[j] là lớn nhất.

```
uses math;
var
        m, n, i, j: longint;
        a: array[-7..3007,-7..3000] of longint;
        res1, res2: array[1..3000] of longint;
const
        fi = 'ARR2D.INP';
        fo = 'ARR2D.OUT';
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(m,n);
        for i:=1 to m do
                for j:=1 to n do
                        begin
                                read(a[i,j]);
                                inc(res1[i],a[i,j]);
                                inc(res2[j],a[i,j]);
```

```
                end;
        for i:=1 to m do write(res1[i],#32);
        writeln();
        for i:=1 to n do write(res2[i],#32);
        close(input); close(output);
end.
```

7) ARR2DPRI.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy đưa ra vị trí các phần tử là số nguyên tố.

Thuật toán:

Tổ chức hàm Function nguyento(u:longint) : boolean;

Duyệt từ đầu đến cuối mảng a, đến phần tử thứ (i,j) ta kiểm tra a[i,j] có là số nguyên tố, nếu có đưa (i,j) ra.

```
// Function nguyento(u:longint) : boolean; đã có trong các bài trước
For i:=1 to m do
  For j:=1 to n do
    If ngto(a[i,j]) then writeln(i,#32,j);
```

```
uses math;
var
        m, n, i, j: longint;
        f: boolean = FALSE;
        a: array[1..300,1..300] of longint;
const
        fi = 'ARR2DPRI.INP';
        fo = 'ARR2DPRI.OUT';
function snt(x: longint): boolean;
var o: longint;
begin
        if (x = 2) or (x = 3) then exit(TRUE);
        if (x < 2) or (x mod 2 = 0) or (x mod 3 = 0) then exit(FALSE);
        o := 5;
        while sqr(o) <= x do
                begin
                        if (x mod o = 0) or (x mod (o+2) = 0) then exit(FALSE);
                        inc(o,6);
                end;
        exit(TRUE);
end;
```

```
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(m,n);
    for i:=1 to m do begin
        for j:=1 to n do
            begin
                read(a[i,j]);
                if snt(a[i,j]) then
                    begin
                        f := TRUE;
                        writeln(i,#32,j);
                    end;
            end;
    end;
    if f = FALSE then write(-1);
    close(input); close(output);
end.
```

8) ARR2DSQU.*

Cho mảng a là mảng hai chiều kích thước M x N. Hãy tìm hình vuông có cạnh bằng K có tổng lớn nhất. Đưa ra các góc trái trên của các hình vuông và tổng lớn nhất tìm được.

Sub1: Duyệt qua tất cả các hình vuông cạnh K, với mỗi hình vuông ta tính tổng các phần tử trong hình vuông đó, cập nhật lại res.

Ta duyệt lại các hình vuông cạnh K, nếu tổng các phần tử trong hình vuông đó = res thì đưa ra góc trái trên.

Độ phức tạp thuật toán: O(m.n.k)

Sub2: Goi S[i,j] là tổng các phần tử từ ô (1,1) đến ô (i,j)

(1,1)				
	(i-1,j-1)	(i-1,j)		
	(i,j-1)	(i,j)		

Ta có s[0,0] := 0;

For(i,1,m)

For(j,1,n) S[i,j] := S[i,j-1] + S[i-1,j] – S[i-1,j-1] + a[i,j];

Ta tính sẵn được mảng s[i,j].

Duyệt qua các hình vuông cạnh K, ta gọi temp là tổng các phần tử trong hình vuông này.

(i-1,j-1)			(i-1,j+k-1)	
	(i,j)			
(i+k-1,j-1)			(i+k-1,j+k-1)	

Tìm hình vuông có tổng lớn nhất.

```
Res:= low(longint);
For i:=1 to m + k -1 do
  For j:=1 to n+k-1 do
     Begin
        Temp:= s[i+k-1,j+k-1] – s[i+k-1,j-1] – s[i-1,j+k-1] + s[i-1,j-1]
        Res := max(res,temp);
End;
```

Duyệt tìm các hình vuông cạnh K tổng = res, đưa góc trái trên ra.

```
For i:=1 to m + k -1 do
  For j:=1 to n+k-1 do
     Begin
        Temp:= s[i+k-1,j+k-1] – s[i+k-1,j-1] – s[i-1,j+k-1] + s[i-1,j-1]
        If temp = res then write(i,#32,j);
End;
```

Độ phức tạp thuật toán: O(m.n).

NOTE: Mở rộng

- Tìm hình vuông cạnh K có tổng là số nguyên tố, số chính phương, số Fibonacci, bằng X…
- Tìm hình vuông lớn nhất có tổng bằng X.

```
uses math;
var
      m, n, i, j, k, res, resX, resY, t: longint;
      a: array[1..300,1..300] of longint;
const
      fi = 'ARR2DSQU.INP';
      fo = 'ARR2DSQU.OUT';
function PlsWork(x, y: longint): longint;
var
```

```
        u, v: longint;
        t2 : longint = 0;
begin
        for u:=x to x1 do
                for v:=y to y1 do inc(t2,a[u,v]);
        exit(t2);
end;

begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(m,n,k);
        for i:=1 to m do
                for j:=1 to n do read(a[i,j]);
        for i:=1 to m-k do
                for j:=1 to n-k do
                        begin
                                t := PlsWork(i,j);
                                if t > res then
                                        begin
                                                res := t;
                                                resX := i;
                                                resY := j;
                                        end;
                        end;
        writeln(res);
        write(resX,#32,resY);
        close(input); close(output);
end.
```

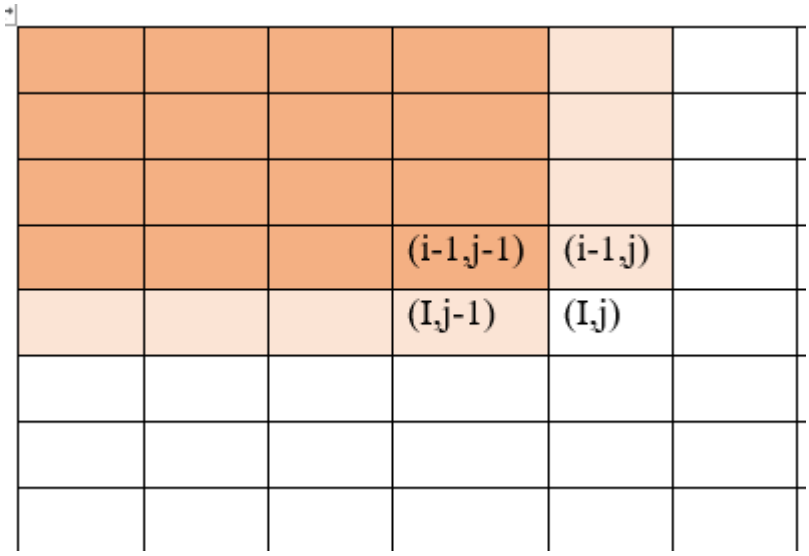
Success is achieved and maintained by those who try and keep trying.

W. Clement Stone

Ngày 4/7

1) ARR2HVS1.*

Cho mảng a là mảng hai chiều kích thước M x N. ***Hãy tìm tất cả các hình vuông có tổng bằng S***.



- Bước 1: Chuẩn bị mảng cộng dồn S[i,j] với S[i,j] là tổng từ ô (1,1) đến ô (i,j). // s[i,j] gọi là tổng tích luỹ.

Ta có s[0,0] = 0;

For(I,1,m)

For (j,1,n)

S[i,j] = s[i-1,j] + s[i,j-1] – s[i-1,j-1] + a[i,j];

- Bước 2: Duyệt qua tất cả các hình vuông kích thước k, ta gọi Q là tổng của các phần tử trong hình vuông này.

Ta có

```
For(k=1,min(m,n))
    For(i,1,m)
      For(j,1,n)
        Begin
            Q:= s[i+k-1,j+k-1] – S[i-1,j+k-1] – s[i+k-1,j-1] + s[i-1,j-1];
            Writeln(I,#32,j,#32,k);
        End;
```

Độ phức tạp thuật toán: O(m.n.min(m,n)).

```
uses math;
var
    n, i, m, s, j, t2, k: longint;
    t, a: array[0..107,0..107] of longint;
    f: boolean = FALSE;
const
```

```
    fi = 'ARR2HVS1.INP';
    fo = 'ARR2HVS1.OUT';
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    read(m,n,s);
    for i:=1 to m do
        for j:=1 to n do
            begin
                read(a[i,j]);
                t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];
            end;
    for k:=1 to min(m,n) do
        for i:=1 to m-k+1 do
            for j:=1 to n-k+1 do
                begin
                    t2 := t[i+k-1,j+k-1] - t[i-1,j+k-1] - t[i+k-1,j-1] + t[i-1,j-1];
                    if t2 = s then
                        begin
                            writeln(i,#32,j,#32,k);
                            f := TRUE;
                        end;
                end;
    if f = FALSE then write(-1);
    close(input); close(output);
end.
```

2) ARR2HVS2.*

Cho mảng a là mảng hai chiều kích thước M x N. **Hãy tìm hình vuông lớn nhất có tổng bằng S**.

Tương tự bài 1, đến khi tính được Q ta cập nhật lại **res** với k: res = max(res,k);

Kiểm soát trường hợp vô nghiệm để ghi ra -1.

Sau đó ta đưa **res** ra.

```
    Z:=0; res:=low(longint);
    For(k=1,min(m,n))
        For(I,1,m)
          For(j,1,n)
            Begin
                Q:= s[i+k-1,j+k-1] – S[i-1,j+k-1] – s[i+k-1,j-1] + s[i-1,j-1];
                 If Q=s then
                   Begin
```

```
                        res:=max(res,k);
                        z:=1;
                    end;
                End;
            If z=0 then write(-1) else write(res);
```

```
uses math;
var
    n, i, m, s, j, t2, k, res: longint;
    t, a: array[0..107,0..107] of longint;
    f: boolean = FALSE;
const
    fi = 'ARR2HVS2.INP';
    fo = 'ARR2HVS2.OUT';
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    read(m,n,s);
    for i:=1 to m do
        for j:=1 to n do
            begin
                read(a[i,j]);
                t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];
            end;
    for k:=1 to min(m,n) do
        for i:=1 to m-k+1 do
            for j:=1 to n-k+1 do
                begin
                    t2 := t[i+k-1,j+k-1] - t[i-1,j+k-1] - t[i+k-1,j-1] + t[i-1,j-1];
                    if t2 = s then
                        begin
                            res := max(res,k);
                            f := TRUE;
                        end;
                end;
    if f = FALSE then write(-1) else write(res);
    close(input); close(output);
end.
```

3) ARR2HCN1.*

Cho mảng a là mảng hai chiều kích thước M x N. ***Hãy tìm tất cả các hình chữ nhật có tổng bằng S***.

- **Bước 1:** Chuẩn bị mảng cộng dồn S[I,j] với S[I,j] là tổng từ ô (1,1) đến ô (I,j). // s[I,j] gọi là tổng tích luỹ.

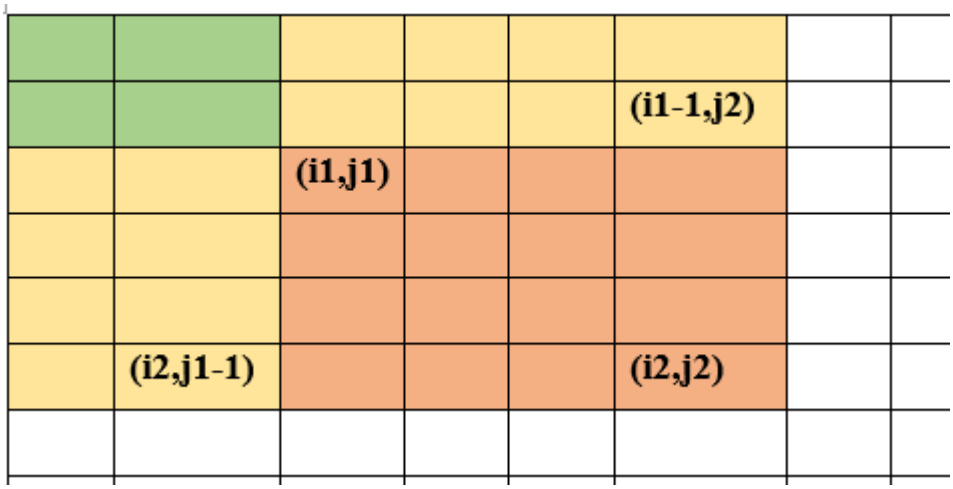
Ta có s[0,0] = 0;

For(i,1,m)

For (j,1,n)

S[I,j] = s[i-1,j] + s[i,j-1] – s[i-1,j-1] + a[i,j];

- **Bước 2:** Duyệt qua tất cả các hình chữ nhật, ta tính tổng các ô trong hình chữ nhật.
-




```
Z:=0;
For(i1,1,m)
  For(j1,1,n)
     For(i2,i1,m)
       For(j2,j1,n)
            Begin
                  Q:= s[i2,j2]-s[i2,j1-1]-s[i1-1,j2] + s[i1-1,j1-1];
                  If q = s then
                        Begin
                              Write(i1,#32,j1,#32,i2,#32,j2);
                               Z:= 1;
                         End;
            End;
If z=0 then write(-1);
```

```
uses math;
var
      m, n, s, i, j, x, y: longint;
```

```
    a, t: array[0..57,0..57] of longint;
    f: boolean = FALSE;
const
    fi = 'ARR2HCN1.INP';
    fo = 'ARR2HCN1.OUT';
function rect(x1,y1,x2,y2: longint): longint;
var
    o, p: longint;
    res2 : longint;
begin
    res2 := 0;
    for o:=x1 to y1 do
        for p:=x2 to y2 do
            inc(res2,a[o,p]);
    exit(res2);
end;
function rect2(x1,x2,y1,y2: longint): longint;
var
    o, p: longint;
    t2 : longint;
begin
    t2 := 0;
    t2 := t[y1,y2] - t[x1-1,y2] - t[y1,x2-1] + t[x1-1,x2-1];
    exit(t2);
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    read(m,n,s);
    for i:=1 to m do
        for j:=1 to n do
            begin
                read(a[i,j]);
                t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];
            end;
        {
        // SUB1
    for i:=1 to m do
        for j := 1 to n do
            for x := i to m do
```

```
                    for y := j to n do
                                        if (rect(i,x,j,y) = s) then
                                    begin
                                        writeln(i,#32,j,#32,x,#32,y);
                                        f := TRUE;
                                    end;
            }
      for i:=1 to m do
            for j := 1 to n do
                  for x := i to m do
                        for y := j to n do
                              if (rect2(i,j,x,y) = s) then
                                    begin
                                        writeln(i,#32,j,#32,x,#32,y);
                                        f := TRUE;
                                    end;

      if f = FALSE then write(-1);
      close(input); close(output);
end.
```

4) ARR2HCN2.*

Cho mảng a là mảng hai chiều kích thước M x N. **Hãy đếm số lượng hình chữ nhật có tổng chia hết cho K.**

Tương tự bài 3, ta tăng đếm lên khi ***Q mod K = 0;***

```
Z:=0; res:=0;
For(i1,1,m)
  For(j1,1,n)
     For(i2,i1,m)
       For(j2,j1,n)
            Begin
                  Q:= s[i2,j2]-s[i2,j1-1]-s[i1-1,j2] + s[i1-1,j1-1];
                  If q mod k=0 then
                        Begin
                              Inc(res);
                               Z:= 1;
                         End;
              End;
          If z=0 then write(-1) else write(res);
```

```
uses math;
var
	m, n, s, i, j, x, y: longint;
	a, t: array[0..57,0..57] of longint;
	d: longint = 0;
	f: boolean = FALSE; d2: int64;
const
	fi = 'ARR2HCN2.INP';
	fo = 'ARR2HCN2.OUT';
function rect(x1,x2,y1,y2: longint): longint;
var   t2 : longint;
begin
	t2 := 0;
	t2 := t[y1,y2] - t[x1-1,y2] - t[y1,x2-1] + t[x1-1,x2-1];
	exit(t2);
end;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	read(m,n,s);
	for i:=1 to m do
		for j:=1 to n do
			begin
				read(a[i,j]);
				a[i,j] := a[i,j] mod s;
				t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];
			end;
	for i:=1 to m do
		for j := 1 to n do
			for x := i to m do
				for y := j to n do
					if (rect(i,j,x,y) mod s = 0) then
							begin
								inc(d); f := TRUE;
							end;
	if f = TRUE then writeln(d) else writeln(-1);
	close(input); close(output);
end.
```

Ngày 5/7

1) ARR2REC1.*

Cho ma trận kích thước MxN mà các phần tử có giá trị bằng 0 hoặc 1.

Hãy tìm hình chữ nhật chứa toàn số 1 có diện tích lớn nhất.

Đưa ra diện tích lớn nhất đó.

Thuật toán:

Điều kiện một hình chữ nhật chứa toàn số 1 là: tổng các ô trong HCN bằng diện tích của HCN đó.

Ta gọi t[i,j] là tổng các ô trong HCN có góc trái trên là (1,1) góc phải dưới là (i,j).

t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];

Duyệt qua tất các các hình chữ nhật con, nếu tổng các ô trong HCN con bằng điện tích HCN đó thì cập nhật lại res – lưu diện tích HCN thỏa mãn lớn nhất.

```
uses math;
var
    m, n, s, i, j, x, y, t3: longint;
    a, t: array[-7..107,-7..107] of longint;
    d: longint = 0;
    res: longint = low(longint);
const
    fi = 'ARR2RECT.INP';
    fo = 'ARR2RECT.OUT';
function rect(x1,x2,y1,y2: longint): longint;
var
    t2 : longint;
begin
    t2 := 0;
    t2 := t[y1,y2] - t[x1-1,y2] - t[y1,x2-1] + t[x1-1,x2-1];
    exit(t2);
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    read(m,n);
    for i:=1 to m do
        for j:=1 to n do
            begin
                read(a[i,j]);
                t[i,j] := t[i-1,j] + t[i,j-1] - t[i-1,j-1] + a[i,j];
            end;
    for i:=1 to m do
```

```
            for j := 1 to n do
                for x := i to m do
                    for y := j to n do begin
                        t3 := rect(i,j,x,y);
                        if t3 = ((x-i+1) * (y-j+1)) then
                                begin
                                    res := max(res,t3);
                                end;
                    end;
        write(res);
        close(input); close(output);
end.
```

2) ARR2REC2.*

Cho ma trận MxN. Quân xe trong cờ vua hoặc cờ tướng có đặc điểm di chuyển và ăn quân theo hình dấu (+).

Người ta đặt quân xe ở ô (i, j) thì số điểm thu được bằng tổng giá trị các phần tử ở hàng i và tổng giá trị các phần tử ở cột j và a[i,j].

Hãy tìm vị trí đặt quân xe để đạt tổng điểm lớn nhất. Đưa ra tổng điểm lớn nhất đó.

Thuật toán:

Gọi h[i] là tổng các ô ở hàng thứ i, c[j] là tổng các ô ở cột j.

Ta chuẩn bị trước mảng h[i] và c[j] với chi phí $O(N^2)$.

Khi quân xe ở ô (i,j) thì điểm đạt được là s = h[i]+c[j] – a[i,j];

Ta phải trừ đi a[i,j] do a[i,j] đã được tính hai lần.

Từ đó ta đi tìm smax là tổng điểm lớn nhất đạt được.

Duyệt từ đầu đến cuối nếu S = Smax thì ta đưa (i,j) ra.

```
uses math;
const
        fi = 'ARR2REC2.INP';
        fo = 'ARR2REC2.OUT';
var
        n, i, m, j: longint;
        th, tc: array[-7..107] of longint;
        a: array[-7..107,-7..107] of longint;
        res: longint = low(longint);
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(m,n);
        for i:=1 to m do
                for j:=1 to n do
```

```
                begin
                    read(a[i,j]);
                    inc(th[i],a[i,j]);
                    inc(tc[j],a[i,j]);
                end;
    for i:=1 to m do
        for j:=1 to n do
            res := max(res,th[i]+tc[j]-a[i,j]);
    write(res);
    close(input); close(output);
end.
```

3) ARR2REC3*. Quân mã

Cho ma trận kích thước M, N. Tìm vị trí đặt quân Mã để kiểm soát được các ô có tổng lớn nhất (bao gồm cả ô con Mã đang đứng).

Thuật toán:

Đặt quân Mã ở ô (i,j) thì nó có thể kiểm soát được các ô (x) như bảng dưới đây:

	x		x		
x				x	
		(i,j)			
x				x	
	x		x		

Gọi T là tổng điểm có được khi đặt quân Mã ở ô (i,j), thì:

T = a[i-2,j-1] + a[i-2,j+1] + a[i-1,j+2] + a[i+1,j+2] + a[i+2,j+1] + a[i+2,j-1] + a[i+1,j-2] + a[i-1,j-2] + a[i-2,j-1] + a[i,j];

Kết quả res = max (res,T); với Res ban đầu là giá trị vô cùng bé.

Ta lưu ý các vị trí ngoài bảng khi quân mã đặt gần biên. Ta khai báo thêm các "chỉ số âm" cho mảng hai chiều a: a: array[-7..107,-7..107] of longint;

Trong ngôn ngữ lập trình C++, ta dùng ***kĩ thuật dịch bảng*** ngay từ lúc đọc dữ liệu

Cin >> a[i+2,j+2];

```
uses math;
const
    fi = 'ARR2REC3.INP';
    fo = 'ARR2REC3.OUT';
var
    n, i, m, j, t, neg: longint;
    a: array[-7..107,-7..107] of longint;
    res: longint = low(longint);
```

```
begin
	neg := -250000000; // (-2^31-8*10^4)/8
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(m,n);
	for i:=1 to m do
		for j:=1 to n do
			begin
				read(a[i,j]);
			end;
	for i:=1 to m do
		for j:=1 to n do
			begin
				t := a[i-2,j-1] + a[i-2,j+1] + a[i-1,j+2] + a[i+1,j+2] +
a[i+2,j+1] + a[i+2,j-1] + a[i+1,j-2] + a[i-1,j-2] + a[i-2,j-1] + a[i,j];
				res := max(res,t);
			end;
	write(res);
	close(input); close(output);
end.
```

4) STR1NUM.*

Nhập xâu S. Tách các số trong xâu ra.

Thuật toán:

Duyệt từ đầu xâu đến cuối xâu, nếu s[i] là kí tự số thì ta đẩy sang xâu tạm, ngược lại, ta đẩy xâu tạm vào mảng C có K phần tử.

Duyệt từ 1 tới C, đưa mảng C ra.

```
uses math;
const
	fi = 'STR1NUM.INP';
	fo = 'STR1NUM.OUT';
var
	n, i: longint;
	s, res: string;
	f: boolean;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(s); i:= 1;
	while i <= length(s) do
```

```
            begin
                f:=false;
                inc(i);
                while (s[i] in ['0'..'9']) and (i <= length(s)) do
                    begin
                    f := true; write(s[i]); inc(i);
                    end;
                if f then write(#32);
            end;
        close(input); close(output);
end.
```

5) STR2NUM.*

Nhập xâu S. Tìm số lớn nhất trong xâu.

Thuật toán:

Duyệt từ đầu xâu đến cuối xâu, ta tách các chữ số trong xâu lưu vào mảng C có K phần tử. Khi lưu các xâu số vào mảng C, ta nên đổi nó sang dạng số.

Duyệt trên mảng C tìm giá trị lớn nhất res và đưa res ra.

```
uses math;
const
    fi = 'STR2NUM.INP';
    fo = 'STR2NUM.OUT';
var
    n, i, t: longint;
    s, ts, res: string;
    f: boolean;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(s);
    res := '';
    while i <= length(s) do
        begin
            ts := '';
            inc(i);
            while (s[i] in ['0'..'9']) and (i <= length(s)) do
                begin
                    ts := ts + s[i];
                    inc(i);
                end;
```

```
                if length(ts) > length(res) then res := ts
                else if length(ts) = length(res) then if ts >= res then res := ts;
            end;
        write(res);
        close(input); close(output);
end.
```

6) STR22NUM .* Nén xâu

Cho xâu S chỉ gồm các chữ cái từ ‘a’ đến ‘z’. Hãy nén xâu S. Đưa xâu nén ra.

Ví dụ: S = ‘aaabbccccd’ → Snen = ’a3b3c4d1’.

Thuật toán:

Duyệt từ đầu xâu đến cuối xâu, đến phần tử thứ i nếu s[i]=s[i+1] thì ta tăng đếm lên, ngược lại ta đẩy s[i] vào xâu sres và đẩy đếm vào mảng c[k].

Duyệt từ 1 tới K, ta đưa (sres[i],c[i]) ra.

```
uses math;
const
        fi = 'STR3NUM.INP';        fo = 'STR3NUM.OUT';
var
        n, i, d: longint;        s, snen: ansistring;
        ds: string;
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(s);
        while i < length(s) do
                begin
                        inc(i);
                        snen := snen + s[i]; d := 1;
                        while (i < length(s)) and (s[i] = s[i+1]) do
                                begin
                                        inc(d);
                                        inc(i);
                                end;
                        str(d,ds);
                        snen := snen + ds;
                end;
        write(snen);
        close(input); close(output);
end.
```

7) **STR3NUM.* Giải nén xâu.**

Nhập xâu Snen. Hãy giải nén xâu.

Ví dụ: Snen = ’a3b3c4d1’ → S = ‘aaabbccccd’

Thuật toán:

Vận dụng thuật toán tách các chữ số từ xâu ra. Ta làm như sau:

Duyệt từ đầu đến cuối xâu, nếu s[i] là kí tự số, ta đẩy vào temp, ngược lại ta đổi temp sang kiểu số nguyên, lưu vào mảng C[k] và lưu s[i] vào xâu sres.

Res = “”;

Duyệt từ 1 tới K

{

Duyệt từ 1 tới c[i] tính res = res + sres[i];

}

Đưa xâu res ra.

```
uses math;
var
    s, snen, t, t2: string;
    i, j, t3: longint;
const
    fi = 'STR4NUM.INP';
    fo = 'STR4NUM.OUT';
begin
    assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
    readln(snen);
    i := 1;
    while i <= length(snen) do
        begin
            t := ''; t2 := snen[i];
            inc(i);
            while (snen[i] in ['0'..'9']) and (i <= length(snen)) do
                begin
                    t := t + snen[i];
                    inc(i);
                end;
            val(t,t3);
            for j := 1 to t3 do s := s + t2;
        end;
    write(s);
    close(input); close(output);
end.
```

Ngày 6/7

1) DQBIT01.*

Cho N. Hãy liệt kê các dãy nhị phân có độ dài N.

Thuật toán:

0	0	0	0
		1	0

Với mỗi dãy nhị phân thu được ta gọi đó là *một cấu hình*. Các cấu hình lần lượt được sinh ra theo thứ tự từ điển, gọi là thứ tự của cấu hình.

Phương pháp sinh thực hiện thử điền vào ô vị trí i giá trị 0 hoặc 1. Nếu điền đến đến ô thứ N, chứng tỏ ta đã tìm được một cấu hình. Ta đưa cấu hình đó ra. Rồi lại tìm cấu hình kế tiếp.

Trong chương trình chính ta chỉ cần gọi: sinh(1);

```
const
      fi = 'DQBIT01.INP';
      fo = 'DQBIT01.OUT';
var
      n, i: longint;
      a: array[0..27] of byte;
procedure print();
begin
      for i:=1 to n do
            write(a[i],#32);
      writeln();
end;
procedure pps(x: longint);
var j: byte;
begin
      for j:=0 to 1 do
            begin
                  a[x] := j;
                  if x = n then print()
                  else pps(x+1);
            end;
end;
begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
      readln(n);
      pps(1);
      close(input); close(output);
end.
```

2) DQBIT02.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có số lượng số 0 bằng số lượng số 1.

Thuật toán:

Sử dụng phương pháp sinh dãy nhị phân có độ dài N.

Với mỗi cấu hình thu được ta đếm số lượng bit 0 và số lượng bit 1.

Nếu số lượng các bít này bằng nhau thì ta đưa cấu hình đó ra.

Độ phức tạp: $O(2^N)$.

Lưu ý xử lí trường hợp vô nghiệm ghi ra -1.

```
uses math;
const
	fi = 'DQBIT02.INP';
	fo = 'DQBIT02.OUT';
var
	n, i: longint;
	a: array[0..27] of byte;
procedure print();
var
	d1: longint;
begin
	d1 := 0;
	for i:=1 to n do if a[i] = 0 then inc(d1);
	if  2*d1 = n then
	    begin
	        for i:=1 to n do write(a[i],#32);
	        writeln();
	     end;
end;
procedure pps(x: longint);
var j: byte;
begin
	for j:=0 to 1 do
		begin
			a[x] := j;
			if x = n then print()
			else pps(x+1);
		end;
end;
begin
	assign(input,fi); reset(input);
```

```
        assign(output,fo); rewrite(output);
        readln(n);
        if n mod 2 <> 0 then write(-1) else pps(1);
        close(input); close(output);
end.
```

3) DQBIT03.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N không có 3 số 1 đứng cạnh nhau.

Thuật toán:

Sử dụng phương pháp sinh dãy nhị phân có độ dài N.

Với mỗi cấu hình thu được ta kiểm ta xem cấu hình đó có a[i]=a[i+1]=a[i+2]=1.

Nếu cấu hình đó không thỏa mãn điều kiện trên thì ta đưa cấu hình đó ra và tăng biến số lượng nghiệm lên.

Độ phức tạp: $O(2^N)$.

Lưu ý xử lí trường hợp vô nghiệm ghi ra -1.

```
uses math;
const
    fi = 'DQBIT03.INP';
    fo = 'DQBIT03.OUT';
var
    n, i: longint;
    a: array[0..27] of byte;
procedure print();
var
    k: longint;
    f: boolean;
begin
    f := true;
    for i:=1 to n-2 do
        begin
            if (a[i] = 1) and (a[i] = a[i+1]) and (a[i+1] = a[i+2]) then
                f := false;
        end;
    if f then
        begin
            for k:=1 to n do write(a[k],#32);
            writeln();
        end;
end;
procedure pps(x: longint);
var j: byte;
```

```
begin
    for j:=0 to 1 do
        begin
            a[x] := j;
            if x = n then print()
            else pps(x+1);
        end;
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    pps(1);
    close(input); close(output);
end.
```

4) DQBIT04.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có đúng K số 1.

Thuật toán:

Sử dụng phương pháp sinh dãy nhị phân có độ dài N.

Với mỗi cấu hình thu được ta đếm số lượng bit 1.

Nếu số lượng bít 1 bằng K thì ta đưa cấu hình đó ra.

Độ phức tạp: $O(2^N)$.

Lưu ý xử lí trường hợp vô nghiệm ghi ra -1.

```
uses math;
const
    fi = 'DQBIT04.INP';
    fo = 'DQBIT04.OUT';
var
    n, i, k: longint;
    a: array[0..27] of byte;
procedure print();
var
    d: longint;
begin
    d := 0;
    for i:=1 to n do if a[i] = 1 then inc(d);
    if d = k then
        begin
            for i:=1 to n do write(a[i],#32);
```

```
                writeln();
            end;
end;
procedure pps(x: longint);
var j: byte;
begin
    for j:=0 to 1 do
        begin
            a[x] := j;
            if x = n then print()
            else pps(x+1);
        end;
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n,k);
    pps(1);
    close(input); close(output);
end.
```

5) DQBIT05.*

Cho N. Hãy liệt kê các dãy nhị phân độ dài N có không quá K số 1.

Thuật toán: Tương tự bài 6) ta làm như sau

Sử dụng phương pháp sinh dãy nhị phân có độ dài N.

Với mỗi cấu hình thu được ta đếm số lượng bit 1.

Nếu số lượng bít 1 nhỏ hơn hoặc bằng K thì ta đưa cấu hình đó ra.

Độ phức tạp: $O(2^N)$.

Lưu ý xử lí trường hợp vô nghiệm ghi ra -1.

```
uses math;
const
    fi = 'DQBIT05.INP';
    fo = 'DQBIT05.OUT';
var
    n, i, k: longint;
    a: array[0..27] of byte;
procedure print();
var
    d: longint;
begin
```

```
        d := 0;
        for i:=1 to n do if a[i] = 1 then inc(d);
        if d <= k then
                begin
                        for i:=1 to n do write(a[i],#32);
                        writeln();
                end;
end;
procedure pps(x: longint);
var j: byte;
begin
        for j:=0 to 1 do
                begin
                        a[x] := j;
                        if x = n then print()
                        else pps(x+1);
                end;
end;
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(n,k);
        pps(1);
        close(input); close(output);
end.
```

Ngày 7/7

1) TRIANGLE.*

Cho toạ độ các điểm A(x1,y1), B(x2,y2), C(x3,y3). Kiểm tra A, B, C có là ba đỉnh của một tam giác? Nếu có hãy diện tích tam giác ABC, ngược lại ghi ra -1.

Thuật toán:

Ta viết hàm tính độ dài đoạn thẳng khi biết tọa độ hai điểm A(x1,y1), B(x2,y2):

$$d_1 = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Tương tự ta tính được độ dài các đoạn thẳng AC và BC.

Điều kiện A, B, C là ba đỉnh của một tam giác là: tổng hai cạnh lớn hơn cạnh còn lại.

Nếu thỏa mãn, ta tính diện tích của tam giác ABC theo công thức Hê-rông:

$$p = \frac{d1 + d2 + d3}{2};$$

$$S = \sqrt{p.(p - d1).(p - d2).(p - d3)}$$

```
 uses math;
var
      x1, x2, x3, y1, y2, y3: longint;
      p, s, a, b, c: real;
const
      fi = 'TRIANGLE.INP';
      fo = 'TRIANGLE.OUT';
begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
      read(x1,y1,x2,y2,x3,y3);
      a := sqrt(sqr(x2-x3)+sqr(y2-y3));
      b := sqrt(sqr(x1-x3)+sqr(y1-y3));
      c := sqrt(sqr(x2-x1)+sqr(y2-y1));
      if (a + b > c) and (a + c > b) or (b + c > a) then
            begin
                  p := (a+b+c)/2;
                  s := sqrt(p*(p-a)*(p-b)*(p-c));
                  write(s:0:3);
            end
      else write(-1);
      close(input); close(output);
end.
```

2) RECTAB1.*

Cho hình chữ nhật có kích thước a x b, được chia thành lưới ô vuông đơn vị. Gọi X là số ô đường viền, Y là số ô trong lõi. Hãy tìm X, Y.

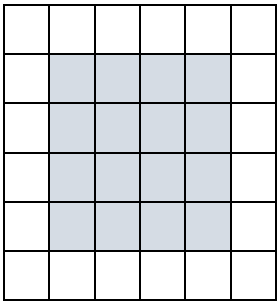
Thuật toán:

Vẽ hình bên dưới để tìm công thức tính:

Số ô trên đường viền: X = 2a+2(b-2)

Số ô trong lõi: Y = 2(a-2) + 2*(b-2).

Lưu ý: Do giới hạn của a, b là kiểu longint nên X và Y lớn. Ta phải chọn kiểu dữ liệu int64.



```
uses math;
var
	a, b: longint;
	x, y: int64;
const
	fi = 'RECTAB1.INP';
	fo = 'RECTAB1.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	read(a, b);
	x := 2*(a+b)-4;
	y := (a*b)-x;
	write(x,#32,y);
	close(input); close(output);
end.
```

3) CIRCLE.*

Cho điểm M (x1, y1) và đường tròn tâm I(x0, y0) bán kính R. Hãy xét vị trí tương đối của điểm M với đường tròn. Nếu M nằm trong hoặc trên đường tròn ghi ra 1, ngược lại ghi ra 0.

Thuật toán:

Gọi d là khoảng các từ điểm M tới tâm I của đường tròn. Khi đó ta có:

$$d = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$$

Điểm M nằm trong hoặc trên đường tròn khi khoảng các từ M tới tâm I nhỏ hơn hoặc bằng bán kính R ($d \leq R$).

```
uses math;
var
```

```
        x1, y1, x0, y0, r: longint;
        distance: real;
const
        fi = 'CIRCLE.INP';
        fo = 'CIRCLE.OUT';
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(x1,y1,x0,y0,r);
        distance := sqrt(sqr(x0-x1)+sqr(y0-y1));
        if distance <= r then write(1) else write(0);
        close(input); close(output);
end.
```

4) RECTM2.*

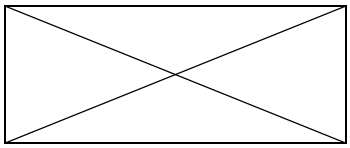
Cho điểm M(x1, y1) và hình chữ nhật có góc trái trên (x2, y2) và góc phải dưới (x3, y3).

Kiểm tra M nằm trong hay ngoài hình chữ nhật. Nếu nằm trong và trên cạnh thì ghi ra 1, ngược lại, M nằm ngoài hình chữ nhật ghi ra 0.

Thuật toán:

Vẽ hình: Cho điểm M trong HCN. Nối MA, MB, MC, MD.

Ta nhận thấy bài này có hai cách làm:



Cách 1: Ta tính diện tích các tam giác MAB, MBC, MCD, MDA lưu vào s1, s2, s3, s4.

Nếu s1+s2+s3+s4 = S_{HCN} thì

M nằm trong tam giác, ta ghi ra 1

Ngược lại

ghi ra 0.

Cách 2: Ta có góc trái trên A(x1,y1), góc phải dưới C(x2,y2). Gọi điểm M(x0,y0) thì Để M nằm trong HCN trên thì

$$\begin{cases} x1 \le x0 \le x2 \\ y2 \le y0 \le y1 \end{cases}$$

```
uses math;
var
      x1, y1, x2, y2, x3, y3: longint;
const
      fi = 'RECTM2.INP';
      fo = 'RECTM2.OUT';
begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
```

```
    readln(x1,y1,x2,y2,x3,y3);
    if (x1 <= x3) and (y1 <= y2) then write(1) else write(0);
    close(input); close(output);
end.
```

5) PERFECTN.*

Số H được gọi là *số hoàn thiện* nếu tổng tất cả các ước của H gấp đôi H.

Ví dụ: Số 6 là số hoàn thiện, 6 chia hết cho 1, 2, 3, 6 và tổng ước = 12 = 2x6.

Kiểm tra H có hoàn thiện ghi ra 1 ngược lại ghi 0.

Thuật toán:

Sub1: Tính tổng các ước bằng cách for(i,1,N) nếu i là ước của H thì đẩy i vào tổng S.

Nếu tổng S = 2.N thì đưa ra 1 ngược lại đưa ra 0.

Độ phức tạp thuật toán: O(N). Làm theo cách này bạn đạt không quá 50% điểm.

Sub2: Các bài tập liên quan tới ước ta vận dụng for(i,1,[$\sqrt{N}$]) nếu i là ước của N thì ta thực hiện:

+ Đẩy i vào tổng S.

+ Nếu **i** khác **N div i** thì đẩy **N div i** vào tổng.

Nếu tổng = 2. N thì đưa 1 ra ngược lại ghi ra 0.

Ta nên tổ chức thành hàm để kiểm tra tính hoàn thiện của một số nguyên.

```
uses math;
var
    n, i: longint;
    f: boolean = false;
const
    fi = 'PERFECTN.INP';
    fo = 'PERFECTN.OUT';
function perfect(x: longint): boolean;
var
    j: longint;
    res: int64 = 0;
begin
    for j:=1 to trunc(sqrt(x)) do
        begin
            if x mod j = 0 then
                begin
                    inc(res,j);
                    if j <> x div j then inc(res,x div j);
                end;
        end;
    if res = 2*x then exit(true) else exit(false);
```

```
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);   readln(n);
    for i:=1 to n do
        if perfect(i) then
            begin
                write(i,#32); f := true;
            end;
    if f = false then write(-1);
    close(input); close(output);
end.
```

6) FIBPRIME.*

Dãy số Fibonacci có đặc điểm: F[1] = 1, F[2] = 1, F[i] = F[i-1] + F[i-2] với i ≥ 3.

Cho số nguyên dương M và N. Hãy tìm các số vừa là số Fibonacci, vừa là số nguyên tố nằm trên đoạn [M, N].

Thuật toán:

Sub1: Tạo mảng F lưu các số Fibonacci nhỏ hơn hoặc bằng N.

Viết hàm kiểm tra tính nguyên tố của số nguyên U.

Duyệt từ đầu mảng F đến cuối mảng F,

nếu M ≤ F[i] ≤ N thì

nếu nguyên tố của F[i] = true thì đưa F[i] ra;

Độ phức tạp thuật toán: O(100.$\sqrt{N}$)

Sub2: Ta sử dụng sàng nguyên tố Eratosthenes để tạo sẵn mảng ngto[i] = true nếu i nguyên tố, ngto[i]= false nếu i không nguyên tố.

Duyệt từ đầu mảng F đến cuối mảng F,

nếu M ≤ F[i] ≤ N thì

nếu ngto[F[i]] = true thì đưa F[i] ra;

Độ phức tạp thuật toán: O(100 + $\sqrt{N}$).

Lưu ý: Số Fibonacci có giá trị lớn, nên phải chọn kiểu dữ liệu int64.

```
uses math;
var
    m, x, i, maxC: longint;
    f: array[-7..107] of longint;
    e, f2: array[-7..2000007] of boolean;
    p: boolean = false;
const
    fi = 'FIBPRIME.INP';
    fo = 'FIBPRIME.OUT';
```

```
procedure snt(n: longint);
var
	i, j: longint;
begin
	fillchar(e,sizeof(e),TRUE);
	maxC := trunc(sqrt(n));
	i := 2;
	while i <= maxC do
		begin
			while e[i] = FALSE do inc(i);
			for j:=2 to n div i do
				e[i*j] := FALSE;
			inc(i);
		end;
end;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	read(m,x);
	f[1] := 1;
	f[2] := 1;
	snt(x);
	i := 2;
	repeat
		inc(i);
		f[i] := f[i-1] + f[i-2];
		f2[f[i]] := true;
	until f[i] > x;
	for i:=m to x do
		if f2[i] and e[i] then
			begin
				p := true;
				write(i,#32);
			end;
	if p = false then write(-1);
	close(input); close(output);
end.
```

7) NBEAUTY.*

Số đẹp là số có tổng các chữ số chia hết cho 9. Như vậy, các số 9, 18, 27, 36,... là các số đẹp.

Cho số nguyên dương N. Tìm số đẹp thứ N.

Thuật toán:

Nhận xét: Số có tổng các chữ số chia hết cho 9 thì số đó là bội của 9. Các bội của 9 lần lượt là 9, 18, 27, 36….

Số đẹp thứ i có giá trị là: 9*i.

```
uses math;
var
	n, i: longint;
const
	fi = 'NBEAUTY.INP';
	fo = 'NBEAUTY.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	write(9*n);
	close(input); close(output);
end.
```

8) NPALIN.*

Số Palidrom là số đối xứng, nghĩa là đọc từ trái sang phải hay từ phải sang trái ta đều được một số. Ví dụ: Các số palidrom 11, 121, 1331, 2222, 1556551,... Các số có một chữ số cũng là số Palidrom.

Cho số nguyên dương N, hãy tìm số Palidrom thứ N.

Thuật toán:

Tổ chức chương trình thành hàm palin(u: longint): boolean; để kiểm tra tính đối xứng của số nguyên dương u.

Ta lần lượt tìm các số Palidrom. Đến số thứ N thì đưa số đó ra.

```
uses math;
var
	n, i: longint;
	x: int64;
const
	fi = 'NPALIN.INP';
	fo = 'NPALIN.OUT';
function xaudao(t: int64): int64;
var
	s, s2: string;
	j: longint;
	t2: int64;
```

```
begin
	s2 := '';
	str(t,s);
	for j:=length(s) downto 1 do s2 := s2 + s[j];
	val(s2,t2);
	exit(t2);
end;
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	readln(n);
	repeat
		inc(x);
		if x = xaudao(x) then inc(i);
	until i = n;
	write(x);
	close(input); close(output);
end.
```

9) LASTNUM.*

Cho số nguyên dương a và N. Hãy tìm chữ số tận cùng của a^N.

Thuật toán:

Nhận xét: Để tìm N chữ số tận cùng của số P, ta chỉ cần giữ phần dư của P mod 10^N.

Vậy, ta sẽ tổ chức chương trình gồm hàm amub(a,b: longint):int64;

Kết quả của bài toán là amub(a,n) mod L; // với L = amub(10,N);

Tuy nhiên, khi dữ liệu lớn, chương trình bị tràn kiểu dữ liệu. Để xử lí trường hợp này, ta khi tính amub(a,n) tới đâu ta chia dư cho L tới đó.

```
uses math;
var
	a, n, i: longint;
	res: longint = 1;
const
	fi = 'LASTNUM.INP';
	fo = 'LASTNUM.OUT';
begin
	assign(input,fi); reset(input);
	assign(output,fo); rewrite(output);
	read(a,n);
	for i:=1 to n do res := (res * a) mod 10;
```

```
        write(res);
        close(input); close(output);
end.
```

PHẦN III. QUAY LUI

10) DQBIT06

Cho N, K và dãy a1, a2,…,aN. Hãy chọn nhiều nhất các phần tử có tổng bằng K.

Thuật toán:

Áp dụng phương pháp sinh ta có: Với mỗi cấu hình lưu vào mảng b. Ta thực hiện duyệt từ 1 tới N, nếu b[i] = 1 thì ta chọn a[i] và đẩy vào tổng.

Nếu tổng = k thì cập nhật lại số lượng phần tử được chọn.

Lưu ý: Ta có thể thêm ***nhánh cận***: khi đẩy a[i] vào tổng, nếu tổng > k thì exit luôn, không cần duyệt đến cuối dãy.

```
uses math;
const
	fi = 'DQBIT06.INP';
	fo = 'DQBIT06.OUT';
var
	n, i, k: longint;
	a: array[-7..1007] of longint;
	b: array[-7..1007] of byte;
	res: longint = low(longint);
	f: boolean = false;
procedure print();
var
	d, d2: longint;
begin
	d := 0;
	d2 := 0;
	for i:=1 to n do
		if b[i] = 1 then
			begin
				inc(d,a[i]);
				inc(d2);
			end;
	if d = k then
		begin
			f := true;
			res := max(res,d2);
		end;
end;
procedure pps(x: longint);
var j: byte;
begin
	for j:=0 to 1 do
```

```
            begin
                b[x] := j;
                if x = n then print()
                else pps(x+1);
            end;
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n, k);
    for i:=1 to n do read(a[i]);
    pps(1);
    if f then write(res) else write(-1);
    close(input); close(output);
end.
```

11) DQBIT07

Cho N và dãy a1, a2,…,aN. Hãy liệt kê các cách chia các phần tử thành hai nhóm có tổng bằng nhau. Nếu không có cách chia thì ghi ra -1.

Thuật toán:

Áp dụng phương pháp sinh. Với mỗi cấu hình thu được ta thực hiện tính tổng các phần tử có b[i]=1. Nếu tổng = S/2 thì ta đưa cấu hình đó ra.

Lưu ý: Kĩ thuật ghi nhận có nghiệm bằng cách sử dụng biến p. Ban đầu khởi tạo p = 0; mỗi khi phát hiện một nghiệm ta đổi giá trị p = 1; Sau khi duyệt xong, nếu p = 0 thì chứng tỏ vô nghiệm, ta ghi ra -1. Đây là kĩ năng cần thiết để phát hiện trường hợp vô nghiệm cần nhớ.

```
uses math;
const
    fi = 'DQBIT07.INP';
    fo = 'DQBIT07.OUT';
var
    n, i, k: longint;
    a: array[-7..1007] of longint;
    b: array[-7..1007] of byte;
    res: longint = low(longint);
    f: boolean = false;
    t, t2: int64;
procedure printf();
begin
    t2 := 0;
    for i:=1 to n do
```

```
            if b[i] = 0 then inc(t2,a[i]);
        if t2 = t/2 then
            begin
                f := true;
                for i:=1 to n do if b[i] = 0 then write(a[i],#32);
                writeln;
            end;
end;
procedure pps(x: longint);
var j: byte;
begin
    for j:=0 to 1 do
        begin
            b[x] := j;
            if x = n then printf()
            else pps(x+1);
        end;
end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(n);
    for i:=1 to n do begin read(a[i]); inc(t,a[i]); end;
    pps(1);
    if f = false then write(-1);
    close(input); close(output);
end.
```

12) DQBIT08

Cho N và dãy a1, a2,…,aN. Hãy chọn một số các phần tử để có tổng lớn nhất sao cho không được chọn 3 phần tử liên tiếp.

Thuật toán:

Áp dụng phương pháp sinh. Với mỗi cấu hình thu được ta kiểm tra có 3 bít 1 liên tiếp được chọn không? Nếu có thì ta bỏ qua cấu hình này, ngược lại, ta tính tổng các phần tử được chọn và cập nhật lại res.

Lưu ý: Tổng các phần tử có thể lớn, nên ta chọn kiểu dữ liệu int64.

```
uses math;
const
    fi = 'DQBIT08.INP';
    fo = 'DQBIT08.OUT';
```

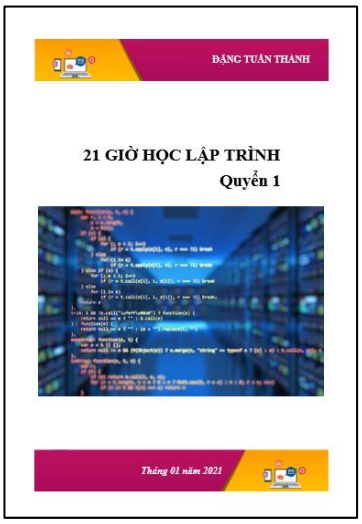
```
var
	n, i: longint;
	a: array[0..27] of byte;
	b: array[-7..27] of longint;
	res: longint = low(longint);
procedure print();
var
	t: longint;
	f: boolean;
begin
	f := true; t := 0;
	for i:=1 to n-2 do
		begin
			if (a[i] = 1) and (a[i] = a[i+1]) and (a[i+1] = a[i+2]) then
		   f := false;
		end;
	if f then
		begin
			for i:=1 to n do if a[i] = 1 then inc(t,b[i]);
			res := max(res,t);
		end;
end;
procedure pps(x: longint);
var j: byte;
begin
	for j:=0 to 1 do
		begin
			a[x] := j;
			if x = n then print()
			else pps(x+1);
		end;
end;
begin
	assign(input,fi); reset(input); assign(output,fo); rewrite(output);
	readln(n);  for i:=1 to n do read(b[i]);
	pps(1);	write(res);
	close(input); close(output);
end.
```

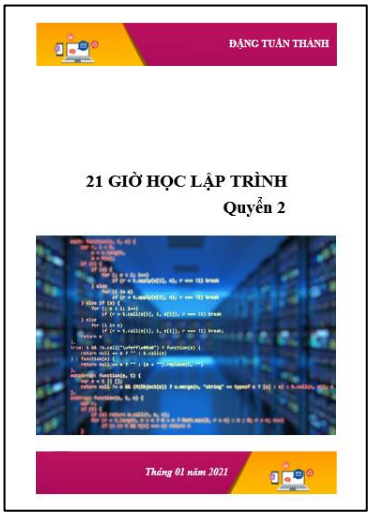
---Hết---

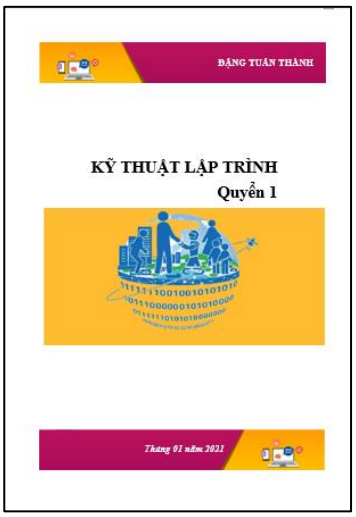
"Mỗi ngày tôi chọn một niềm vui,

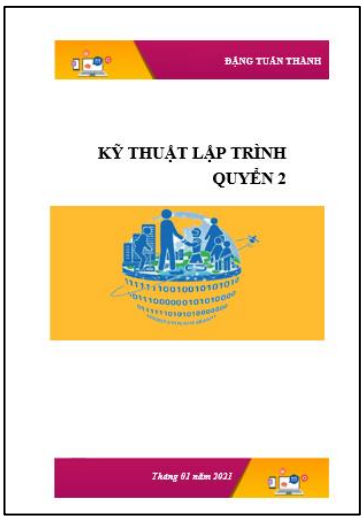
Chọn những bông hoa, chọn những nụ cười…"

Các bạn tìm đọc tài liệu của cùng tác giả:









Hỗ trợ qua zalo: 0854518333.