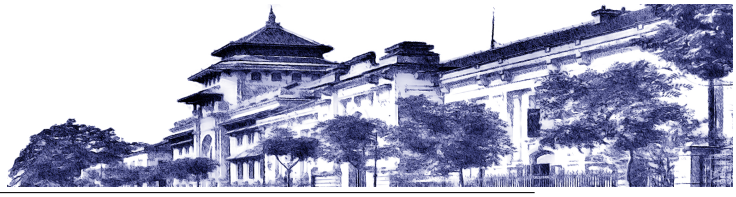
ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
Khoa Vật lý, Bộ môn Tin học Vật lý
P. 308D-T1, 334 Nguyễn Trãi, Thanh Xuân, Hà Nội



Phần 6

Nội dung thực hành

– Lập hàm: *khai báo tên hàm, khai báo các tham số hình thức, cách lấy kết quả ra khỏi hàm*
– Gọi hàm: *truyền tham số thực tế*

Ví dụ

1. Lập hàm tính điểm tổng kết khi biết 3 đầu điểm thành phần

```c
#include <stdio.h>

// Input: 3 so thuc: tx (diem thuong xuyen), gk (diem giua ky), ck (diem cuoi ky)
// --> Tinh diem tong ket theo cong thuc: diem = 20% tx + 20% gk + 60% ck
// Output: 1 so thuc: diem (diem tong ket)
float diem_tongket (float tx, float gk, float ck) {
    float diem = 0.2 * tx + 0.2 * gk + 0.6 * ck;
    return diem;
}

// Input: 3 so thuc: tx (diem thuong xuyen), gk (diem giua ky), ck (diem cuoi ky)
// --> Tinh diem tong ket theo cong thuc: diem = 20% tx + 20% gk + 60% ck
// Output: None (tra ve ket qua thong qua con tro diemtk)
void tinh_diem_tongket (float tx, float gk, float ck, float *diemtk) {
    float diem = 0.2 * tx + 0.2 * gk + 0.6 * ck;
    *diemtk = diem;
}

// Ham chinh
int main() {
    float a, b, c;
    printf("Nhap vao 3 dau diem thanh phan: \n");
    scanf("%f%f%f", &a, &b, &c);
    printf("(Cach 1) Diem tong ket la: %f \n", diem_tongket(a, b, c));

    float DiemTK;
    tinh_diem_tongket(a, b, c, &DiemTK);
    printf("(Cach 2) Diem tong ket la: %f \n", DiemTK);
    return 0;
}
```

Lưu ý:
– Hàm được tạo ra nhằm mục đích đóng gói một đoạn chương trình cụ thể để có thể được tái sử dụng nhiều lần.
– *main()* là một hàm. Ở C89, cách viết *main()* tương đương với cách viết *int main()*. Tuy nhiên ở C99, cách viết *main()* không còn được cho phép và người dùng bắt buộc phải khai báo *int main()*.
– *int main()* nghĩa là hàm chính cần trả về một giá trị nguyên khi kết thúc chương trình và để làm vậy ta thường trả về giá trị 0. Theo "truyền thống", 0 là giá trị chuẩn giá trị cho "sự thực hiện thành công của chương trình".
– Khi lập một hàm, ta cần xác định rõ các đối số đầu vào (input) cũng như các đối số đầu ra (output) của hàm đó.
– Đối với những hàm con ở trong một chương trình lớn và phức tạp, ta nên có chú thích để tránh nhầm lẫn hoặc quên nội dung.

2. Lập hàm kiểm tra một số có phải số nguyên tố

```c
#include <stdio.h>

int so_uoc (int n) { // ham dem so uoc
    int i, dem = 0; // bien cuc bo

    for (i = 1; i <= n; i++)
        if (n % i == 0)
            dem++;

    return dem;
}
```

```c
int main() {
    int n;
    puts("Nhap vao so can kiem tra:");
    scanf("%d", &n);

    if (so_uoc(n) == 2)
        puts("Day la so nguyen to!");
    else
        puts("Day khong la so nguyen to!");

    return 0;
}
```

3. Lập hàm tráo giá trị hai biến (*swap*)

```c
#include <stdio.h>

void trao (int *x, int *y) {
    int tg;
    tg = *x;
    *x = *y;
    *y = tg;
}
```

```c
int main() {
    int a, b;
    puts("Nhap a:"); scanf("%d", &a);
    puts("Nhap b:"); scanf("%d", &b);
    trao(&a, &b);
    printf("a = %d \n", a);
    printf("b = %d \n", b);
    return 0;
}
```

4. Lập hàm sắp xếp dãy tăng dần (*interchange sort*)

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

int *nhap_day (int N) {
    int i, *A;
    A = (int*)calloc(N, sizeof(int));
    for (i = 0; i < N; i++) {
        printf("\n Nhap A[%d] = ", i);
        scanf("%d", A + i);
    }
    return A;
}

void in_day (int *A, int N) {
    int i;
    for (i = 0; i < N; i++)
        printf("%d\t", A[i]);
}

void trao (int *x, int *y) {
    int tg;
    tg = *x;
    *x = *y;
    *y = tg;
}
```

```c
void sap_xep(int *A, int N) {
    int i, j;
    for (i = 0; i < N - 1; i++)
        for (j = i + 1; j < N; j++)
            if (A[i] > A[j]) {
                // int tg;
                // tg = A[i];
                // A[i] = A[j];
                // A[j] = tg;
                trao(&A[i], &A[j]);
            }
}

int main() {
    int n, *a;
    puts("Nhap vao so phan tu:");
    scanf("%d", &n);
    a = nhap_day(n);
    sap_xep(a, n);
    puts("Day sau khi da sap xep la:");
    in_day(a, n);
    free(a);
    return 0;
}
```

Lưu ý:

- Nếu hàm có kiểu là void, lệnh *return* được sử dụng để thoát khỏi hàm ngay lập tức.
- Ở ví dụ 3, hàm *trao* có đầu vào cũng là đầu ra.
- Ở ví dụ 4, hàm *sap_xep* gọi đến hàm *trao* bên trong nó.

5. (Nâng cao) Lập hàm tính N giai thừa

```c
#include <stdio.h>

int giai_thua (int n) {
    if (n == 0) return 1;
    return n * giai_thua(n - 1);
}
```

```c
int main() {
    int n;
    puts("Nhap vao N: "); scanf("%d", &n);
    printf("N giai thua = %d", giai_thua(n));
    return 0;
}
```

Lưu ý:

Ở ví dụ 5, hàm *giai_thua* gọi đến hàm *giai_thua* (chính nó) bên trong nó. Việc một hàm tự gọi (hoặc định nghĩa lại) chính nó được gọi là *đệ quy (recursion)*. Trong lập trình, đây là một kỹ thuật mạnh nhưng cũng rất nguy hiểm nếu như người sử dụng không nắm rõ cách thức hoạt động của chương trình!

Bài tập

Lưu ý:

- Các bài tập dưới đây bắt buộc phải sử dụng kiến thức về *hàm*!
- Các hàm có trong ví dụ có thể được sử dụng lại.

1. Giả sử bạn đang là giáo viên và cần tính điểm điểm tổng kết cho sinh viên của mình. Biết rằng có n sinh viên, mỗi sinh viên đã có đủ 3 đầu điểm. Áp dụng ví dụ 1, hãy tính điểm tổng kết cho n sinh viên đó.

Input
dòng 1 là số n
n dòng tiếp theo mỗi dòng chứa 3 số ứng với điểm chuyên cần, giữa kỳ, và cuối kỳ của mỗi sinh viên
Output
n dòng mỗi dòng là điểm tổng kết của sinh tương ứng (làm tròn đến chữ số thứ 2 sau dấu phẩy)

Input	Output
4 5 6 7 6 7 8 6 6 8 9 9 6	6.40 7.40 7.20 7.20

Mở rộng bài toán: Sau khi tính điểm tổng kết, hãy phân loại điểm theo hệ chữ $A+$, A, $B+$, B, $\cdots$

2. Cho một mảng có n phần tử. Lập hàm tìm giá trị lớn nhất của mảng đó.

Input
dòng thứ nhất là số nguyên n
dòng thứ hai là n số nguyên
Output
1 số nguyên là giá trị lớn nhất của mảng

Input	Output
7 5 3 6 2 8 1 4	8

Mở rộng bài toán: Lập hàm tìm giá trị nhỏ nhất.

3. Ta biết hàm *rand* của thư viện *stdlib* trả về một số nguyên (giả) ngẫu nhiên trong khoảng từ 0 đến *RAND_MAX* (thường là 32767). Sử dụng hàm *rand* này, hãy lập một hàm trả về một số ngẫu nhiên trong khoảng $[a, b]$ với a, b nguyên dương cho trước.

Input
2 số nguyên dương a và b ($a < b$)
Output
1 số ngẫu nhiên

Input	Output
1 100	41

Lưu ý: con số 41 ở ví dụ sẽ thay đổi sau mỗi lần chạy lại chương trình, kể cả khi a và b giữ nguyên.

Mở rộng bài toán:
+ Lập một hàm trả về một mảng chứa các số ngẫu nhiên trong khoảng cho trước có đúng n phần tử.
+ Lập một hàm trả về một số thập phân ngẫu nhiên trong khoảng $[0, 1]$.
+ Ứng dụng những hàm ngẫu nhiên này để tạo ra *input* để kiểm tra cho các bài tiếp theo.

4. Cho ba số thực a, b, và c. Lập hàm tìm các nghiệm thực x_1 và x_2 ($x_1 \le x_2$) của phương trình $ax^2+bx+c=0$. Hàm này trả về 3 giá trị: *flag*, x_1, và x_2, trong đó *flag* có giá trị 0, 1, 2, hoặc 3 lần lượt tương ứng với các trường hợp phương trình vô nghiệm, nghiệm kép, hai nghiệm riêng biệt, hoặc vô số nghiệm; x_1 và x_2 là các nghiệm của phương trình. Khi phương trình vô nghiệm hoặc vô số nghiệm, $x_1 = x_2 = 0$.

Input
3 số a, b, và c là hệ số của phương trình
Output
3 số là *flag*, x_1, và x_2

Input	Output
2 5 -6	2 -3.386001 0.886001

5. Cho một *đa giác lồi* có n đỉnh có tọa độ (x_i, y_i) với $i = 1..n$. Tính diện tích đa giác.

Input
dòng 1 là 1 số n
n dòng tiếp mỗi dòng có 2 số x_i, y_i là tọa đỉnh i của đa giác
Output
1 số duy nhất là diện tích đa giác

Input	Output
5 0 2 1 1 4 2 3 6 1 5	12.500000

☞ Gợi ý:
+ Lập một hàm tính diện tích một tam giác khi biết tọa độ ba đỉnh.
+ Nên lập một hàm tính khoảng cách giữa hai điểm.
+ *GeoGebra* có thể hữu ích trong quá trình làm bài.

6. Cho n số. Tìm ước chung lớn nhất của n số đó.

Input
dòng 1 là 1 số n
dòng 2 là n số nguyên dương
Output
1 số duy nhất là ước chung lớn nhất của n số đã cho

Input	Output
4 42 36 54 48	6

7. Cho một mảng có n phần tử. Lập hàm sắp xếp lại các phần tử của mảng đó theo chiều tăng dần, sau đó loại bỏ các phần tử có giá trị trùng lặp.

Input
dòng 1 là số nguyên n
dòng 2 là n số nguyên là các phần tử của mảng
Output
Các phần tử của mảng sau khi đã xử lý

Input	Output
9 2 5 3 1 4 1 4 5 2	1 2 3 4 5

8. Cho hai số thực x và a. Lập một hàm tính x^a sử dụng khai triển Taylor với một sai số *err* cho trước.

Input
3 số thực x, a, và *err*
Output
giá trị của x^a

Input	Output
2.7 3.14 1e-9	22.619459

☞ Gợi ý: $x^a = e^{a \log_e(x)}$

9. Đa thức Legendre bậc n tại điểm x (ký hiệu $P_n(x)$ được định nghĩa như sau:

$$P_0(x) = 1, \qquad P_1(x) = x, \qquad P_n(x) = \frac{2n-1}{n}\, x\, P_{n-1}(x) - \frac{n-1}{n}\, P_{n-2}(x)$$

Cho n và x. Lập hàm tính đa thức Legendre bậc n tại điểm x, $P_n(x)$.

Input
2 số n và x
Output
1 số là giá trị của $P_n(x)$

Input	Output
3 3.14	72.68786

☞ : *Đa thức Legendre* là một họ các đa thức trực giao, có rất nhiều tính chất thú vị cũng như tính ứng dụng trong toán học và vật lý học.

10. *(Algebra)*
Cho một ma trận có kích thước $m \times n$. Lập hàm đọc và hàm ghi ra ma trận đó khi biết kích thước m và n.

Input
dòng đầu là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận
Output
m dòng, mỗi dòng chứa n số là các phần tử ma trận

Input	Output
2 3 1 2 3 4 5 6	1 2 3 4 5 6

☞ Gợi ý: Nên sử dụng các hàm này cho các bài tập về sau liên quan tới ma trận.

11. *(Algebra)*
Cho ma trận A có m hàng và n cột, và ma trận B có p hàng q cột. Lập hàm xét xem hai ma trận A và B có thực hiện được phép nhân ma trận không. Hàm này trả về 2 đối số đầu ra: *flag* và C, trong đó *flag* có giá trị 0 hoặc 1 tương ứng với các trường hợp không thực hiện được hoặc có thực hiện được phép nhân ma trận; C là kết quả của phép nhân ma trận, trong trường hợp không thực hiện được phép nhân, $C = NULL$.

Input
dòng đầu là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
dòng tiếp là 2 số p và q
p dòng tiếp, mỗi dòng chứa q số là các phần tử ma trận B
Output
dòng 1 là giá trị của *flag*
nếu $flag \neq 0$, các dòng tiếp theo là các phần tử ma trận C

Input	Output
1 2 2 1 2 3 1 2 3 4 5 6	1 6 9 12

12. *(Algebra)*
Cho ma trận A có m hàng và n cột. Cho hai u và v là chỉ số của hai hàng nào đó của ma trận A. Lập một hàm thực hiện thao tác tráo hai hàng u và v của ma trận A.

Input
dòng đầu là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
dòng tiếp là 2 số u và v
Output
m dòng, mỗi dòng chứa n số là các phần tử ma trận A sau khi đã tráo hai hàng

Input	Output
3 3 1 2 3 4 5 6 7 8 9 1 3	7 8 9 4 5 6 1 2 3

☞ Mở rộng bài toán: Lập các hàm thực hiện các thao tác tương tự là nhân hàng và cộng hàng của ma trận.

13. *(Algebra)*
Cho ma trận A có m hàng và n cột, và ma trận B có m hàng p cột. Lập hàm trả về ma trận C là ma trận mở rộng của A và B, $C = [A|B]$ (dễ thấy C có kích thước m hàng $n + p$ cột).

Input
dòng đầu là 3 số m, n, và p
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
m dòng tiếp, mỗi dòng chứa p số là các phần tử ma trận B
Output
m dòng là các phần tử ma trận C

Input	Output
2 2 3 1 2 3 4 5 6 7 8 9 0	1 2 5 6 7 3 4 8 9 0

14. *(Algebra)*
Cho ma trận A có m hàng và n cột. Lập một hàm thực hiện phương pháp khử Gauss-Jordan (không tráo hàng) lên ma trận A rồi trả về ma trận dạng bậc thang rút gọn sau khi đã khử xong.

Input
dòng đầu là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
Output
m dòng, mỗi dòng chứa n số là các phần tử ma trận sau khi đã khử Gauss-Jordan xong, làm tròn đến hai chữ số sau dấu phẩy

Input	Output
3 4 2 1 -1 8 -3 -1 2 -11 -2 1 2 -3	1.00 0.00 0.00 2.00 0.00 1.00 0.00 3.00 0.00 0.00 1.00 -1.00

Input	Output
3 7 2 4 -3 9 1 -8 6 1 2 4 21 -2 24 -4 3 6 1 30 2 22 17	1.00 2.00 0.00 9.00 0.00 4.00 2.00 0.00 0.00 1.00 3.00 0.00 6.00 1.00 0.00 0.00 0.00 0.00 1.00 2.00 5.00

👉 Gợi ý: Tham khảo bài số 7 của phần thực hành số 4. Ví dụ tham khảo tại: *link.*

15. *(Algebra)*
Cho ma trận vuông A có kích thước n. Lập hàm trả về giá trị định thức của A.

Input
dòng đầu là 1 số n
n dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
Output
1 số duy nhất là định thức của A
(làm tròn đến chữ số thứ 2 sau dấu phẩy)

Input	Output
3 5 0 2 1 3 4 -1 1 0	-12.00

👉 Gợi ý: Nên tính định thức bằng cách khử Gauss.

16. *(Algebra)*
Cho ma trận A có m hàng và n cột. Lập hàm trả về giá trị hạng của A.

Input
dòng đầu là 2 số m và n
m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
Output
1 số duy nhất là hạng của A

Input	Output
3 3 1 2 3 2 3 4 3 5 7	2

17. *(Algebra)*
Trong đại số tuyến tính, *định lý Kronecker–Capelli* phát biểu như sau:
Một hệ phương trình tuyến tính $Ax = b$ với n ẩn có nghiệm khi và chỉ khi hạng của ma trận hệ số A bằng hạng của ma trận mở rộng $[A|b]$. Nếu tồn tại nghiệm, đồng thời nếu $n = rank(A)$ thì nghiệm là duy nhất, nếu không thì hệ có vô số nghiệm.
Cho 3 số n (số ẩn), r_A (hạng của A), và r_{Ab} (hạng của $[A|b]$). Lập hàm trả về *flag*, trong đó *flag* có giá trị 0, 1, hoặc 2 ứng với các trường hợp hệ phương trình vô nghiệm, nghiệm duy nhất, hoặc vô số nghiệm.

Input
3 số n, r_A, và r_{Ab}
Output
1 số duy nhất là *flag*

Input	Output
3 3 3	1

18. *(Algebra)*
Cho ma trận vuông A có kích thước n. Lập hàm tính ma trận nghịch đảo của A. Hàm này trả về 2 đối số đầu ra: *flag* và A^{-1}, trong đó *flag* có giá trị 0 hoặc 1 tương ứng với các trường hợp ma trận A không khả đảo hoặc khả đảo; A^{-1} là ma trận nghịch đảo của A, trong trường hợp A không khả đảo, $A^{-1} = NULL$.

Input
dòng đầu là 1 số n
n dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A
Output
dòng 1 là giá trị của *flag*
nếu *flag* $\neq 0$, các dòng tiếp theo là các phần tử ma trận A^{-1}
(các giá trị của A^{-1} làm tròn đến chữ số thứ 2 sau dấu phẩy)

Input	Output
3 1 0 4 1 1 6 -3 0 -10	-5.00 0.00 -2.00 -4.00 1.00 -1.00 1.50 0.00 0.50

👉 Gợi ý: Nên áp dụng hàm của bài 13 kết hợp phương pháp khử Gauss (bài 14). Để kiểm tra lại kết quả, áp dụng hàm của bài 11.

19. *(Algebra)*

Cho ma trận hệ số A và vector cột b của hệ phương trình tuyến tính $Ax = b$. Lập một hàm giải hệ phương trình khi biết A và b. Hàm này trả về 2 đối số đầu ra: *flag* và x, trong đó *flag* có giá trị 0, 1, hoặc 2 ứng với các trường hợp hệ phương trình vô nghiệm, nghiệm duy nhất, hoặc vô số nghiệm; x là vector nghiệm của hệ, nếu $flag = 2$, x là một nghiệm bất kỳ, nếu $flag = 0$, $x = NULL$.

Input

dòng đầu là 2 số m và n

m dòng tiếp, mỗi dòng chứa n số là các phần tử ma trận A

dòng tiếp theo chứa n số là các phần tử vector cột b

Output

dòng 1 là giá trị của *flag*

nếu $flag \neq 0$, dòng 2 là n phần tử của x

(các giá trị của x làm tròn đến chữ số thứ 4 sau dấu phẩy)

Input	Output
3 3 1 1 0 -3 0 2 0 1 -2 3 0 2	1 0.2500 2.7500 0.3750

☞ Gợi ý: Có nhiều cách để giải quyết bài toán. Hãy áp dụng linh hoạt các hàm đã lập trong các bài từ 10 đến 18.

20. *(Recursion*)*

Cho số nguyên không âm n. Lập hàm tính giá trị phần tử thứ n, F_n, của dãy Fibonacci.

$$F_n = \begin{cases} 0 & \text{nếu } n = 0 \\ 1 & \text{nếu } n = 1 \\ F_{n-1} + F_{n-2} & \text{nếu } n > 1 \end{cases}$$

Input

1 số nguyên n

Output

1 số nguyên là giá trị của F_n

Input	Output
8	21

☞ Mở rộng bài toán: Để tăng tốc chương trình, nên khai báo mảng toàn cục để kiểm tra và lưu giữ các giá trị Fibonacci đã tính, tránh việc gọi đệ quy nhiều lần.

21. *(Recursion*)*

Cho số nguyên x và số nguyên dương n. Lập hàm tính x^n.

Input

2 số x và n

Output

1 số là giá trị của x^n

Input	Output
2 11	2048

☞ Gợi ý:

$$x^n = \begin{cases} 1 & \text{nếu } n = 0 \\ \left(x^{\frac{n}{2}}\right) \times \left(x^{\frac{n}{2}}\right) & \text{nếu } n \text{ chẵn} \\ \left(x^{\frac{n-1}{2}}\right) \times \left(x^{\frac{n-1}{2}}\right) \times x & \text{nếu } n \text{ lẻ} \end{cases}$$

22. *(Recursion*)*

Cho một dãy số nguyên gồm n phần tử $a_1, a_2, \ldots, a_n$. Biết rằng dãy con tăng đơn điệu là một dãy $a_{i_1}, a_{i_2}, \ldots, a_{i_k}$ thỏa mãn $i_1 < i_2 < \cdots < i_k$ và $a_{i_1} < a_{i_2} < \cdots < a_{i_k}$. Hãy cho biết dãy con tăng đơn điệu dài nhất (*LIS - Longest Increasing Subsequence*) của dãy đã cho có bao nhiêu phần tử.

Input

dòng thứ nhất là số nguyên n

dòng thứ hai là n số nguyên a_i

Output

độ dài của dãy con tăng đơn điệu dài nhất

Giải thích test ví dụ: dãy con dài nhất là dãy $a_1 = 1 < a_2 = 2 < a_4 = 4 < a_5 = 6$.

Input	Output
6 1 2 5 4 6 2	4

☞ Gợi ý: Gọi L_i là độ dài LIS tính đến phần tử thứ i của dãy ban đầu, ta có:

$$L_i = \begin{cases} 1 + max(L_j) & \text{nếu } 0 < j < i \text{ và } a_j < a_i \\ 1 & \text{nếu } \nexists\, j \end{cases}$$

23. *(Recursion*)*

Cho 2 dãy số nguyên a gồm m phần tử, b gồm n phần tử. Một dãy con của a là một dãy gồm một số các phần tử của a (không thay đổi thứ tự và không cần liên tiếp). Tương tự với b. Tìm độ dài dãy con chung dài nhất (*LCS - Longest common subsequence*) của hai dãy a và b.

Input

dòng thứ nhất là 2 số m và n
dòng thứ hai là m số nguyên a_i
dòng thứ ba là n số nguyên b_i

Output

độ dài của dãy con chung dài nhất của a và b

Input	Output
6 7 1 2 5 4 6 2 1 5 3 2 4 1 6	4

Giải thích test ví dụ: dãy con chung dài nhất có thể là $1, 2, 4, 6$ *hoặc* $1, 5, 4, 6$.

☞ Gợi ý: Gọi $L_{i,j}$ là độ dài LCS tính đến phần tử thứ i của a và phần tử thứ j của b, ta có:

$$L_{i,j} = max \begin{cases} 1 + L_{i-1,j-1} & \text{nếu } a_i = b_j \\ max(L_{i,j-1},\ L_{i-1,j}) & \text{nếu } a_i \neq b_j \end{cases}$$

24. *(Recursion*)*

Cho số nguyên dương n, $(0 < n < 10)$. Hãy liệt kê ra tất cả các xâu ký tự có độ dài n có thể có, biết rằng các xâu này chỉ chứa các ký tự 'A' và 'B'. (Hãy liệt kê theo thứ tự từ điển)

Input

1 số nguyên n

Output

2^n dòng tiếp theo là các xâu ký tự có độ dài n

Input	Output
3	AAA AAB ABA ABB BAA BAB BBA BBB

☞ Gợi ý: Theo cách tiếp cận thông thường, nếu n cố định thì ta sẽ viết n vòng *for* lồng nhau để giải quyết bài toán. Tuy nhiên, khi n thay đổi, ta có thể lợi dụng hàm đệ quy để mô tả các vòng *for* này (mỗi hàm mô tả 1 vòng *for*). Kỹ thuật lợi dụng đệ quy để xây dựng dần dần lời giải này được gọi là *backtrack*.

25. *(Recursion*)*

Cho số nguyên dương n. Hãy liệt kê ra tất cả các hoán vị của dãy gồm n số có giá trị từ 1 đến n.

Input

1 số nguyên n

Output

$n!$ dòng tiếp, mỗi dòng chứa n số mô tả 1 hoán vị

Input	Output
3	1 2 3 1 3 2 2 1 3 2 3 1 3 1 2 3 2 1

☞ Gợi ý: Tương tự như bài trước, kỹ thuật nên sử dụng ở đây là *backtrack*, tuy nhiên "vòng *for*" lúc này có kèm thêm điều kiện.