

**ଭଜନ ଚକ୍ର**



# THỰC HÀNH KỸ THUẬT LẬP TRÌNH


*(lưu hành nội bộ)*



**THÀNH PHỐ HỒ CHÍ MINH – NĂM 2022**

## MỤC LỤC

BUỔI 1: KỸ THUẬT XỬ LÝ MẢNG MỘT CHIỀU, CON TRỎ VÀ XỬ LÝ NGOẠI LỆ .....	2
BUỔI 2: CÁC GIẢI THUẬT TÌM KIẾM VÀ SẮP XẾP .....	9
BUỔI 3. MẢNG STRUCT & FILE .....	12
BUỔI 4: KỸ THUẬT XỬ LÝ MẢNG HAI CHIỀU .....	22
BUỔI 5. ÔN TẬP - KIỂM TRA LẦN 1 .....	26
BUỔI 6. XỬ LÝ CHUỖI .....	28
BUỔI 7. KỸ THUẬT ĐỆ QUY .....	35
BUỔI 8. KỸ THUẬT ĐỆ QUY (tt) .....	40
BUỔI 9. BÀI TẬP TỔNG HỢP .....	43
BUỔI 10. ÔN TẬP - KIỂM TRA LẦN 2 .....	47
PHỤ LỤC BÀI TẬP THỰC HÀNH NÂNG CAO .....	49

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p><b>BUỔI 1: KỸ THUẬT XỬ LÝ</b></p> <p><b>MẢNG MỘT CHIỀU, CON</b></p> <p><b>TRỎ VÀ XỬ LÝ NGOẠI LỆ</b></p>	
--	--	---

## A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng một chiều, con trỏ và xử lý ngoại lệ cho các bài toán trong những trường hợp thực tế.

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

### PHẦN 1. TÓM TẮT LÝ THUYẾT

#### 1. Các quy định cơ bản trong viết code:

- Qui tắc đặt tên.
- Chú thích.
- Viết code rõ ràng – Clean code.

#### 2. Xử lý ngoại lệ

```

try {
    //Các lệnh được kiểm tra lỗi trong khi thực hiện
}
catch ( ) {
    //Các lệnh được thực hiện khi xảy ra lỗi
}

```

#### 3. Con trỏ

- Con trỏ là biến dùng để lưu địa chỉ những biến khác hay địa chỉ vùng nhớ hợp lệ (Con trỏ dùng để trỏ đến địa chỉ của biến khác).
- Con trỏ NULL là con trỏ đặc biệt và không trỏ đến bất kỳ một vùng nhớ nào cả.
- Toán tử để lấy địa chỉ của biến vào con trỏ: **&**

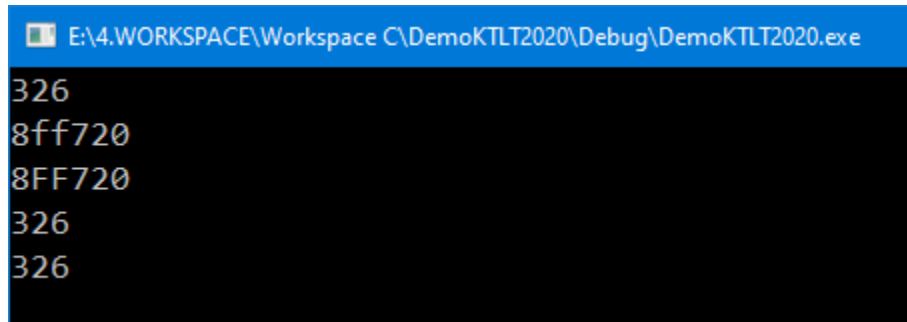
$$\text{<tên biến con trỏ> = \&\text{<tên biến>;}$$
- Toán tử truy xuất đến ô nhớ mà con trỏ trỏ đến: **\***

$$\text{*<tên biến con trỏ>}$$

Ví dụ:

```
int a = 326;
int* p;
p = &a;
int b = *p;

printf("%d \n", a); // giá trị biến a
printf("%x \n", *(&a)); // địa chỉ ô nhớ của biến a, in thường
printf("%X \n", p); // địa chỉ ô nhớ con trỏ p trỏ đến, in hoa
printf("%d \n", *p); // giá trị ô nhớ mà con trỏ p trỏ đến
printf("%d \n", b); // b chứa giá trị ô nhớ mà con trỏ p trỏ đến
```



```
E:\4.WORKSPACE\Workspace C\DemoKTLT2020\Debug\DemoKTLT2020.exe
326
8ff720
8FF720
326
326
```

#### 4. Con trỏ và mảng 1 chiều

Có thể sử dụng con trỏ và mảng một chiều tương đương nhau trong một số trường hợp

```
void main()
{
    int a[10], n = 4, *pa;
    p = a; // hoặc pa = &a[0];
    for (int i = 0; i < n; i++){
        scanf("%d", &a[i]); // scanf("%d", &p[i]);
        scanf("%d", a + i); // scanf("%d", p + i);
        scanf("%d", a++); // scanf("%d", p++);
    }
}
// &a[i] ⇔ (a + i) ⇔ (pa + i) ⇔ &pa[i]
```

```
void main()
{
    int a[10], n = 5, *pa;
    pa = a; // hoặc pa = &a[0];

    for (int i = 0; i < n; i++)
    {
        printf("%d", a[i]); // printf("%d", p[i]);
        printf("%d", *(a + i)); // printf("%d", *(p + i));
        printf("%d", *(a++)); // printf("%d", *(p++));
    }
}
```

#### 5. Con trỏ và cấu trúc

Truy xuất thành phần cấu trúc qua biến con trỏ

<tên biến con trỏ cấu trúc>-><tên thành phần>  
(<tên biến con trỏ cấu trúc>).<tên thành phần>

**Ví dụ:**

```
struct PhanSo
{
    int tu, mau;
};
PhanSo ps1, *ps2 = &ps1; // ps2 là con trỏ
ps1.tu = 1; ps1.mau = 2;
ps2->tu = 1; ps2->mau = 2; // tương đương (*ps2).tu = 1; (*ps2).mau = 2;
```

## PHẦN 2. BÀI TẬP CÓ HƯỚNG DẪN

**Bài 1.** Viết chương trình thực hiện tính chia 2 số nguyên a và b.

```
void main()
{
    int a = 0, b = 0;
    printf("Nhap a va b: ");
    scanf("%d %d", &a, &b);

    try
    {
        if(b == 0)
            throw "So chia khong duoc la 0";
        else
            int t=a/b;
    }
    catch(const char *st)
    {
        printf("Loi: %s", st);
    }
    getch();
}
```

**Bài 2.** Viết hàm tính tuổi dựa theo năm sinh. Giả sử chỉ xét đến những người sinh từ năm 1920. Cần xử lý thông báo lỗi cho các trường hợp ngoại lệ.

```
int tinhTuoi (int namSinh) {
    //cần khai báo thư viện #include <ctime> để dùng được time_t và tm
    time_t now = time(0);
    tm *ltm = localtime(&now);
    int namHH = 1900 + ltm->tm_year;
    try {
        if(namSinh <= 0 || namSinh > namHH)
            throw 101; //mã lỗi 101
        else
            if(namSinh < 1920)
                throw 102; //mã lỗi 102
    }
```

```

        else
            return namHH - namSinh;
    }
    catch (int errCode) {
        if( errCode == 101)
            printf("nam sinh khong hop le\n");
        else
            printf("nam sinh <1920 \n");
        return -1;
    }
}

```

**Bài 3.** Viết chương trình nhập 2 số thực bất kỳ a, b. Tạo 2 con trỏ pa và pb trỏ đến a, b. Xuất giá trị các con trỏ pa, pb.

```

float a = 5.7, b = 4.8;
float *pa = &a;
float *pb = &b;
printf("\nDia chi cua a: %x", pa);
printf("\nDia chi cua b: %x", pb);

```

**Bài 4.** Viết chương trình nhập/xuất mảng 1 chiều chứa số nguyên bằng cách dùng con trỏ là tên mảng.

```

void nhapM1C (int * &a, int &n)
{ //a và n là tham chiếu vì sau khi nhập giá trị, a và n cần giữ các giá trị mới nhận trong hàm
    a = (int *)malloc(10*sizeof(int)); // cấp phát a có 10 phần tử
    n = 10;
    for(int i = 0; i < n; i++)
        *(a+i) = rand()%100;
}

void xuatM1C(int *a, int n)
{
    for(int i = 0; i < n; i++)
    {
        printf("\nPhan tu thu %d co gia tri %d va dia chi o nho la %x", i, *(a+i), a+i);
        printf("\nPhan tu thu %d co gia tri %d va dia chi o nho la %x", i, a[i], a+i);
    }
}

void main()
{
    int *a = new int, n=0;
    nhapM1C(a, n);
    xuatM1C(a, n);
    getch();
}

```

### PHẦN 3. BÀI THỰC HÀNH TRÊN LỚP

**Bài 5.** Nhập 3 số nguyên a, b, c. Xuất kết quả  $c/(a-b)$ .

**Bài 6.** Viết chương trình nhập họ tên, ngày sinh và giới tính của người lao động. Hãy tính thời gian người lao động được nghỉ hưu, biết rằng tuổi hưu của nam là đủ 62 tuổi, và nữ là đủ 60 tuổi.

*Lưu ý:* Xét năm hiện tại, nếu tuổi nhập vào không nằm trong tuổi lao động (18→60 hoặc 62 theo đúng giới tính) và giới tính không phải nam/nữ thì phải xử lý ngoại lệ. Cụ thể:

- Nếu tuổi không thuộc trong tuổi lao động thì “ném” lỗi mã 101
- Nếu giới tính không phải chuỗi nam/nữ thì “ném” lỗi là chuỗi errcode.

Ví dụ:

- Nguyễn Văn An, giới tính nam, sinh ngày 20/03/1990. Hiện tại (năm 2021) An đã 31 tuổi. Thời gian An được nghỉ hưu là tháng 03/2052.
- Lê Thị Hoa, giới tính nữ, sinh ngày 14/12/1995. Hiện tại (năm 2021) Hoa đã 26 tuổi. Thời gian Hoa được nghỉ hưu là tháng 01/2056.

**Bài 7.** Xét bài tập 4, viết các hàm sau theo dạng thao tác trên con trỏ

- Tìm phần tử lớn nhất của a, xuất ra phần tử lớn nhất và địa chỉ của nó thông qua con trỏ mảng.
- Xuất địa chỉ của phần tử chẵn lớn nhất và phần tử lẻ nhỏ nhất, nếu không có thì báo không có phần tử chẵn/lẻ trong mảng.
- Xóa phần tử có giá trị 0.
- Thêm phần tử x vào sao phần tử đầu tiên.
- Tính tổng các phần tử là số chính phương.
- Xuất các số cực đại trong a. Biết rằng số cực đại là số lớn hơn các số quanh nó.

**Bài 8.** Tạo cấu trúc Phân số chứa 2 thành phần tử và mẫu số (mẫu  $\neq 0$ ). Tạo con trỏ mảng 1 chiều chứa các phân số. Viết các hàm:

- Nhập/xuất các phần tử của mảng. Lưu ý ngoại lệ khi mẫu là 0.
- Xuất các phân số có mẫu > tử.
- Đếm số phân số có mẫu và tử chẵn.
- Rút gọn phân số.
- Tính tích các phần tử của mảng.
- Tìm phần tử lớn nhất.

### PHẦN 4. BÀI THỰC HÀNH VỀ NHÀ

**Bài 9.** Xét lại bài 3, tính các giá trị tổng, hiệu, tích, thương của 2 số a, b thông qua các con trỏ pa, pb. Xuất ra kết quả và địa chỉ các ô nhớ chứa tổng, hiệu, tích thương đó.

**Bài 10.** Viết chương trình nhập vào chuỗi st (dạng con trỏ).

- Xuất giá trị từng ký tự của st thông qua con trỏ trỏ đến chuỗi.
- Chuyển các ký tự của chuỗi về dạng chữ hoa (gợi ý: thay đổi mã ASCII)

c. Chuyển các ký tự đầu mỗi từ (đứng sau dấu cách) của chuỗi về dạng chữ hoa (gợi ý: thay đổi mã ASCII)

Ví dụ: “truong dai hoc CNTP TPHCM” → “Truong Dai Hoc CNTP TPHCM”

**Bài 11.** Xét tiếp bài số 7. Viết các hàm sau, chú ý xử ngoại lệ (nếu có)

- Xuất các số cực tiểu trong a. Biết rằng số cực tiểu là số nhỏ hơn các số quanh nó.
- Xóa phần tử tại vị trí k
- Thêm phần tử x tại vị trí k
- Chuyển số chẵn lên đầu mảng, số lẻ xuống cuối mảng.
- Kiểm tra mảng có chứa chẵn lẻ xen kẽ không?

**Bài 12.** Xét lại bài 8, Viết các hàm sau:

- Tìm phân số lớn nhất/nhỏ nhất
- Xóa phần tử tại vị trí k
- Thêm phần tử x tại vị trí k

**Bài 13.** Xét 2 mảng 1 chiều a và b. Tính và xuất kết quả các phép chia của phần tử mảng a cho phần tử mảng b. Hãy xét các trường hợp ngoại lệ có thể có trong bài toán này.

**Bài 14.** Xét lại bài 6, viết chương trình nhập họ tên, ngày sinh và giới tính của người lao động. Hãy tính thời gian người lao động được nghỉ hưu dựa theo quy định của Bộ luật lao động Việt Nam 2019 như sau:

Lao động nam			Lao động nữ		
Năm đủ tuổi nghỉ hưu	Tuổi nghỉ hưu	Năm sinh	Năm đủ tuổi nghỉ hưu	Tuổi nghỉ hưu	Năm sinh
2021	60 tuổi 3 tháng	Từ tháng 01/1961 đến tháng 9/1961	2021	55 tuổi 4 tháng	Từ tháng 01/1966 đến tháng 8/1966
2022	60 tuổi 6 tháng	Từ tháng 10/1961 đến tháng 6/1962	2022	55 tuổi 8 tháng	Từ tháng 9/1966 đến tháng 4/1967
2023	60 tuổi 9 tháng	Từ tháng 7/1962 đến tháng 3/1963	2023	56 tuổi	Từ tháng 5/1967 đến tháng 12/1967
2024	61 tuổi	Từ tháng 4/1963 đến tháng 12/1963	2024	56 tuổi 4 tháng	Từ tháng 01/1968 đến tháng 8/1968
2025	61 tuổi 3 tháng	Từ tháng 01/1964 đến tháng 9/1964	2025	56 tuổi 8 tháng	Từ tháng 9/1968 đến tháng 5/1969
2026	61 tuổi 6 tháng	Từ tháng 10/1964 đến tháng 6/1965	2026	57 tuổi	Từ tháng 6/1969 đến tháng 12/1969



2027	61 tuổi 9 tháng	Từ tháng 7/1965 đến tháng 3/1966	2027	57 tuổi 4 tháng	Từ tháng 01/1970 đến tháng 8/1970
2028	62 tuổi	Từ tháng 4/1966 trở đi	2028	57 tuổi 8 tháng	Từ tháng 9/1970 đến tháng 4/1971
			2029	58 tuổi	Từ tháng 5/1971 đến tháng 12/1971
			2030	58 tuổi 4 tháng	Từ tháng 01/1972 đến tháng 8/1972
			2031	58 tuổi 8 tháng	Từ tháng 9/1972 đến tháng 4/1973
			2032	59 tuổi	Từ tháng 5/1973 đến tháng 12/1973
			2033	59 tuổi 4 tháng	Từ tháng 01/1974 đến tháng 8/1974
			2034	59 tuổi 8 tháng	Từ tháng 9/1974 đến tháng 4/1975
			2035	60 tuổi	Từ tháng 5/1975 trở đi

**Gợi ý:**

Mỗi người lao động có năm sinh càng về sau thì số tuổi nghỉ hưu càng tăng. Do vậy cần phân loại từng năm sinh để tính cho đúng quy định.

**--HẾT--**

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p><b>BUỔI 2: CÁC GIẢI THUẬT TÌM KIẾM VÀ SẮP XẾP</b></p>	
--	--	--

### A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng một chiều cho các bài toán trong những trường hợp thực tế.

### B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

### C. NỘI DUNG THỰC HÀNH:

#### Phần 1. Tóm tắt lý thuyết

##### – Tìm kiếm tuyến tính – **LinearSearch**

Gọi x là giá trị cần tìm và a là mảng chứa dãy số dữ liệu. Các bước tiến hành như sau:

**Bước 1:**  $i=1$ ;

**Bước 2 :** So sánh  $a[i]$  với x, có 2 khả năng :

$a[i] = x$ : Tìm thấy. Dừng.

$a[i] \neq x$ : Sang bước 3.

**Bước 3 :**  $i = i + 1$ ; //xét phần tử kế

Nếu  $i > N$ : hết mảng, không tìm thấy. Dừng

Ngược lại: lặp lại bước 2.

##### – Tìm kiếm nhị phân – **BinarySearch**

**Bước 1:** Khởi đầu tìm kiếm trên tất cả các phần tử

$left = 1$ ;  $right = N$ ;

**Bước 2:**  $middle = (left+right)/2$ ;

So sánh  $a[middle]$  với x, có 3 khả năng :

$a[middle] = x$  : Tìm thấy. Dừng

$a[middle] > x$  : Chuẩn bị tìm tiếp x trong dãy con  $a_{left}..a_{middle-1}$  :  $right = middle - 1$ ;

$a[middle] < x$ : Chuẩn bị tìm tiếp x trong dãy con  $a_{middle+1}..a_{right}$  :  $left = middle + 1$ ;

##### – Sắp xếp theo giải thuật đổi chỗ trực tiếp – **InterchangeSort**

Lặp  $i = 1, 2, \dots, n-1$

Lặp  $j = i+1, i+2, \dots, n$

Nếu  $a[i] > a[j]$  thì

```

x = a[i];
a[i] = a[j];
a[j] = x;
    Kết thúc nếu.
    Kết thúc lặp j.
    Kết thúc lặp i.

```

– Sắp xếp theo giải thuật chọn trực tiếp - **SelectionSort**

```

Bước 1: i = 0;
Bước 2: Tìm phần tử a[min] nhỏ nhất trong dãy hiện hành từ a[i] đến a[n-1]
Bước 3: Đổi chỗ a[min] và a[i]
Bước 4: Nếu i < n-1 thì
        i = i+1; Lặp lại Bước 2;
    Ngược lại: Dừng.

```

– Sắp xếp theo giải thuật **QuickSort**

```

Bước 1: Nếu left ≥ right //dãy có ít hơn 2 phần tử
        Kết thúc; //dãy đã được sắp xếp
Bước 2: Phân hoạch dãy aleft ... aright thành 3 đoạn: aleft.. aj, aj+1.. ai-1, ai.. aright
        Đoạn 1: aleft.. aj có giá trị nhỏ hơn x
        Đoạn 2: aj+1.. ai-1 có giá trị bằng x
        Đoạn 3: ai.. aright có giá trị lớn hơn x
Bước 3: Sắp xếp đoạn 1: aleft.. aj
Bước 4: Sắp xếp đoạn 3: ai.. aright

```

## Phần 2. Bài tập thực hành trên lớp.

**Bài 1.** Cho mảng một chiều a chứa n số nguyên. Xây dựng hàm thực hiện các yêu cầu sau:

**Lưu ý:**

- Để lấy số ngẫu nhiên từ 0 đến k, ta sử dụng  $\text{rand}() \% (k+1)$
  - Để lấy số ngẫu nhiên từ a đến b, ta sử dụng  $a + \text{rand}() \% (b-a+1)$
1. Tạo mảng một chiều ngẫu nhiên có số phần tử lớn hơn hoặc bằng 15.
  2. Tạo mảng chứa toàn số chẵn.
  3. Tìm kiếm x trong a (trả về vị trí của x/ trả lời có hoặc không?) theo giải thuật Linear Search.
  4. Sắp xếp a tăng dần/giảm dần theo giải thuật Interchange Sort.
  5. Sắp xếp a giảm dần/giảm dần theo giải thuật Interchange Sort.
  6. Tìm kiếm x trong a (trả về vị trí của x/ trả lời có hoặc không?) theo giải thuật Binary Search với mảng a tăng/giảm.
  7. Sắp xếp a tăng dần theo giải thuật Selection Sort.
  8. Sắp xếp a giảm dần theo giải thuật Selection Sort.
  9. Sắp xếp a tăng dần theo giải thuật Quick Sort.
  10. Sắp xếp a giảm dần theo giải thuật Quick Sort.

**Bài 2.** Tạo mảng 1 chiều b chứa hỗn số (gồm 3 thành phần: phần nguyên và phân số). Viết các hàm:

1. Tạo mảng b chứa giá trị hỗn số ngẫu nhiên. Lưu ý mẫu khác 0.
2. Xuất danh sách hỗn số.
3. So sánh 2 hỗn số.
4. Chuyển hỗn số  $\rightarrow$  phân số
5. Chuyển phân số  $\rightarrow$  hỗn số.
6. Tính tổng, hiệu, tích, thương 2 hỗn số.
7. Sắp xếp b tăng/giảm theo 3 giải thuật sắp xếp đã học.

### **Phần 3. Bài tập về nhà**

#### **Tiếp theo bài số 1.**

1. Viết hàm nhập vào số nguyên n và liệt kê các số nguyên tố nhỏ hơn n, nếu mảng không tồn tại số nguyên tố nào nhỏ hơn n thì phải xuất ra một câu thông báo.
2. Tính tổng các phân tử có chữ số đầu là chữ số lẻ.
3. Liệt kê số lần xuất hiện của các phân tử trong mảng.
4. Sắp xếp mảng có số chẵn tăng dần, số lẻ giảm dần.
5. Tìm dãy con giảm dài nhất trong a.
6. Tìm số nhỏ thứ 2 trong mảng.
7. Cho x là số có 2 chữ số. Tìm trong a các phân tử có trùng với x đúng 2 chữ số đó, không nhất thiết 2 số đó nằm gần nhau. Ví dụ: x=28, các phân tử thỏa: 208, 382, 15298, ... chỉ cần chữ 2 và 8
8. Sắp xếp mảng sao cho các phân tử chẵn tăng dần, các phân tử lẻ giữ nguyên vị trí.
9. Sắp xếp mảng: số lẻ ở đầu mảng, số chẵn ở cuối mảng.

#### **Tiếp theo bài số 2.**

1. Tìm hỗn số x trong mảng b theo giải thuật linear search.
2. Sắp xếp b sao cho các phân tử có phần nguyên chẵn lên đầu, phân tử có phần nguyên lẻ ở cuối mảng.
3. Tìm hỗn số x theo giải thuật binary search trong mảng b đã được sắp xếp tang/giảm.
4. Chia mảng b thành 2 mảng s1 và s2, với s1 chứa các phần nguyên, s2 chứa phân số.
5. Xóa phân tử thứ k trong mảng b.
6. Thêm hỗn số x vào mảng b tại vị trí k.
7. Tạo mảng c chứa các phân tử là phân số được đổi từ các phân tử hỗn số của mảng b.
8. Tính tổng các phân tử của mảng b.
9. Tìm phân tử lớn nhất/nhỏ nhất của b.
10. Xuất vị trí phân tử có phần nguyên chẵn của b.
11. Xuất vị trí của phân tử lớn nhất/nhỏ nhất của b.

-- HẾT--

<p><b>Trường ĐH CNTP TP. HCM</b></p> <p><b>Khoa Công nghệ thông tin</b></p> <p><b>Bộ môn Công nghệ phần mềm</b></p> <p><b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b></p>	<p><b>BUỔI 3.</b></p> <p><b>MẢNG STRUCT &amp; FILE</b></p>	
--	--	--

### A. MỤC TIÊU:

- Định nghĩa được kiểu cấu trúc trong bài toán cụ thể.
- Khai báo được biến kiểu mảng cấu trúc.
- Phân tích được tình huống sử dụng mảng cấu trúc để lưu trữ dữ liệu.
- Cài đặt được hàm nhập/ đọc dữ liệu từ file text, xuất dữ liệu bằng cách sử dụng mảng cấu trúc.
- Áp dụng các kỹ thuật cài đặt xử lý trên mảng cấu trúc vào những bài toán thực tế.

### B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

### C. NỘI DUNG THỰC HÀNH

#### I. Tóm tắt lý thuyết:

##### 1. Cấu trúc

- Cấu trúc là cách cho người lập trình tự định nghĩa một kiểu dữ liệu mới bao gồm nhiều thành phần có thể thuộc nhiều kiểu dữ liệu khác nhau. Các thành phần được truy nhập thông qua một tên.

##### Cú pháp:

```
struct [tên_cấu_trúc]
{
    //khai báo các thành phần
};
```

##### Ví dụ:

```
struct SinhVien
{
    char mssv[11];
    char hoTen[50];
    float diemTB;
};
```

- Các thành phần của cấu trúc được truy nhập thông qua tên biến cấu trúc và tên thành phần.

*<Tên biến cấu trúc>.<tên\_thành\_phần>*

*Hoặc: <tên biến con trỏ cấu trúc> -> <tên thành phần>*

## 2. Mảng cấu trúc

Cú pháp: `<tên cấu trúc> <tên mảng> [ <kích thước> ];`

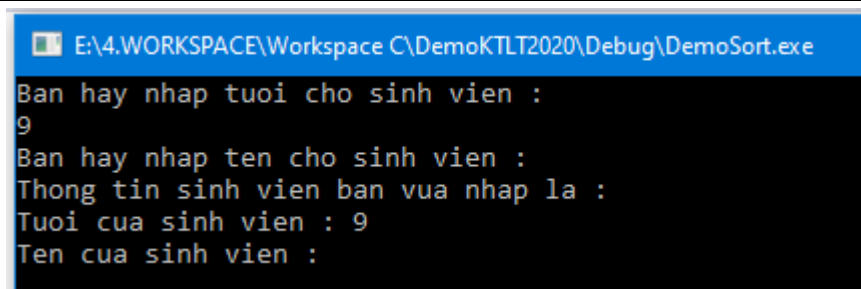
Ví dụ:

```
SinhVien dssv[100];
puts("Nhap ma so sinh vien:"); //xuất chuỗi; thuộc thư viện stdio.h
gets(a[2].mssv); // nhập chuỗi, gets khác scanf ở chỗ chấp nhận chuỗi các khoảng cách.
puts("Nhap ho ten:");
gets(a[2].hoten);
printf("Nhap so lan vang mat:");
scanf("%d", a[2].solanvang);
```

### Lưu ý: Cách dùng fflush (stdin)

Xét ví dụ sau:

```
void main() {
    char ten[100];
    int tuoi;
    printf("Ban hay nhap tuoi cho sinh vien: \n");
    scanf("%d", &tuoi);
    printf("Ban hay nhap ten cho sinh vien: \n");
    gets(ten);
    printf("Thong tin sinh vien ban vua nhap la: \n");
    printf("Tuoi cua sinh vien: %d\n", tuoi);
    printf("Ten cua sinh vien: %s\n", ten);
    getch();
}
```



Chúng ta có thể thấy sau khi nhập tuổi sinh viên thì trình biên dịch đã bỏ qua bước nhập tên cho sinh viên. Vào thời điểm chúng ta sử dụng lệnh `scanf` để nhập tuổi và kết thúc lệnh `scanf` bằng phím `enter`, thì trong bộ nhớ đệm của ta đã lưu kí tự `"\n"` và gán vào cho hàm `gets` tiếp theo, do vậy chương trình bỏ qua lệnh `gets` của chúng ta, không cho chúng ta nhập chuỗi ở `gets`. Đây là lí do tại sao chúng ta cần sử dụng câu lệnh `fflush(stdin);` để xóa `"\n"` trong bộ nhớ đệm trước khi thực hiện lệnh `gets`.

## 3. Đọc/ghi file text (txt) trong C.

### 3.1. Cách 1.

Đọc/ghi file txt trong C, ta dùng các hàm đọc/ghi trong thư viện **stdio.h**

Đề đọc/ghi file cần khai báo con trỏ FILE , cú pháp mở file:

**FILE \*fp;**

FILE \*fp;

fp = fopen("fileopen", "mode")

**Ví dụ:**

fopen("E:\\cprogram\\newprogram.txt", "w"); // mở file txt theo đường dẫn để ghi file

fopen("E:\\cprogram\\oldprogram.bin", "r"); // mở file txt theo đường dẫn để đọc file

Một số chuẩn mode để đọc/ghi file

File Mode	Meaning of Mode	During Inexistence of file
r	Open for reading.	If the file does not exist, fopen() returns NULL.
rb	Open for reading in binary mode.	If the file does not exist, fopen() returns NULL.
w	Open for writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb	Open for writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a	Open for append. i.e, Data is added to end of file.	If the file does not exist, it will be created.
ab	Open for append in binary mode. i.e, Data is added to end of file.	If the file does not exist, it will be created.
r+	Open for both reading and writing.	If the file does not exist, fopen() returns NULL.
rb+	Open for both reading and writing in binary mode.	If the file does not exist, fopen() returns NULL.
w+	Open for both reading and writing.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
wb+	Open for both reading and writing in binary mode.	If the file exists, its contents are overwritten. If the file does not exist, it will be created.
a+	Open for both reading and appending.	If the file does not exist, it will be created.
ab+	Open for both reading and appending in binary mode.	If the file does not exist, it will be created.

Ví dụ: Ghi file sử dụng `fprintf()`

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

int main()
{
    int num;
    FILE *fptr;
    fptr = fopen("C:\\program.txt","w");

    if(fptr == NULL)
    {
        printf("Error!");
        exit(1);
    }

    printf("Enter num: ");
    scanf("%d",&num);

    fprintf(fptr,"%d",num);
    fclose(fptr);

    return 0;
}
```

Đọc file sử dụng `fscanf()`:

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

int main()
{
    int num;
    FILE *fptr;

    if ((fptr = fopen("C:\\program.txt","r")) == NULL){
        printf("Error! opening file");

        // Program exits if the file pointer returns NULL.
        exit(1);
    }

    fscanf(fptr,"%d", &num);

    printf("Value of n=%d", num);
    fclose(fptr);

    return 0;
}
```

(Nguồn: <https://www.programiz.com/c-programming/c-file-input-output#example-write>)

### 3.2. Đọc/ghi file dùng thư viện `fstream` trong namespace `std` (C++)

- Khai báo thư viện: `#include <fstream>`

- Mở tạo file để đọc/ghi:

```
void open(const char *ten_file, ios::che_do); // chế độ (phần này không bắt buộc có)
```



Chế độ	Miêu tả
ios::app	Chế độ Append. Tất cả output tới file đó được phụ thêm vào cuối file đó
ios::ate	Mở một file cho output và di chuyển điều khiển read/write tới cuối của file
ios::in	Mở một file để đọc
ios::out	Mở một file để ghi
ios::trunc	Nếu file này đã tồn tại, nội dung của nó sẽ được cắt (truncate) trước khi mở file

**Ví dụ:**

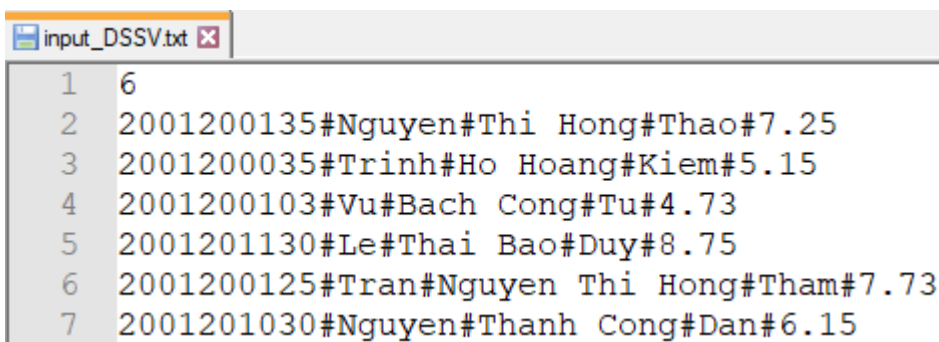
```
ofstream outfile;
outfile.open("file.txt", ios::out | ios::trunc );
//mở một file trong chế độ ghi và muốn cắt (truncate) nó trong trường hợp nó đã tồn tại
fstream infile;
infile.open("file.txt", ios::out | ios::in ); // mở một file với mục đích đọc và ghi
```

## II. Bài tập có hướng dẫn

**Yêu cầu:** dùng NNLTT C/C++ để minh họa, viết chương trình theo dạng cấu trúc (hàm con), tổ chức chương trình theo dạng hiển thị menu các bài toán và cho phép người dùng lựa chọn bài toán cần thực thi.

Cho mảng cấu trúc sinh viên có n phần tử. Hãy đọc/ghi dữ liệu từ file txt. Biết rằng sinh viên có các thông tin: mã số sinh viên, họ, họ lót, tên và điểm trung bình.

### 1. Cách tổ chức file: *input\_DSSV.txt*



```
1 6
2 2001200135#Nguyen#Thi Hong#Thao#7.25
3 2001200035#Trinh#Ho Hoang#Kiem#5.15
4 2001200103#Vu#Bach Cong#Tu#4.73
5 2001201130#Le#Thai Bao#Duy#8.75
6 2001200125#Tran#Nguyen Thi Hong#Tham#7.73
7 2001201030#Nguyen#Thanh Cong#Dan#6.15
```

**Chương trình mẫu:**

```
#include<conio.h>
#include<stdio.h>
#include<string.h>
//=====
//* Khai biến toàn cục, kiểu dữ liệu cấu trúc
```

```

//=====
#define MAXSIZE 100
#define fNameIn_DSSV "input_DSSV.txt" //Định nghĩa tên mới để đọc file.
#define fNameOut_DSSV "output_DSSV.txt" //Định nghĩa tên mới để ghi file.
//-----
typedef struct SinhVien
{
    char mssv[11];
    char ho[8], hoLot[20], ten[8];
    double diemTB;
}SV;
//=====
/* Định nghĩa các hàm
//=====
void xuấtTTSV(SV x)
{ //Xuất ra màn hình thông tin của 1 sinh viên
    //printf(fo, "%s \t %s %s %s \t %5.2lf\n", x.mssv, x.ho, x.hoLot, x.ten, x.diemTB);
    printf("%-15s%-10s%-20s%-10s%5.2lf\n", x.mssv, x.ho, x.hoLot, x.ten, x.diemTB);
}
//=====
void docTTSV(FILE *fi, SV &x)
{ //Đọc thông tin của 1 sinh viên từ file
    //Dùng phím tab (\t) để ngăn cách các thành phần thông tin
    //fscanf(fi, "%[^\t]*%^[^\t]*%^[^\t]*%^[^\t]*%lf", &x.mssv, &x.ho, &x.hoLot,
    &x.ten, &x.diemTB);
    //Dùng phím # để ngăn cách các thành phần thông tin
    fscanf(fi, "%[^\t]*%^[^\t]*%^[^\t]*%^[^\t]*%lf", &x.mssv, &x.ho, &x.hoLot, &x.ten,
    &x.diemTB);
}
//=====
void ghiTTSV(FILE *fo, SV x)
{ //Xuất (ghi) ra file thông tin của 1 sinh viên
    fprintf(fo, "%-15s%-10s%-20s%-10s%5.2lf\n", x.mssv, x.ho, x.hoLot, x.ten, x.diemTB);
}
//=====
void docDSSV(char fNameIn[], SV ds[], int &n)
{ //Đọc thông tin của 1 danh sách có n sinh viên từ file
    FILE *fi = fopen(fNameIn, "rt"); //rt: read text
    if (fi == NULL)
    {
        printf("Loi: mo file de doc du lieu.");
        getch();
        return;
    }
    fscanf(fi, "%d\n", &n); //Đọc dòng đầu tiên của file để lấy số lượng
    for (int i = 0; i < n; i++)
    {

```

```

        SV x;
        docTTSV(fi, x);
        ds[i] = x;
    }
    fclose(fi); //Đóng file vừa mở
}

//=====
void ghiDSSV(char fNameOut[], SV ds[], int n)
{ //Ghi thông tin của 1 danh sách có n sinh viên xuống file
    FILE *fo = fopen(fNameOut, "wt"); //wt: write text
    if (fo == NULL)
    {
        printf("Loi: mo file de ghi du lieu.");
        getch();
        return;
    }
    fprintf(fo, "%-5s%-15s%-40s%-10s\n", "STT", "MSSV", "HO VA TEN SINH VIEN", "DIEM TB");
    for (int i = 0; i < n; i++)
    {
        fprintf(fo, "%-5d", i + 1);
        ghiTTSV(fo, ds[i]);
    }
    fclose(fo); //Đóng file vừa mở
}

//=====
void xuatDSSV(SV ds[], int &n)
{ //Xuất danh sách có n sinh viên ra màn hình
    printf("%-5s%-15s%-40s%-10s\n", "STT", "MSSV", "HO VA TEN SINH VIEN", "DIEM TB");
    for (int i = 0; i < n; i++)
    {
        printf("%-5d", i + 1);
        xuatTTSV(ds[i]);
    }
}

//=====
//* Phần Hàm main
//=====
void main() {
    SV DS[MAXSIZE];
    int N;
    docDSSV(fNameIn_DSSV, DS, N);
    printf("DANH SACH SINH VIEN DOC TU FILE.\n");
    xuatDSSV(DS, N);
    printf("GHI DANH SACH SINH VIEN XUONG FILE\n");
    ghiDSSV(fNameOut_DSSV, DS, N);
    getch();
}

```

## Kết quả file output\_DSSV.txt

STT	MSSV	HO VA TEN	SINH VIEN	DIEM TB
1	2001200135	Nguyen	Thi Hong	7.25
2	2001200035	Trinh	Ho Hoang	5.15
3	2001200103	Vu	Bach Cong	4.73
4	2001201130	Le	Thai Bao	8.75
5	2001200125	Tran	Nguyen Thi Hong	7.73
6	2001201030	Nguyen	Thanh Cong	6.15

### Bổ sung thêm phần Menu vào chương trình:

```
//=====
/* Phần giao diện và Hàm main
//=====
void showMenu()
{
    printf("\n*****");
    printf("\n*          MENU          *");
    printf("\n*****");
    printf("\n* 1. Doc danh sach sinh vien tu file      *");
    printf("\n* 2. Xuat danh sach ra man hinh          *");
    printf("\n* 3. Ghi danh sach sinh vien xuong file   *");
    printf("\n* 0. Thoat chuong trinh                  *");
    printf("\n*****");
}
//=====
void process()
{
    SV DS[MAXSIZE];
    int N;
    int chonChucNang;
    do
    {
        showMenu();
        printf("\nBan hay chon mot chuc nang bat ky: ");
        scanf("%d", &chonChucNang);
        switch (chonChucNang)
        {
            case 1:
                docDSSV(fNameIn_DSSV, DS, N);
                printf("DANH SACH SINH VIEN DOC TU FILE.\n");
                xuatDSSV(DS, N);
                break;
            case 2:
                printf("DANH SACH SINH VIEN LA:\n");
                xuatDSSV(DS, N);
                break;
```

```

        case 3:
            printf("GHI DANH SACH SINH VIEN XUONG FILE\n");
            ghiDSSV(fNameOut_DSSV, DS, N);
        }
    } while (chonChucNang != 0);
}
//=====
void main()
{
    process();
}

```

### III. Bài thực hành trên lớp

**Bài 1.** Sử dụng cấu trúc SinhVien và mảng cấu trúc lưu trữ danh sách sinh viên trong bài tập hướng dẫn mẫu, bổ sung thêm một số yêu cầu sau:

- Tổ chức chương dạng menu cho phép sử dụng các chức năng theo yêu cầu.
- Thêm vào cấu trúc SV thông tin xếp loại. Kết quả xếp loại danh sách sinh viên dựa theo điểm trung bình.
  - Nếu  $dtb \geq 8$ : “Giỏi”
  - Nếu  $6.5 \leq dtb < 8$ : “Khá”
  - Nếu  $5.0 \leq dtb < 6.5$ : “Trung bình”
  - Ngược lại là “Yếu”.
- Sắp xếp danh sách nhân viên tăng dần/giảm dần theo điểm trung bình.
 

**Lưu ý:** dùng 3 giải thuật sắp xếp đã học.
- Tìm kiếm và in ra thông tin sinh viên có điểm trung bình cao nhất/thấp nhất.
- In ra danh sách các sinh viên có xếp loại giỏi và khá trong danh sách.
- Đếm xem có bao nhiêu SV họ “Nguyễn”.

**Bài 2.** Viết chương trình quản lý lớp học của một trường. Các thông tin của một lớp học như sau:


- Tên lớp.
  - Sĩ số.
  - Danh sách các sinh viên trong lớp với các thuộc tính như bài tập 1.
- Khai báo cấu trúc LopHoc theo mô tả trên.
  - Tạo file txt (dslop.txt) chứa thông tin n lớp trong trường.
  - Đọc file txt vào danh sách các lớp với thông tin yêu cầu.
  - Xuất danh sách lớp vừa tạo với đầy đủ thông tin yêu cầu.
  - In danh sách các lớp có trên 5 sinh viên có điểm trung bình loại giỏi.
  - Tìm và in thông tin lớp có nhiều sinh viên nhất.
  - Tìm và in thông tin sinh viên có điểm trung bình cao nhất trong các lớp.

#### IV. Bài thực hành về nhà

**Bài 3.** Xây dựng chương trình quản lý thông tin đặt vé của khách hàng trong một rạp chiếu phim. Mỗi khách hàng đặt vé cần cung cấp các thông tin sau:

- Họ và Tên (Chuỗi ký tự)
  - Số điện thoại (Chuỗi ký tự)
  - Số vé người lớn (số nguyên)
  - Số vé trẻ em (số nguyên)
  - Tên phim (Chuỗi ký tự)
  - Phòng chiếu (Chuỗi ký tự)
  - Xuất chiếu (Chuỗi ký tự)
  - Tiền phải trả (số nguyên)
- a. Khai báo cấu trúc `KhachHang` theo mô tả và mảng cấu trúc `dskh` lưu danh sách khách hàng đặt vé.
  - b. Đọc danh sách khách hàng từ file txt đã cho vào mảng `dskh`.
  - c. Xuất danh sách khách hàng từ mảng `dskh` ra màn hình.
  - d. Tính tiền mỗi khách hàng cần trả. Quy định giá vé như sau:  
Trẻ em: 20000 VNĐ/ 1 vé  
Người lớn: 40000 VNĐ/ 1 vé
  - e. Tính và xuất tổng doanh thu của rạp phim.
  - f. Tính tổng doanh thu của từng phim.
  - g. Xuất thông tin khách hàng đặt vé của phòng chiếu `x` và xuất chiếu `y` (với `x, y` được nhập từ bàn phím).
  - h. Sắp xếp danh sách khách hàng theo số tiền phải trả tăng dần (áp dụng 3 giải thuật sắp xếp đã học)
  - i. Sắp xếp danh sách tăng dần theo họ tên. (`a→z`)
  - j. Tìm khách hàng có họ tên “Nguyễn Văn An” với danh sách đã sắp xếp theo giải thuật binary search.

---HẾT---

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p><b>BUỔI 4: KỸ THUẬT XỬ LÝ</b></p> <p><b>MẢNG HAI CHIỀU</b></p>	
--	---	---

#### A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng hai cho các bài toán.
- Áp dụng các kỹ thuật cài đặt xử lý trên mảng hai chiều cho những bài toán thực tế.

#### B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

#### C. NỘI DUNG THỰC HÀNH

##### I. Tóm tắt lý thuyết

##### Kỹ thuật xử lý mảng hai chiều (sau đây gọi tắt là Ma trận)

- Duyệt từng dòng từ trên xuống dưới

```
for(i = 0; i < so_dong; i++)
    for (j = 0; j < so_cot; j++)
    {
        //Các lệnh Xử lý phần tử a[i][j] của ma trận
    }
```

- Tính tổng/tích và đếm các phần tử trong ma trận thỏa điều kiện

```
Tổng = 0/Tích=1/Đếm=1 //Lệnh khởi tạo
for (int i=0; i < so_dong; i++)
    for (int j=0; j < so_cot; j++)
        Nếu a[i][j] thỏa điều kiện thì
            Tổng = Tổng + a[i][j]/Tích=Tích* a[i][j]/Đếm=Đếm + 1
```

- Sắp xếp các phần tử tăng dần trên từng dòng

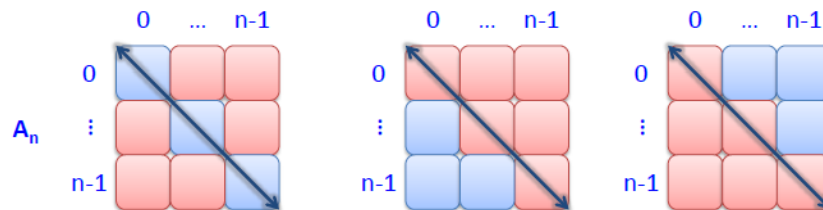
```
Cho k chạy từ dòng 0 đến hết số dòng trong ma trận
Cho i chạy từ cột 0 đến số cột -1 trong ma trận
Cho j chạy từ cột i+1 đến hết số cột trong ma trận
So sánh nếu A[k][i] < A[k][j] thì
    Đổi chỗ A[k][i] và A[k][j]
```

- Trong ma trận có các phần tử đặc biệt:

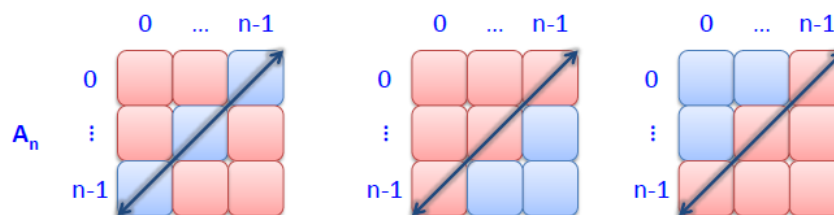
- Phần tử hoàng hậu: một phần tử  $a[i][j]$  được gọi là phần tử hoàng hậu nếu như nó lớn nhất trên dòng, lớn nhất trên cột và lớn nhất trên hai đường chéo chứa nó.
- Phần tử yên ngựa: một phần tử  $a[i][j]$  được gọi là điểm yên ngựa của ma trận nếu như nó là phần tử nhỏ nhất trên dòng  $i$  và lớn nhất trên cột  $j$  của ma trận hoặc nó là phần

tử lớn nhất trong dòng  $i$  và nhỏ nhất trên cột  $j$  của ma trận.

- **Phần tử cực đại:** Một phần tử được gọi là cực đại khi nó lớn hơn tất cả các phần tử lân cận của nó.
  - **Phần tử cực tiểu:** Một phần tử được gọi là cực tiểu khi nó nhỏ hơn tất cả các phần tử lân cận của nó.
- Các phần tử đặc biệt trên Ma trận vuông cấp  $n$ .
- + Đường chéo chính chứa các phần tử  $a[i][j]$  với  $i == j$
  - + Tam giác dưới đường chéo chính chứa các phần tử  $a[i][j]$  với:  $i > j$
  - + Tam giác trên đường chéo chính chứa các phần tử  $a[i][j]$  với:  $i < j$



- + Đường chéo phụ chứa các phần tử  $a[i][j]$  với  $(i + j) == (n - 1)$
- + Tam giác dưới của đường chéo phụ chứa các phần tử  $a[i][j]$  với:  $(i + j) > (n - 1)$
- + Tam giác trên của đường chéo phụ chứa các phần tử  $a[i][j]$  với:  $(i + j) < (n - 1)$



## II. Bài tập có hướng dẫn

Cho ma trận 2 chiều  $a$  có  $m$  dòng  $n$  cột chứa số nguyên.

a. Giải thuật tìm phần tử điểm yên ngựa của ma trận  $a$ .

```
void timDiemYenNgua (int a[][100], int m, int n) {
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            if (( a[i][j] == GTNN_DongI(a, m, n, i) && a[i][j] == GTLN_CotJ(a, m, n, j) )
                || ( a[i][j] == GTLN_DongI(a, m, n, i) && a[i][j] == GTNN_CotJ(a, m, n, j) ) )
                printf("%5d", a[i][j]);
}
```

b. Giải thuật tìm phần tử hoàng hậu của mảng  $a$

```
int xetDuongCheo (int a[][100], int dong, int cot, int i, int j) {
    int k, h; //k: dong, h: cot
    for (k=i, h=j; k >= 0 && h < cot; k--, h++)
        if (a[k][h] > a[i][j])
            return 0;
    for (k=i+1, h=j-1; k < dong && h >= 0; k++, h--)
```



```

        if (a[k][h] > a[i][j])
            return 0;
    for (k=i-1, h=j-1; k>=0 && h>=0; k--, h--)
        if (a[k][h] > a[i][j])
            return 0;
    for (k=i+1, h=j+1; k<dong && h<cot; k++, h++)
        if (a[k][h] > a[i][j])
            return 0;
    return 1;
}
int xetCotDong (int a[][100], int dong, int cot, int i, int j) {
    for (int k=0; k<dong; k++)
        if (a[k][j]>a[i][j])
            return 0;
    for (int k=0; k<cot; k++)
        if (a[i][k]>a[i][j])
            return 0;
    return 1;
}
int xetTongQuat (int a[][MAX], int dong, int cot, int i, int j) {
    if (xetCotDong(a, dong, cot, i, j)==1)
        if (xetDuongCheo(a, dong, cot, i, j)==1)
            return 1;
    return 0;
}

```

### III. Bài tập thực hành trên lớp.

**Bài 1.** Cho mảng 2 chiều a có m dòng, n cột chứa số nguyên. Viết các hàm xử lý sau, chú ý các ngoại lệ nếu có:

1. Tạo và xuất ma trận a chứa các phần tử ngẫu nhiên.

**Lưu ý:**

- Để lấy số ngẫu nhiên từ 0 đến k, ta sử dụng  $\text{rand}() \% (k+1)$
- Để lấy số ngẫu nhiên từ a đến b, ta sử dụng  $a + \text{rand}() \% (b-a+1)$

2. Tính và xuất tổng giá trị từng dòng.
3. Xuất phần tử lớn nhất trên từng cột.
4. Xuất các phần tử thuộc các đường biên trên, dưới, trái và phải.
5. Xuất các phần tử cực đại
6. Xuất các phần tử hoàng hậu.
7. Xuất các phần tử là điểm yên ngựa.
8. Xuất dòng chỉ chứa số chẵn.
9. Sắp xếp mảng a tăng theo từng dòng.

**Bài 2.** Cho ma trận vuông a cấp n, chứa số nguyên. Viết các hàm sau:

1. Tạo/xuất ma trận vuông a chứa số nguyên ngẫu nhiên có cấp  $n \geq 5$ .

2. Xuất các phần tử trên đường chéo chính.
3. Xuất các phần tử thuộc đường chéo song song với đường chéo chính.
4. Tìm phần tử max thuộc tam giác trên của đường chéo chính.
5. Sắp xếp ma trận tăng dần theo kiểu zic-zắc (tăng từ trái qua phải và từ trên xuống dưới)
6. Sắp xếp đường chéo chính tăng dần từ trên xuống dưới.



#### **Phần IV. Bài tập thực hành về nhà**

##### **Bài 3. Tiếp tục các hàm xử lý trên mảng 2 chiều a (bài 1)**

1. Xuất các cột chỉ chứa số lẻ
2. Tìm phần tử lớn nhất trong các phần tử trên biên của ma trận.
3. Trong ma trận có bao nhiêu phần tử có chữ số 2 xuất hiện trong các chữ số của nó.
4. Xuất các phần tử cực tiểu của ma trận.
5. Sắp xếp ma trận sao cho: các dòng có chỉ số lẻ thì tăng dần, còn các dòng có chỉ số chẵn thì giảm dần.
6. Sắp xếp ma trận sao cho: các cột có chỉ số lẻ thì giảm dần, còn các cột có chỉ số chẵn thì tăng dần.
7. Kiểm tra các giá trị trong ma trận có giảm dần theo cột và dòng (ziczac)
8. Liệt kê chỉ số các dòng chứa toàn giá trị chẵn
9. Liệt kê các dòng chứa giá trị giảm dần
10. Tìm giá trị xuất hiện nhiều nhất trong ma trận
11. Tìm các chữ số xuất hiện nhiều nhất trong ma trận
12. Liệt kê các cột có tổng nhỏ nhất trong ma trận
13. Hoán vị hai cột i và j trong ma trận
14. Hoán vị hai dòng k và l trong ma trận.

##### **Bài 4. Tiếp tục các hàm xử lý trên ma trận vuông a cấp n (bài 2)**

1. Sắp xếp đường chéo phụ tăng dần/giảm dần.
2. Sắp xếp ma trận sao cho: các dòng có chỉ số lẻ thì tăng dần, còn các dòng có chỉ số chẵn thì giảm dần.
3. Sắp xếp ma trận sao cho: các cột có chỉ số lẻ thì giảm dần, còn các cột có chỉ số chẵn thì tăng dần.
4. Sắp xếp mảng sao cho các phần tử trên các đường chéo chính và các đường chéo song song với đường chéo chính tăng dần.
5. Di chuyển các phần tử trong ma trận sao cho các phần tử chẵn nằm ở các dòng đầu mảng, các phần tử lẻ nằm ở các dòng cuối mảng.
6. Kiểm tra các phần tử trong ma trận có đối xứng nhau qua đường chéo chính không?

**--HẾT--**

<b>Trường ĐH CNTP TP. HCM</b> <b>Khoa Công nghệ thông tin</b> <b>Bộ môn Công nghệ phần mềm</b> <b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b>	<b>BUỔI 5.</b> <b>ÔN TẬP - KIỂM TRA LẦN 1</b>	
---	--	--

#### A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng hai cho các bài toán.
- Áp dụng các kỹ thuật cài đặt xử lý con trỏ, xử lý ngoại lệ, cấu trúc, mảng một chiều, hai chiều cho những bài toán thực tế.

#### B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

#### C. NỘI DUNG THỰC HÀNH

##### ÔN TẬP - ĐỀ KIỂM TRA MẪU:

<b>KIỂM TRA LẦN 1</b> <b>MÔN: TH KỸ THUẬT LẬP TRÌNH – THỜI GIAN: 120 phút</b> <b>(SV không được sử dụng tài liệu)</b>	
<b>Câu 1.</b> Tạo chương trình theo cấu trúc (khai báo thư viện, khai báo cấu trúc, khai báo hàm con, hàm main, thân hàm con), có hàm hiển thị danh sách bài thực hiện được và cho người dùng lựa chọn từng bài cần thực hiện. Trong chương trình cần xử lý ngoại lệ nếu có	<b>(1 điểm)</b>
<b>Câu 2.</b> Tạo mảng 1 chiều a dạng con trỏ, chứa số thực. Viết hàm:	
2.1. Tạo và xuất mảng a chứa giá trị ngẫu nhiên	(0.5 điểm)
2.2. Xuất giá trị và địa chỉ các phần tử của mảng a.	(1 điểm)
2.3. Tìm phần tử chẵn nhỏ nhất trong mảng a.	(1 điểm)
<b>Câu 3.</b> Cho ma trận vuông a cấp n. Viết hàm:	
3.1. Tạo và xuất mảng a chứa giá trị ngẫu nhiên.	(0.5 điểm)
3.2. Sắp xếp trên từng dòng sao cho các số chẵn tăng dần.	(1 điểm)
3.2. Sắp xếp đường chéo phụ giảm dần theo giải thuật SelectionSort	(1 điểm)
<b>Câu 4.</b>	
4.1. Tạo cấu trúc chứa thông tin Sản phẩm gồm thông tin:	(0.5 điểm)

- Mã sản phẩm (5 ký tự)
- Tên sản phẩm (10 ký tự)
- Đơn giá (số thực)
- Số lượng trong kho (số nguyên)


4.2. Nhập hoặc đọc dữ liệu từ file tạo mảng 1 chiều dssp chứa thông tin danh sách sản phẩm. **(1 điểm)**

4.3. Xuất thông tin danh sách. **(0.5 điểm)**

4.3. Sắp xếp danh sách tăng dần theo đơn giá. **(1 điểm)**

4.4. Xuất thông tin các sản phẩm có mã số “A01” theo giải thuật BinarySort **(1 điểm)**

**--HẾT--**

<p>Trường ĐH CNTP TP. HCM</p> <p>Khoa Công nghệ thông tin</p> <p>Bộ môn Công nghệ phần mềm</p> <p>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</p>	<p><b>BUỔI 6. XỬ LÝ CHUỖI</b></p>	
--	-----------------------------------	---

## A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý chuỗi ký tự cơ bản và sử dụng các hàm trong thư viện **string.h** vào các bài toán.
- Áp dụng các kỹ thuật cài đặt xử lý trên chuỗi vào những bài toán thực tế.

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

### I. Tóm tắt lý thuyết

#### 1. Khái niệm

Chuỗi ký tự là một dãy các phần tử, mỗi phần tử là một ký tự.

**Lưu ý:** Chuỗi ký tự được kết thúc bằng ký tự ‘\0’, gọi là ký tự kết thúc chuỗi. Do đó khi khai báo độ dài của chuỗi luôn luôn khai báo dư một phần tử để chứa ký tự ‘\0’.

Ví dụ 1.1. `char s[5] = “CNTT”;` //khai báo chuỗi có 5 phần tử kiểu char và gán dãy ký tự CNTT và chuỗi được lưu trữ như sau:

‘C’	‘N’	‘T’	‘T’	‘\0’
s[0]	s[1]	s[2]	s[3]	s[4]

**Lưu ý:** Chuỗi rỗng là chuỗi chưa có ký tự nào trong mảng và được ký hiệu “”.

#### 2. Khai báo chuỗi

Để khai báo một chuỗi, ta có 2 cách khai báo sau:

##### Cách 1: Con trỏ hằng

**char <Tên\_chuỗi>[<Số ký tự tối đa của chuỗi>;**

Ví dụ 2.1. `char hoten[50];`

##### Cách 2: Con trỏ

**char \*<Tên\_chuỗi>;**

Ví dụ 2.2. `char *hoten;`

Ta có thể vừa khai báo vừa gán giá trị cho chuỗi như sau:

Ví dụ 2.3. `char monhoc[50] = “Ky thuat lap trinh”;`

`char s[10] = {‘K’, ‘T’, ‘L’, ‘T’, ‘\0’};`

**Lưu ý:** Việc gán nhiều giá trị hằng chuỗi như việc sử dụng dấu ngoặc kép (“”) chỉ hợp lệ khi khởi tạo mảng, tức là lúc khai báo mảng. Các biểu thức trong chương trình như:

```
s = "Hello";
```

```
s[] = "Hello";
```

là không hợp lệ, cả câu lệnh dưới đây cũng vậy:

```
s = {'K', 'T', 'L', 'T', '\0'};
```

Vì về trái của một lệnh gán chỉ có thể là một phần tử của mảng chứ không thể là cả mảng, chúng ta có thể gán một chuỗi kí tự cho một mảng kiểu char sử dụng một phương pháp như sau:

```
s[0] = 'K';      s[1] = 'T';      s[2] = 'L';      s[3] = 'T';      s[4] = '\0';
```

Nhưng đây không phải là một phương pháp thực tế. Để gán giá trị cho một chuỗi kí tự, chúng ta có thể sử dụng các hàm kiểu **strcpy**, hàm này được định nghĩa trong thư viện **string.h** và được giải thích cụ thể trong phần 3.

### 3. Các thao tác xử lý chuỗi

#### a. Nhập dữ liệu cho chuỗi

Sử dụng hàm `gets()` (thuộc thư viện `<stdio.h>`) được coi là cách thức đơn giản và phổ biến nhất để ta tiến hành nhập liệu cho chuỗi. Cú pháp, sử dụng hàm này như sau: `gets(Tên_chuỗi);`

Ví dụ 3.1. Nhập dữ liệu từ bàn phím cho chuỗi `s` có kích thước tối đa 30 ký tự:

```
char s[30];
```

```
printf("\n Moi ban nhap mot chuoi: ");
```

```
gets(s);
```

Ngoài ra, ta cũng có thể sử dụng hàm `scanf()` với chuỗi định dạng “%s” để nhập dữ liệu cho chuỗi. Tuy nhiên, cách nhập này không cho phép nhập chuỗi có chứa khoảng trắng.

#### b. Xuất dữ liệu cho chuỗi

Có 2 cách xuất chuỗi ra màn hình như sau:

**Cách 1:** Sử dụng hàm `printf` với đặc tả “%s”

```
char monhoc[50] = "Tin hoc co so A";
```

```
printf("%s", monhoc);    //Không xuống dòng
```

**Cách 2:** Sử dụng hàm `puts`

```
char monhoc[50] = "Tin hoc co so A";
```

```
puts(monhoc);    //Tự động xuống dòng
```

**c. Các hàm xử lý chuỗi trong thư viện <string.h>**

STT	TÊN HÀM	CHỨC NĂNG	VÍ DỤ
1	int <b>strlen</b> (char s[]);	Trả về độ dài của chuỗi s.	char *s = "Borland International"; printf("Do dai s: %d", strlen(s)); <b>Kết quả: Do dai s: 21</b>
2	<b>strcpy</b> (char dest[], char src[]);	Sao chép nội dung chuỗi src vào chuỗi dest.	char dest[10]; char *src = "abcdefghi"; strcpy(dest, src); printf("%s\n", dest); <b>Kết quả: abcdefghi</b>
3	<b>strncpy</b> (char dest[], char src[], int n);	Chép n ký tự từ chuỗi src sang chuỗi dest. Nếu chiều dài src < n thì hàm sẽ điền khoảng trắng cho đủ n ký tự vào dest.	char dest[4]; char *src = "abcdefghi"; strncpy(dest, src, 3); printf("%s\n", dest); <b>Kết quả: abc</b>
4	<b>strcat</b> (char s1[], char s2[]);	Nối chuỗi s2 vào chuỗi s1.	char *s1 = "Khoa"; char *s2 = "CNTT"; strcat(s1, s2); printf("%s\n", s1); <b>Kết quả: Khoa CNTT</b>
5	<b>strncat</b> (char s1[], char s2[], int n)	Nối n ký tự đầu tiên của chuỗi s2 vào chuỗi s1.	char *s1 = "Khoa"; char *s2 = "CNTT"; strncat(s1, s2, 2); printf("%s\n", s1); <b>Kết quả: Khoa CN</b>
6	int <b>strcmp</b> (char s1[], char s2[])	So sánh 2 chuỗi s1 và s2 theo nguyên tắc thứ tự từ điển. Phân biệt chữ hoa và thường. Trả về: • 0: nếu s1 bằng s2. • >0: nếu s1 lớn hơn s2. • <0: nếu s1 nhỏ hơn s2.	char *s1 = "abcd"; char *s2 = "abCD"; if(strcmp(s1, s2)==0) printf("Giống nhau"); else printf("Khác nhau"); <b>Kết quả: Khác nhau</b>
7	int <b>strncmp</b> (char s1[], char s2[], int n)	Tương tự như strcmp(), nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi.	char *s1 = "abcd"; char *s2 = "abef"; if(strncmp(s1, s2, 2)==0) printf("Giống nhau"); else printf("Khác nhau"); <b>Kết quả: Giống nhau</b>

STT	TÊN HÀM	CHỨC NĂNG	VÍ DỤ
8	int <b>strcmp</b> (char s1[], char s2[])	Tương tự như strcmp(), nhưng không phân biệt hoa thường.	char *s1 = "abcd"; char *s2 = "abCD"; if(strcmp(s1, s2)==0) printf("Giống nhau"); else printf("Khác nhau"); <b>Kết quả: Giống nhau</b>
9	int <b>strnicmp</b> (char s1[], char s2[], int n);	Tương tự như strcmp(), nhưng chỉ so sánh n ký tự đầu tiên của hai chuỗi.	char *s1 = "aBcd"; char *s2 = "AbeF"; if(strnicmp(s1, s2, 2)==0) printf("Giống nhau"); else printf("Khác nhau"); <b>Kết quả: Giống nhau</b>
10	char * <b>strchr</b> (char s[], char c);	Tìm lần xuất hiện đầu tiên của ký tự c trong chuỗi s. Trả về: <ul style="list-style-type: none"> <li>• NULL: nếu không có.</li> <li>• Địa chỉ c: nếu tìm thấy.</li> </ul>	char s[15]; char *ptr, c = 'm'; strcpy(s, "Vi du tim ky tu"); ptr = strchr(s, c); if (ptr) printf("Ky tu %c tai: %d", c, ptr-s); else printf("Khong tim thay"); <b>Kết quả: Ky tu m tai: 8</b>

## II. Bài tập có hướng dẫn

**Bài 4.** Nhập vào một chuỗi từ bàn phím. Tính và xuất độ dài của chuỗi (không dùng hàm trong thư viện string.h).

### Hướng dẫn:

#### Phân tích:

- Input: chuỗi s.
- Output: n (là chiều dài chuỗi s).

#### Chương trình mẫu:

```
#include <stdio.h>
#include <conio.h>
int length(char s[])
{
    int i = 0;
    while(s[i] != '\0')
        i++;
}
```



```

        return i;
    }
    void main()
    {
        //sinh viên khai báo chuỗi, gọi và in giá trị cho hàm length
        getch();
    }

```

**Bài 5.** Nhập vào 1 chuỗi và 1 ký tự c, kiểm tra ký tự c có trong chuỗi hay không, nếu có đưa ra số lần xuất hiện của ký tự đó trong chuỗi.

**Hướng dẫn:**

**Phân tích:**

- Input: chuỗi s và ký tự c.
- Output: Trả lời ký tự c có trong chuỗi s hay không.

**Chương trình mẫu:**

```

#include <stdio.h>
#include <conio.h>
#include <string.h>
int kiemTra(char s[], char c) {
    int dem = 0;
    //for(int i = 0; i < strlen(s); i++) {
    for(int i = 0; s[i] != '\0'; i++) {
        if(s[i] == c)
            dem++;
    }
    return dem;
}
void main() {
    /*sinh viên khai báo chuỗi và 1 ký tự, gọi và in giá trị cho hàm kiemtra*/
    getch();
}

```

**Bài 6.** Nhập vào một chuỗi từ bàn phím. Đếm có bao nhiêu từ trong chuỗi vừa nhập (giả sử chuỗi nhập vào đúng chuẩn).

**Hướng dẫn:**

**Phân tích:**

- Input: chuỗi s.
- Output: n (là số từ của chuỗi s).

**Chương trình mẫu:**

```

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

```

```

int demTu(char s[]) {
    int dem = 0;
    //for(int i = 0; i < strlen(s); i++) {
    for(int i = 0; s[i] != '\0'; i++) {
        if(s[i] == ' ')
            dem++;
    }
    return dem;
}

void main() {
    //sinh viên tự viết code gọi hàm và xuất kết quả
    getch();
}

```

### III. Bài thực hành trên lớp

**Bài 1.** Kiểm tra xem chuỗi s có chứa toàn ký số hay không?

**Bài 2.** Viết hàm đổi những kí tự đầu tiên của mỗi từ thành chữ in hoa và những từ không phải đầu câu sang chữ thường.

**Bài 3.** Viết hàm xóa những khoảng trắng thừa trong chuỗi (hay còn gọi là chuẩn hóa chuỗi).

**Bài 4.** Viết hàm tìm kiếm tên trong một chuỗi họ tên. Nếu có thì xuất ra là tên chuỗi họ tên, ngược lại thông báo tên không tồn tại.

**Bài 5.** Viết hàm cắt chuỗi họ tên thành 2 chuỗi con: họ lót và tên.

Ví dụ: chuỗi họ tên là: “Nguyễn Văn Anh” cắt ra 2 chuỗi là chuỗi họ lót: “Nguyễn Văn”, chuỗi tên là: “Anh”.

**Bài 6.** Nhập vào một danh sách sinh viên và hiển thị danh sách sinh viên ra màn hình.

Áp dụng giải thuật Brute Force, tìm và xuất vị trí các chuỗi P trong T,

a) Với T là chuỗi nhập vào từ bàn phím.

b) Với T là chuỗi văn bản đọc từ file text.

### IV. Bài thực hành về nhà

**Bài 7.** Kiểm tra xem chuỗi nhập vào có đối xứng hay không?

**Bài 8.** Nhập vào chuỗi s1 và s2, cho biết vị trí xuất hiện của chuỗi s2 trong s1 (nếu có). Nếu s2 không có trong s1, thực hiện nối s2 vào cuối s1.

**Bài 9.** Nhập vào chuỗi str, chuỗi cần chèn strInsert và vị trí cần chèn vt. Hãy chèn chuỗi strInsert vào chuỗi str tại vị trí vt.

**Bài 10.** Nhập một chuỗi bất kì, yêu cầu nhập 1 kí tự muốn xóa. Thực hiện xóa tất cả những kí tự đó trong chuỗi.

**Bài 11.** Nhập vào 1 mảng các chuỗi. Tìm xuất ra màn hình những chuỗi chứa toàn ký tự số.

**Bài 12.** Cho 2 chuỗi s1 và s2 là 2 từ (chỉ chứa chữ cái hoặc ký số). Tính xem cần thực hiện bao nhiêu phép biến đổi để chuyển chuỗi s1 thành s2. Biết rằng có 3 phép biến đổi


- Thêm ký tự
- Xóa ký tự
- Đổi ký tự

Ví dụ: xét 2 từ s1= “Kitten” và s2=“Sitting”

Cần thực hiện 3 phép đổi để chuyển s1 thành s2. Đó là:

- Đổi ‘K’ thành ‘S’
- Đổi ‘e’ thành ‘i’
- Thêm ‘g’ vào cuối.

**---HẾT---**

<p><b>Trường ĐH CNTP TP. HCM</b></p> <p><b>Khoa Công nghệ thông tin</b></p> <p><b>Bộ môn Công nghệ phần mềm</b></p> <p><b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b></p>	<p><b>BUỔI 7.</b></p> <p><b>KỸ THUẬT ĐỆ QUY</b></p>	
--	---	---

## A. MỤC TIÊU:

Thực hành kỹ thuật đệ quy dạng viết hàm cơ bản:

- Đệ quy tuyến tính
- Đệ quy nhị phân
- Đệ quy phi tuyến
- Đệ quy lồng
- Đệ quy tương hỗ

Thực hành kỹ thuật khử đệ quy.

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

### I. Bài tập có hướng dẫn

**1. Đệ quy tuyến tính:** là hàm bên trong thân hàm có duy nhất một lời gọi đệ quy

Ví dụ 1: Viết hàm đệ quy tính  $X^n$  với  $X$  là số thực,  $n$  là số nguyên được xác định như sau:

$$X^n = \begin{cases} 1, & n = 0 \\ X * X^{n-1}, & n \neq 0 \end{cases}$$

```
float tinhLuyThua(float x, int n)
```

```
{
    float kq;
    if(n == 0)                //Điều kiện dừng
        kq = 1;
    else
        kq = x*tinhLuyThua(x, n-1); //Gọi đệ quy
    return kq;
}
```

**2. Đệ quy nhị phân:** là hàm trong thân hàm có 2 lời gọi đệ quy

Ví dụ 2: Viết hàm đệ quy tính các tổ hợp chập  $k$  của  $n$  phần tử theo công thức sau:

$$C_n^k = \begin{cases} 1, & \text{nếu } k = 0 \text{ hoặc } k = n \\ C_{n-1}^{k-1} + C_{n-1}^k, & \text{nếu } 0 < k < n \end{cases}$$

```
int toHop (int n, int k)
```

```
{
    if(k==0 || k==n)
        return 1;                // Phần cơ sở
    else
```

```

        return toHop(n-1, k-1) + toHop(n-1, k); //2 lần gọi hàm đệ quy
    }

```

3. **Đệ quy phi tuyến:** là hàm có lời gọi hàm chính nó được đặt trong vòng lặp

Ví dụ 3: Viết hàm đệ quy tính  $S(n)$  với  $n$  là số nguyên theo công thức sau:

$$S(n) = \begin{cases} 1, & \text{nếu } n = 1 \\ S(1) + S(2) + \dots + S(n-1), & \text{nếu } n > 1 \end{cases}$$

```

int Sn(int n)
{
    if(n == 1)
        return 1;
    int kq = 1;
    for (int i = 1; i < n; i++)
        kq += Sn(i); // hàm đệ quy được đặt trong vòng lặp
    return kq;
}

```

4. **Đệ quy lồng:** là hàm Tham số trong lời gọi đệ quy là một lời gọi đệ quy

Ví dụ 4: Viết hàm tính giá trị Ackermann's theo công thức sau:

$$Acker(m, n) = \begin{cases} n + 1, & \text{nếu } m = 0 \\ Acker(m - 1, 1), & \text{nếu } n = 0 \wedge m > 0 \\ Acker(m - 1, Acker(m, n - 1)), & \text{nếu } m > 0 \wedge n > 0 \end{cases}$$

Hàm số Ackermann là một hàm thực được mang tên nhà toán học người Đức Wilhelm Ackermann (1896–1962). Hàm Ackermann đôi khi còn được gọi là hàm Ackermann-Peter

```

int ackerman (int m, int n)
{
    if (m == 0)
        return (n+1);
    if (n == 0)
        return ackerman(m-1, 1);
    return ackerman(m-1, ackerman(m, n-1));
}

```

5. **Đệ quy tương hỗ:** Trong đệ quy tương hỗ có 2 hàm, và trong thân của hàm này có lời gọi của hàm kia, điều kiện dừng và giá trị trả về của cả hai hàm có thể giống nhau hoặc khác nhau

Yêu cầu: Viết hàm đệ quy tính  $X(n)$ ,  $Y(n)$  với  $n$  là số nguyên theo công thức sau:

$$X(0) = Y(0) = 1$$

$$X(n) = X(n-1) + Y(n-1)$$

$$Y(n) = X(n-1) * Y(n-1)$$

```

int X(int n)
{
    if (n == 0)
        return 1;
}

```

```

    else
        return X(n-1) + Y(n-1);    //Hàm X(n) có lời gọi hàm đến Y(n)
}
int Y(int n)
{
    if(n == 0)
        return 1;
    else
        return X(n-1) * Y(n-1);    //Hàm Y(n) có lời gọi hàm đến X(n)
}

```

**Khử đệ qui:** Giải thuật giải bài toán bằng đệ quy thường gọn, dễ hiểu. Tuy nhiên, bài toán xử lý bằng giải thuật đệ quy thường tốn nhiều bộ nhớ và thời gian. Nên mọi giải thuật đệ qui đều thay thế bằng một giải thuật không đệ quy. Khử đệ qui để có chương trình không đệ quy. Có 2 cách khử đệ quy: khử đệ quy bằng vòng lặp, khử đệ quy bằng Stack

Yêu cầu: Viết hàm đệ qui tính  $X^n$  với  $X$  là số thực,  $n$  là số nguyên được xác định như sau:

$$X^n = \begin{cases} 1, & n = 0 \\ X * X^{n-1}, & n = 0 \end{cases}$$

```

float tinhLuyThua(float x, int n)
{
    float kq = 1;
    for(int i = 1; i <= n; i++)
    {
        kq *= x;
    }
    return kq;
}

```

## II. Bài tập thực hành trên lớp

**Bài 1.** Viết hàm tính các biểu thức  $S(n)$  theo 2 cách đệ quy và khử đệ quy (nếu có thể), với  $n$  là số nguyên dương nhập từ bàn phím:

$$S(n) = 1 + 2 + 3 + \dots + n.$$

$$S(n) = \sqrt{5 + \sqrt{5 + \dots + \sqrt{5 + \sqrt{5}}}} \text{ có } n \text{ dấu căn.}$$

$$S(n) = \frac{1}{2} + \frac{2}{3} + \dots + \frac{n}{n+1}$$

$$S(n) = 1 + \frac{1}{3} + \frac{1}{5} + \dots + \frac{1}{2n+1}$$

$$S(n) = 1.2 + 2.3 + 3.4 + 4.5 + \dots + n.(n+1)$$

$$S(n) = \frac{1.2!}{2 + \sqrt{3}} + \frac{2.3!}{3 + \sqrt{4}} + \frac{3.4!}{4 + \sqrt{5}} + \dots + \frac{n.(n+1)!}{(n+1) + \sqrt{(n+2)}}$$

$$S(n) = \frac{1+\sqrt{1+2}}{2+\sqrt{3!}} + \frac{2+\sqrt{2+3}}{3+\sqrt{4!}} + \frac{3+\sqrt{3+4}}{4+\sqrt{5!}} + \dots + \frac{n+\sqrt{n+n+1}}{(n+1)+\sqrt{(n+2)!}}$$

**Bài 2.** Viết hàm tìm ước chung lớn nhất của 2 số nguyên dương  $a, b$ .

Gợi ý: Nếu  $a > b$  thì  $\text{UCLN}(a, b) = \text{UCLN}(b, a-b)$ ,

ngược lại  $\text{UCLN}(a, b) = \text{UCLN}(a, b-a)$ .

**Bài 3.** Viết hàm tìm giá trị phần tử thứ  $n$  của cấp số cộng có hạng đầu là  $a$ , công sai là  $r$ :

$$U_n = \begin{cases} a, & \text{nếu } n = 1 \\ U_{n-1} + r, & \text{nếu } n > 1 \end{cases}$$

**Bài 4.** Cho dãy số  $A_n$  theo các công thức quy nạp như sau, hãy viết chương trình tính số hạng thứ  $n$ , với  $n$  là số nguyên dương:

a.  $A_0 = 1$  ;  $A_1 = 0$  ;  $A_2 = -1$  ;  $A_n = 2A_{n-1} - 3A_{n-2} - A_{n-3}$ .

b.  $A_1 = 1$  ;  $A_2 = 2$  ;  $A_3 = 3$  ;  $A_{n+3} = 2A_{n+2} + A_{n+1} - 3A_n$

Viết chương trình tính số hạng thứ  $n$ .

**Bài 5.** Cho dãy số  $x_n$  được định nghĩa như sau: `

$$x_0 = 1 ; x_1 = 2 ; x_n = nx_0 + (n-1)x_1 + \dots + x_{n-1}$$

Viết hàm đệ quy tính  $x_n$  với  $n \geq 0$ .

**Bài 6.** Đếm số chữ số nguyên dương  $n$ .

**Bài 7.** Tìm số Fibonacci thứ  $n$ . Biết rằng:

$$\text{Fibonacci}(n) = \begin{cases} 1 & \text{với } n \leq 2 \\ \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2) & \text{với } n > 2 \end{cases}$$

**Bài 8.** Xuất dãy số Fibonacci mà giá trị các số nhỏ hơn  $m$ .

### III. Bài tập về nhà

**Bài 9.** Viết hàm tính các biểu thức  $S(n)$  theo 2 cách đệ quy và khử đệ quy (nếu có thể), với  $n$  là số nguyên dương nhập từ bàn phím:

a.  $S(n) = \frac{1}{1.2.3} + \frac{1}{2.3.4} + \frac{1}{3.4.5} + \dots + \frac{1}{n.(n+1).(n+2)}$

b.  $S(n) = 1^2 + 2^2 + \dots + n^2$

c.  $S(n) = 1 + (1 + 2) + (1 + 2 + 3) + \dots + (1 + 2 + 3 + \dots + n)$

d.  $S(n) = -\frac{1+2}{2!} + \frac{3+4}{4!} - \frac{5+6}{6!} \dots + (-1)^n \frac{(2n-1)+(2n)}{(2n)!}$

**Bài 10.** Viết hàm xuất dãy có số Fibonacci thuộc đoạn từ  $[m, n]$ , biết rằng số Fibonacci là số có dạng:

$$\text{Fibonacci}(n) = \begin{cases} 1 & \text{với } n \leq 2 \\ \text{Fibonacci}(n-1) + \text{Fibonacci}(n-2) & \text{với } n > 2 \end{cases}$$

Ví dụ: nhập  $m = 5, n = 20 \rightarrow$  dãy số Fibonacci có 3 số thỏa là: 5 8 13

**Bài 11.** Viết hàm tìm số Fibonacci lớn nhất nhưng nhỏ hơn số nguyên dương  $n$  cho trước theo 2 cách đệ quy và khử đệ quy.

Ví dụ: nhập  $n = 15 \rightarrow$  số Fibonacci lớn nhất nhỏ hơn 15 là 13.

**Bài 12.** Viết hàm tính số hạng thứ  $n$  của 2 dãy sau:

$$x_0 = 1, y_0 = 0,$$

$$x_n = x_{n-1} + y_{n-1} \text{ với mọi } n > 0$$

$$y_n = 3x_{n-1} + 2y_{n-1} \text{ với mọi } n > 0$$

**Bài 13.** Viết hàm tìm giá trị phần tử thứ  $n$  của cấp số nhân có hạng đầu là  $a$ , công bội là  $q$ :

$$U_n = \begin{cases} a, & \text{nếu } n = 1 \\ qU_{n-1}, & \text{nếu } n > 1 \end{cases}$$

**Bài 14.** Viết hàm tính biểu thức  $U(n)$  sau, với  $n$  là số nguyên dương nhập từ bàn phím:

$$U_n = \begin{cases} n, & \text{với } n < 6 \\ U_{n-5} + U_{n-4} + U_{n-3} + U_{n-2} + U_{n-1} & \text{với } n \geq 6 \end{cases}$$

**Bài 15.** Dãy An được cho như sau:

$$A_1=1 ;$$

$$A_n = n*(A_1+A_2+ \dots + A_{n-1}).$$

Viết hàm tính An sử dụng kỹ thuật đệ quy.

**Bài 16.** Với mỗi  $n \geq 1$ , số Yn được tính như sau:

$$Y_1=1 ; Y_2=2 ; Y_3=3 ; Y_n = Y_{n-1} + 2Y_{n-2} + 3Y_{n-3} \text{ nếu } n \geq 4$$

Viết hàm tính Yn bằng 2 cách đệ quy.

**Bài 17.** Với mỗi  $n \geq 1$ , số Xn được tính như sau:

$$X_1=1 ; X_2=1 ; X_n = X_{n-1} + (n-1)X_{n-2} \text{ với } n \geq 3$$

Viết hàm tính Xn bằng cách đệ quy

**Bài 18.** Cho dãy số xn được định nghĩa như sau:

$$x_0 = 1 ; x_1 = 2 ; x_n = nx_0 + (n-1)x_1 + \dots + x_{n-1}$$

Viết hàm đệ quy tính xn với  $n \geq 0$ .

**Bài 19.** Dãy An được cho như sau:

$$A_1=1$$


$$A_{2n} = n + A_n + 2$$

$$A_{2n+1} = n^2 + A_n.A_{n+1} + 1$$

Viết hàm đệ quy tính An

--HẾT--



<b>Trường ĐH CNTP TP. HCM</b> <b>Khoa Công nghệ thông tin</b> <b>Bộ môn Công nghệ phần mềm</b> <b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b>	<b>BUỔI 8.</b> <b>KỸ THUẬT ĐỆ QUY (tt)</b>	
---	---	---

## A. MỤC TIÊU:

- Thực hành kỹ thuật đệ quy ứng dụng giải các bài toán thông dụng theo 3 bước: thông số hóa bài toán, xác định phần cơ sở, xác định phần đệ quy.

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

### I. Tóm tắt lý thuyết

- Bước 1: Thông số hóa bài toán.
- Bước 2: Tìm các trường hợp cơ bản (phần neo) cùng giải thuật tương ứng cho các trường hợp này.
- Bước 3: Tìm giải thuật giải trong trường hợp tổng quát (phần đệ quy) bằng cách phân rã bài toán theo kiểu đệ quy.

### II. Bài tập có hướng dẫn

**Bài 1.** Tìm phần tử lớn nhất trong mảng 1 chiều a có n phần tử là số nguyên.

a:  $a[0], a[1], a[2] \dots a[n-2], a[n-1]$

Phân tích bài toán:

- Bước 1: Thông số hóa bài toán  
n chính là kích thước dữ liệu và là thông số tổng quát cho bài toán.
- Bước 2: Xác định phần cơ bản (neo)  
Nếu  $n=1$  thì mảng a có 1 phần tử  $a[0]$ , và nó cũng chính là phần tử lớn nhất của mảng.
- Bước 3: Xác định phần đệ quy ( $n \geq 2$ )  
 $n=2$ : max của mảng a chính là  $\max(\max(a[0]), a[1])$   
 $n=3$ : max của mảng a chính là  $\max(\max(\max(a[0]), a[1]), a[2])$   
...

$n=n$ : max của mảng a chính là  $\max(\max(a[0] \dots a[n-2]), a[n-1])$

```
int timMax_Dequy (int a[], int n)
{
    if (n==1)
        return a[0];
    else
        return max2so(timMax_Dequy(a, n-1), a[n-1])
}
int max2so (int x, int y)
{
    return x>y?x:y;
}
```

### III. Bài tập thực hành trên lớp

**Bài 2.** Cho mảng 1 chiều a chứa n số nguyên. VIết các hàm xử lý sau theo kỹ thuật đệ quy:

- Tính tổng các phần tử chẵn của a.
- Tìm kiếm số x trên a theo thuật toán tìm kiếm nhị phân bằng kỹ thuật đệ quy.
- Tìm max chẵn trong a
- Tính tổng lẻ trong a
- Xuất các số ở vị trí lẻ

**Bài 3.** Đếm số chữ số của số nguyên dương n.

**Bài 4.** Đếm số chữ số chẵn của số nguyên dương n.

**Bài 5.** Xuất tất cả các hoán vị của 1 dãy phần tử a có n phần tử.

Gợi ý:

- **Thông số hóa:** số phần tử n của dãy là giá trị xác định kích thước dữ liệu bài toán.
- Phân tích tìm phần neo và đệ quy:
  - Nếu dãy A có  $n=1$  phần tử  $A[1]=a$  thì số hoán vị chỉ có 1 là: a.
  - Nếu dãy A có  $n=2$  phần tử:  $A[1]=a, A[2]=b$  thì số hoán vị là 2 dãy sau: a b; b a.
  - Nếu dãy A có  $n=3$  phần tử:  $A[1]=a, A[2]=b, A[3]=c$  thì các hoán vị của dãy A là 6 dãy sau: a b c; b a c; a c b; c a b; b c a; c b a;

Gọi hàm  $hoanVi(A, m)$  là hàm xuất tất cả các dạng hoán vị khác nhau của A có được bằng cách hoán vị m thành phần đầu của dãy A.

Phần cơ sở bài toán:  $m=1$ , hàm  $HoanVi(A, 1)$  chỉ có 1 cách xuất A.

Phân rã tổng quát:

```
HoanVi(A, m)  $\equiv$  { Swap (A[m], A[m]) ; HoanVi (A, m-1) ;
                    Swap (A[m], A[m-1] ; HoanVi (A, m-1) ;
                    Swap (A[m], A[m-2] ; HoanVi(A,m-1) ;
                    ...
                    Swap (A[m], A[2] ; HoanVi(A,m-1) ;
                    Swap (A[m], A[1] ; HoanVi(A,m-1) ;
}
```

**Bài 6.** Bài toán chia thưởng

Tìm số cách chia  $m$  vật (phần thưởng) cho  $n$  đối tượng (học sinh) có thứ tự.

Gọi Distribute là hàm tính số cách chia, khi đó Distribute là hàm có 2 tham số nguyên  $m$  và  $n$  (Distribute( $m, n$ )).

Gọi  $n$  đối tượng theo thứ tự xếp hạng  $1, 2, 3, \dots, n$ ;  $S_i$  là số vật mà đối tượng thứ  $i$  nhận được.

Khi đó các điều kiện ràng buộc cho cách chia là:

$$S_i \geq 0$$

$$S_1 \geq S_2 \geq \dots \geq S_n$$

$$S_1 + S_2 + \dots + S_n = m$$

**IV. Bài tập thực hành về nhà**

**Bài 7.** Lãi suất tiền VNĐ gởi  $x\%$  mỗi năm, ban đầu người ta gởi  $n$  triệu đồng. Viết hàm đệ quy tính sau  $m$  năm sẽ có bao nhiêu tiền VNĐ (gồm cả vốn lẫn lãi)?

**Bài 8.** Cho số nguyên dương  $n$ .

- Hãy tìm chữ số đầu tiên của  $n$ .
- Hãy tìm chữ số đảo ngược của số nguyên dương  $n$ .
- Tìm chữ số lớn nhất của số nguyên dương  $n$ .
- Tìm chữ số nhỏ nhất của số nguyên dương  $n$ .
- Hãy kiểm tra số nguyên dương  $n$  có toàn chữ số lẻ hay không?
- Hãy kiểm tra số nguyên dương  $n$  có toàn chữ số chẵn hay không?

**Bài 9.** Viết hàm đệ quy tính Tổ hợp chập  $k$  của  $n$  phần tử được tính bằng công thức sau

$$C_n^k = \begin{cases} 1 & \text{ khi } k = 0 \wedge k = n \\ C_{n-1}^k + C_{n-1}^{k-1} & \text{ khi } 0 < k < n \end{cases}$$


**Bài 10.** Viết hàm tìm kiếm phần tử trên mảng đã sắp xếp theo kỹ thuật tìm kiếm nhị phân bằng phương pháp đệ quy.

**Bài 11.** Tính số cách chia  $m$  phần thưởng cho  $n$  học sinh đã xếp theo thứ tự hạng tăng dần, phân chia sao cho học sinh đứng trước có số phần thưởng lớn hơn học sinh đứng sau, biết rằng:

- $m \geq 2n$
- $m = n$
- $m > 2n$  và mỗi học sinh đều có quà.

--HẾT--

## A. MỤC TIÊU:

<b>Trường ĐH CNTP TP. HCM</b> <b>Khoa Công nghệ thông tin</b> <b>Bộ môn Công nghệ phần mềm</b> <b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b>	<b>BUỔI 9.</b> <b>BÀI TẬP TỔNG HỢP</b>	
---	---	---

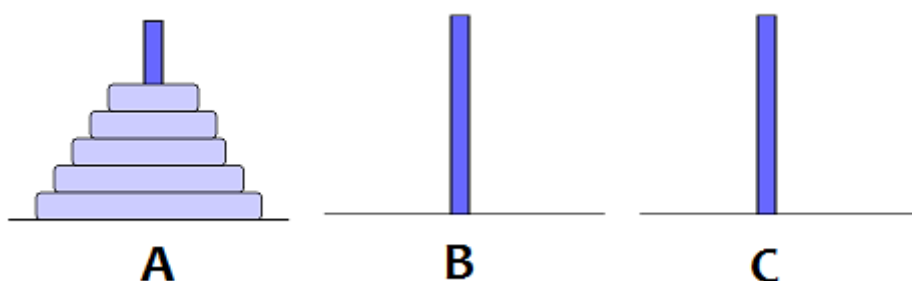
- Vận dụng các kiến thức xử lý ngoại lệ, xử lý mảng, xử lý chuỗi, kỹ thuật đệ quy để giải các bài toán thông dụng.
- Vận dụng các quy tắc để viết “clean code”

## B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

## C. NỘI DUNG THỰC HÀNH

**Bài 1. Bài toán tháp Hà Nội:** Có 3 chồng đĩa được đánh số A, B, C. Khởi đầu chồng đĩa A có n đĩa, được xếp sao cho đĩa lớn hơn luôn nằm dưới và B chồng đĩa còn lại chưa có đĩa nào. Hãy chuyển hết các đĩa từ chồng số 1 sang chồng số C, mỗi lần chỉ chuyển 1 đĩa, được dùng chồng số B làm trung gian, trong quá trình vận chuyển phải đảm bảo đĩa lớn hơn luôn nằm dưới.



Hàm Tower(n, colA, colB, colC)

Nếu  $n = 1$  thì

Chuyển thẳng 1 đĩa (duy nhất) từ colA sang colC

Ngược lại

Tower( $n - 1$ , colA, colC, colB)

// Chuyển  $n - 1$  đĩa colA sang colB (colC làm trung gian)

MoveDisk(colA, colC) // Chuyển đĩa  $n$  từ colA sang colC

Tower( $n - 1$ , colB, colA, colC)

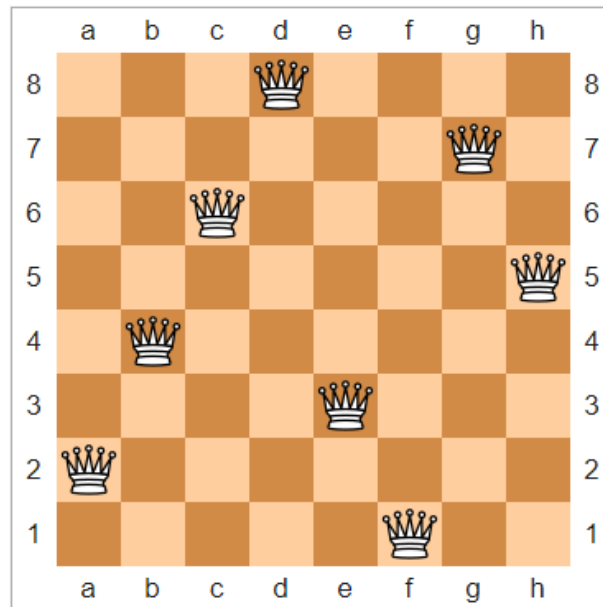
// Chuyển  $n - 1$  đĩa từ colB sang colC (colA trung gian)

Viết chương trình hiển thị thứ tự chuyển các đĩa theo phương pháp đệ quy.

**Bài 2.** Bài toán 8 quân hậu: đặt 8 quân hậu lên bàn cờ vua sao cho chúng không khống chế lẫn nhau. Viết chương trình tìm tất cả các lời giải.

**Gợi ý:**

Xếp tám quân hậu trên bàn cờ sao cho không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo. Bài toán tám quân hậu có thể tổng quát hóa thành bài toán đặt  $n$  quân hậu trên bàn cờ  $n \times n$  (với  $n \geq 4$ ).



Một trong 12 lời giải

***Giải bài toán xếp hậu bằng đệ quy trong C/C++***

- Để tiện trình bày ta dùng biến  $i$  để đánh dấu các hàng từ trên xuống ( 1 đến  $n$ ). Dùng biến  $j$  để đánh dấu các cột từ trái sang phải ( 1 đến  $n$ );
- Các phần tử nằm trên cùng hàng có chỉ số hàng bằng nhau;
- Các phần tử nằm trên cùng cột có chỉ số cột bằng nhau;
- Để tiện cho việc in kết quả ra thì ta chỉ in ra chỉ số các cột tuần tự theo các hàng từ trên xuống.
- Điều kiện để đặt một quân hậu đúng chỗ là không có 2 trên cùng một cột ( chỉ số cột khác nhau). Không có 2 quân hậu nào cùng ở trên một đường chéo.

***Ý tưởng:***

- Đầu tiên ta đặt quân hậu thứ nhất vào các cột trên hàng 1 ( có  $n$  cách đặt ).
- Thử đặt quân hậu 2 vào từng cột ở hàng 2 sao cho không bị quân hậu 1 khống chế. Với mỗi vị trí của quân hậu này ta lại thử đặt quân hậu thứ ba vào các cột sao cho không bị các quân hậu trước khống chế.
- Sau khi đặt xong quân hậu thứ tám thì in ra một cách đặt.

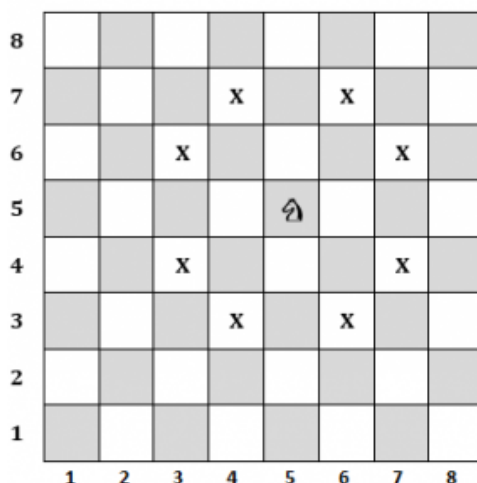
**Bài 3.** Bài toán mã đi tuần. Trên bàn cờ quốc tế  $8 \times 8$  (64 ô), hãy viết hàm đệ quy sao cho con mã đi qua 64 nước sao cho mỗi ô chỉ đi qua 1 lần.

### Gợi ý:

- Mã đi tuần (hay hành trình của quân mã) là bài toán về việc di chuyển một quân mã trên bàn cờ vua (8 x 8). Quân mã được đặt ở một ô trên một bàn cờ trống nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.
- Nếu một quân mã đi hết 64 vị trí và tại vị trí cuối cùng có thể di chuyển đến vị trí bắt đầu thông qua một nước cờ thì đó gọi là một hành trình đóng
- Có những hành trình, trong đó quân mã sau khi đi hết tất cả 64 ô của bàn cờ và từ ô cuối của hành trình không thể đi về ô xuất phát chỉ bằng một nước đi. Những hành trình như vậy được gọi là hành trình mở.
- Cách di chuyển của một quân mã

Nước đi của một quân mã giống hình chữ L và nó có thể di chuyển tất cả các hướng. Ở một vị trí thích hợp thì quân mã có thể di chuyển đến được 8 vị trí.

- Tại 1 ô bất kỳ trên bàn cờ, con mã có thể đi tối đa 8 nước quanh vị trí hiện tại.



- Tại vị trí hiện tại, ta chọn thử 1 bước đi kế tiếp
- Nếu không thành công thì lần ngược để chọn bước khác
- Nếu thành công thì ghi nhận bước đi này.

### Hướng dẫn giải bài toán mã đi tuần:

#### ❖ Xây dựng bước đi cho quân mã

Gọi  $x, y$  là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ của quân mã sẽ như sau:

$$|x| + |y| = 3 \text{ (với } x, y > 0 \text{)}.$$

Khi đó ở một vị trí bất kỳ quân mã có 8 đường có thể di chuyển, chưa xét đến bước đi đó có hợp lệ hay không.

Các bước đi đó là:  $(-2, -1), (-2, 1), (-1, -2), (-1, 2), (1, -2), (1, 2), (2, -1), (2, 1)$

Để đơn giản ta sẽ tạo hai mảng  $X[]$ ,  $Y[]$  để chứa các giá trị trên, với mỗi  $X[i]$ ,  $Y[i]$  sẽ là một cách di chuyển của quân mã ( $0 \leq i \leq 7$ ).

#### ❖ Kiểm tra tính hợp lệ của bước đi

Ta sẽ dùng một mảng hai chiều  $A[n*n]$  để lưu vị trí của từng ô trong bàn cờ. Tất cả mảng đều khởi tạo giá trị là 0 (quân mã chưa đi qua).

Gọi  $x$ ,  $y$  là vị trí hiện tại của quân mã, thì vị trí tiếp theo mà quân mã đi sẽ có dạng  $x + X[i]$ ,  $y + Y[i]$ . Một vị trí được gọi là hợp lệ thì sẽ thỏa mãn tính chất sau:

- $0 \leq x + X[i] \leq n-1$ .
- $0 \leq y + Y[i] \leq n-1$ .

Nếu bước đi đó là bước đi đúng thì ta sẽ lưu thứ tự của bước đi đó vào mảng  $A[x + X[i], y + Y[i]]$ .

--HẾT--

<p><b>Trường ĐH CNTP TP. HCM</b></p> <p><b>Khoa Công nghệ thông tin</b></p> <p><b>Bộ môn Công nghệ phần mềm</b></p> <p><b>THỰC HÀNH KỸ THUẬT LẬP TRÌNH</b></p>	<p><b>BUỔI 10.</b></p> <p><b>ÔN TẬP - KIỂM TRA LẦN 2</b></p>	
--	--	--

#### A. MỤC TIÊU:

- Phân tích các yêu cầu của bài toán.
- Cài đặt được các hàm xử lý mảng hai cho các bài toán.
- Áp dụng các kỹ thuật cài đặt xử lý chuỗi và đệ quy.

#### B. DỤNG CỤ - THIẾT BỊ THỰC HÀNH CHO MỘT SV:

STT	Chủng loại – Quy cách vật tư	Số lượng	Đơn vị	Ghi chú
1	Computer	1	1	

#### C. NỘI DUNG THỰC HÀNH

##### **ÔN TẬP - ĐỀ KIỂM TRA MẪU:**

<p align="center"><b>KIỂM TRA LẦN 2</b></p> <p align="center"><b>MÔN: TH KỸ THUẬT LẬP TRÌNH – THỜI GIAN: 120 phút</b></p> <p align="center"><b>(SV không được sử dụng tài liệu)</b></p>	
<p><b>Câu 1.</b> Tạo chương trình theo cấu trúc (khai báo thư viện, khai báo cấu trúc, khai báo hàm con, hàm main, thân hàm con), có hàm hiển thị danh sách bài thực hiện được và cho người dùng lựa chọn từng bài cần thực hiện. Trong chương trình cần xử lý ngoại lệ nếu có</p>	<b>(1 điểm)</b>
<p><b>Câu 2.</b> Cho mảng 1 chiều a chứa số nguyên. Viết các hàm xử lý sau theo kỹ thuật đệ quy:</p>	
2.1. Tính tích các số chẵn trong a	<b>(1 điểm)</b>
2.2. Xuất các phần tử có chứa chữ số 4 trong a.	<b>(1 điểm)</b>
<p><b>Câu 3.</b> Viết các hàm sau bằng kỹ thuật đệ quy</p>	
3.1. Tính $S(n) = \frac{1}{1.2^1} + \frac{2}{2.3^2} + \frac{3}{3.4^3} + \dots + \frac{n}{n.(n+1)^n}$	<b>(1 điểm)</b>
Viết hàm theo 2 cách đệ quy và khử đệ quy.	
3.2. Tính tổng các chữ số chẵn của số nguyên dương N.	<b>(1 điểm)</b>
Viết hàm theo 2 cách đệ quy và khử đệ quy.	
3.2. Xuất các số Fibonacci lẻ thuộc đoạn [m, n], biết rằng công thức tính số Fibonacci như sau:	<b>(1 điểm)</b>



$$Fibonacci(n) = \begin{cases} 1 & \text{với } n \leq 2 \\ Fibonacci(n-1) + Fibonacci(n-2) & \text{với } n > 2 \end{cases}$$

Ví dụ: Các số chẵn Fibonacci thuộc đoạn [10, 30] gồm: 13, 21

**Câu 4. Nhà thuốc tây quản lý thông tin thuốc gồm:**

- Mã thuốc (5 ký tự)
- Tên thuốc (20 ký tự)
- Nhà sản xuất (20 ký tự)
- Dạng thuốc (10 ký tự) có giá trị là: viên, nước.
- Đơn giá (số thực)
- Công dụng (50 ký tự)

4.1. Tạo và xuất mảng 1 chiều chứa danh sách thuốc. **(2 điểm)**

4.2. Sắp xếp danh sách thuốc tăng dần theo mã thuốc. **(1 điểm)**

4.3. Tìm thuốc có mã số bắt đầu bằng 3 ký tự “T01” theo giải thuật Binary search. **(1 điểm).**

**--HẾT--**

# PHỤ LỤC

## BÀI TẬP THỰC HÀNH NÂNG CAO

### Problem 1. Double-ended Strings

<https://codeforces.com/problemset/problem/1506/C>

You are given the strings  $aa$  and  $bb$ , consisting of lowercase Latin letters. You can do any number of the following operations in any order:

- if  $|a| > 0$  (the length of the string  $a$  is greater than zero), delete the first character of the string  $a$ , that is, replace  $a$  with  $a_2a_3\dots a_n$ ;
- if  $|a| > 0$ , delete the last character of the string  $aa$ , that is, replace  $a$  with  $a_1a_2\dots a_{n-1}$ ;
- if  $|b| > 0$  (the length of the string  $b$  is greater than zero), delete the first character of the string  $b$ , that is, replace  $b$  with  $b_2b_3\dots b_n$ ;
- if  $|b| > 0$ , delete the last character of the string  $b$ , that is, replace  $bb$  with  $b_1b_2\dots b_{n-1}$ .

Note that after each of the operations, the string  $aa$  or  $b$  may become empty.

For example, if  $a = \text{"hello"}$  and  $b = \text{"icpc"}$ , then you can apply the following sequence of operations:

- delete the first character of the string  $a \Rightarrow a = \text{"ello"}$  and  $b = \text{"icpc"}$ ;
- delete the first character of the string  $b \Rightarrow a = \text{"ello"}$  and  $b = \text{"cpc"}$ ;
- delete the first character of the string  $b \Rightarrow a = \text{"ello"}$  and  $b = \text{"pc"}$ ;
- delete the last character of the string  $a \Rightarrow a = \text{"ell"}$  and  $b = \text{"pc"}$ ;
- delete the last character of the string  $b \Rightarrow a = \text{"ell"}$  and  $b = \text{"p"}$ .

For the given strings  $aa$  and  $bb$ , find the minimum number of operations for which you can make the strings  $aa$  and  $bb$  equal. Note that empty strings are also equal.

### Input

The first line contains a single integer  $tt$  ( $1 \leq t \leq 100$ ). Then  $tt$  test cases follow.

The first line of each test case contains the string  $aa$  ( $1 \leq |a| \leq 20$ ), consisting of lowercase Latin letters.

The second line of each test case contains the string  $bb$  ( $1 \leq |b| \leq 20$ ), consisting of lowercase Latin letters.

### Output

For each test case, output the minimum number of operations that can make the strings  $a$  and  $b$  equal.

#### Example

input	Copy
5 a a abcd bc hello codeforces hello helo dhjakjsnasjhfkasafasd adjnsasjhfksvdafdser	
output	Copy
0 2 13 3 20	

## Problem 2. GCD Sum

<https://codeforces.com/problemset/problem/1498/A>

The  $gcdSum$  of a positive integer is the  $gcd$  of that integer with its sum of digits. Formally,  $gcdSum(x) = gcd(x, \text{sum of digits of } x)$  for a positive integer  $x$ .  $gcd(a, b)$  denotes the greatest common divisor of  $a$  and  $b$  — the largest integer  $d$  such that both integers  $a$  and  $b$  are divisible by  $d$ .

For example:  $gcdSum(762) = gcd(762, 7 + 6 + 2) = gcd(762, 15) = 3$ .

Given an integer  $n$ , find the smallest integer  $x \geq n$  such that  $gcdSum(x) > 1$ .

#### Input

The first line of input contains one integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

Then  $t$  lines follow, each containing a single integer  $n$  ( $1 \leq n \leq 10^{18}$ ).

All test cases in one test are different.

#### Output

Output  $t$  lines, where the  $i$ -th line is a single integer containing the answer to the  $i$ -th test case.

#### Example

input	Copy
3 11 31 75	
output	Copy
12 33 75	

## Note

Let us explain the three test cases in the sample.

**Test case 1:**  $n = 11$ :

$$\text{gcdSum}(11) = \text{gcd}(11, 1 + 1) = \text{gcd}(11, 2) = 1.$$

$$\text{gcdSum}(12) = \text{gcd}(12, 1 + 2) = \text{gcd}(12, 3) = 3.$$

So the smallest number  $\geq 11$  whose  $\text{gcdSum} > 1$  is 12.

**Test case 2:**  $n = 31$ :

$$\text{gcdSum}(31) = \text{gcd}(31, 3 + 1) = \text{gcd}(31, 4) = 1.$$

$$\text{gcdSum}(32) = \text{gcd}(32, 3 + 2) = \text{gcd}(32, 5) = 1.$$

$$\text{gcdSum}(33) = \text{gcd}(33, 3 + 3) = \text{gcd}(33, 6) = 3.$$

So the smallest number  $\geq 31$  whose  $\text{gcdSum} > 1$  is 33.

**Test case 3:**  $n = 75$ :

$$\text{gcdSum}(75) = \text{gcd}(75, 7 + 5) = \text{gcd}(75, 12) = 3.$$

The  $\text{gcdSum}$  of 75 is already  $> 1$ . Hence, it is the answer.

## Problem 3. k-LCM (easy version)

<https://codeforces.com/problemset/problem/1497/C1>

It is the easy version of the problem. The only difference is that in this version  $k = 3$ .

You are given a positive integer  $n$ . Find  $k$  positive integers  $a_1, a_2, \dots, a_k$ , such that:

- $a_1 + a_2 + \dots + a_k = n$
- $\text{LCM}(a_1, a_2, \dots, a_k) \leq \frac{n}{2}$

Here  $\text{LCM}$  is the **least common multiple** of numbers  $a_1, a_2, \dots, a_k$ .

We can show that for given constraints the answer always exists.

## Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

## Output

For each test case print  $k$  positive integers  $a_1, a_2, \dots, a_k$ , for which all conditions are satisfied.

## Example

input	Copy
3 3 3 8 3 14 3	
output	Copy
1 1 1 4 2 2 2 6 6	

## Problem 4. Max and Mex

<https://codeforces.com/problemset/problem/1496/B>

You are given a multiset  $S$  initially consisting of  $n$  distinct non-negative integers. A multiset is a set, that can contain some elements multiple times.

You will perform the following operation  $k$  times:

- Add the element  $\lceil \frac{a+b}{2} \rceil$  (rounded up) into  $S$ , where  $a = \text{mex}(S)$  and  $b = \text{max}(S)$ . If this number is already in the set, it is added again.

Here  $\text{max}$  of a multiset denotes the maximum integer in the multiset, and  $\text{mex}$  of a multiset denotes the smallest non-negative integer that is not present in the multiset. For example:

- $\text{mex}(\{1, 4, 0, 2\}) = 3$ ;
- $\text{mex}(\{2, 5, 1\}) = 0$ .

Your task is to calculate the number of **distinct** elements in  $S$  after  $k$  operations will be done.

## Input

The input consists of multiple test cases. The first line contains a single integer  $t$  ( $1 \leq t \leq 100$ ) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers  $n, k$  ( $1 \leq n \leq 10^5, 0 \leq k \leq 10^9$ ) — the initial size of the multiset  $S$  and how many operations you need to perform.

The second line of each test case contains  $n$  **distinct** integers  $a_1, a_2, \dots, a_n$  ( $0 \leq a_i \leq 10^9$ ) — the numbers in the initial multiset.

It is guaranteed that the sum of  $n$  over all test cases does not exceed  $10^5$ .

## Output

For each test case, print the number of **distinct** elements in  $S$  after  $k$  operations will be done.

## Example

input	Copy
5 4 1 0 1 3 4 3 1 0 1 4 3 0 0 1 4 3 2 0 1 2 3 2 1 2 3	

output	Copy
4 4 3 5 3	

#### Note

In the first test case,  $S = \{0, 1, 3, 4\}$ ,  $a = \text{mex}(S) = 2$ ,  $b = \text{max}(S) = 4$ ,  $\lceil \frac{a+b}{2} \rceil = 3$ . So 3 is added into  $S$ , and  $S$  becomes  $\{0, 1, 3, 3, 4\}$ . The answer is 4.

In the second test case,  $S = \{0, 1, 4\}$ ,  $a = \text{mex}(S) = 2$ ,  $b = \text{max}(S) = 4$ ,  $\lceil \frac{a+b}{2} \rceil = 3$ . So 3 is added into  $S$ , and  $S$  becomes  $\{0, 1, 3, 4\}$ . The answer is 4.

## Problem 5. Three swimmers

<https://codeforces.com/problemset/problem/1492/A>

Three swimmers decided to organize a party in the swimming pool! At noon, they started to swim from the left side of the pool.

It takes the first swimmer exactly  $a$  minutes to swim across the entire pool and come back, exactly  $b$  minutes for the second swimmer and  $c$  minutes for the third. Hence, the first swimmer will be on the left side of the pool after  $0, a, 2a, 3a, \dots$  minutes after the start time, the second one will be at  $0, b, 2b, 3b, \dots$  minutes, and the third one will be on the left side of the pool after  $0, c, 2c, 3c, \dots$  minutes.

You came to the left side of the pool exactly  $p$  minutes after they started swimming. Determine how long you have to wait before one of the swimmers arrives at the left side of the pool.

#### Input

The first line of the input contains a single integer  $t$  ( $1 \leq t \leq 1000$ ) — the number of test cases. Next  $t$  lines contains test case descriptions, one per line.

Each line contains four integers  $p, a, b$  and  $c$  ( $1 \leq p, a, b, c \leq 10^{18}$ ), time in minutes after the start, when you came to the pool and times in minutes it take the swimmers to cross the entire pool and come back.

#### Output

For each test case, output one integer — how long you have to wait (in minutes) before one of the swimmers arrives at the left side of the pool.

#### Example

input	Copy
4 9 5 4 8 2 6 10 9 10 2 5 10 10 9 9 9	
output	Copy
1 4 0 8	

### Note

In the first test case, the first swimmer is on the left side in 0, 5, 10, 15, . . . minutes after the start time, the second swimmer is on the left side in 0, 4, 8, 12, . . . minutes after the start time, and the third swimmer is on the left side in 0, 8, 16, 24, . . . minutes after the start time. You arrived at the pool in 9 minutes after the start time and in a minute you will meet the first swimmer on the left side.

In the second test case, the first swimmer is on the left side in 0, 6, 12, 18, . . . minutes after the start time, the second swimmer is on the left side in 0, 10, 20, 30, . . . minutes after the start time, and the third swimmer is on the left side in 0, 9, 18, 27, . . . minutes after the start time. You arrived at the pool 2 minutes after the start time and after 4 minutes meet the first swimmer on the left side.

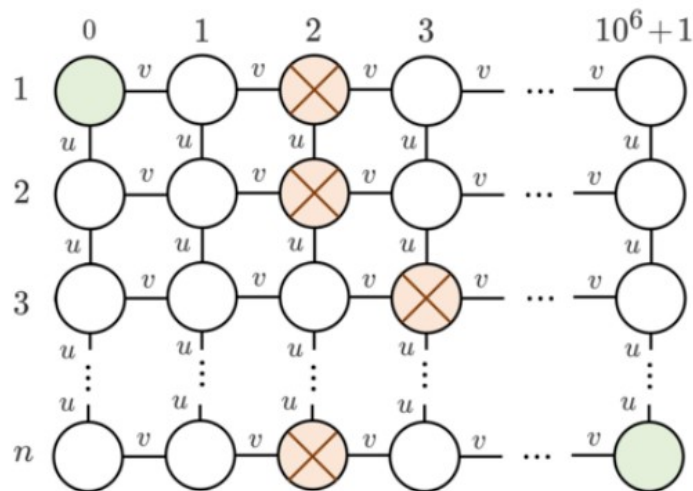
In the third test case, you came to the pool 10 minutes after the start time. At the same time, all three swimmers are on the left side. A rare stroke of luck!

In the fourth test case, all swimmers are located on the left side in 0, 9, 18, 27, . . . minutes after the start time. You arrived at the pool 10 minutes after the start time and after 8 minutes meet all three swimmers on the left side.

## Problem 6. Minimal Cost

<https://codeforces.com/problemset/problem/1491/B>

There is a graph of  $n$  rows and  $10^6 + 2$  columns, where rows are numbered from 1 to  $n$  and columns from 0 to  $10^6 + 1$ :



Let's denote the node in the row  $i$  and column  $j$  by  $(i, j)$ .

Initially for each  $i$  the  $i$ -th row has exactly one obstacle — at node  $(i, a_i)$ . You want to move some obstacles so that you can reach node  $(n, 10^6 + 1)$  from node  $(1, 0)$  by moving through edges of this graph (you can't pass through obstacles). Moving one obstacle to an adjacent by edge free node costs  $u$  or  $v$  coins, as below:

- If there is an obstacle in the node  $(i, j)$ , you can use  $u$  coins to move it to  $(i - 1, j)$  or  $(i + 1, j)$ , if such node exists and if there is no obstacle in that node currently.
- If there is an obstacle in the node  $(i, j)$ , you can use  $v$  coins to move it to  $(i, j - 1)$  or  $(i, j + 1)$ , if such node exists and if there is no obstacle in that node currently.
- Note that you **can't move obstacles outside the grid**. For example, you can't move an obstacle from  $(1, 1)$  to  $(0, 1)$ .

Refer to the picture above for a better understanding.

Now you need to calculate the minimal number of coins you need to spend to be able to reach node  $(n, 10^6 + 1)$  from node  $(1, 0)$  by moving through edges of this graph without passing through obstacles.

### Input

The first line contains a single integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

The first line of each test case contains three integers  $n$ ,  $u$  and  $v$  ( $2 \leq n \leq 100$ ,  $1 \leq u, v \leq 10^9$ ) — the number of rows in the graph and the numbers of coins needed to move vertically and horizontally respectively.

The second line of each test case contains  $n$  integers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^6$ ) — where  $a_i$  represents that the obstacle in the  $i$ -th row is in node  $(i, a_i)$ .

It's guaranteed that the sum of  $n$  over all test cases doesn't exceed  $2 \cdot 10^4$ .

### Output

For each test case, output a single integer — the minimal number of coins you need to spend to be able to reach node  $(n, 10^6 + 1)$  from node  $(1, 0)$  by moving through edges of this graph without passing through obstacles.

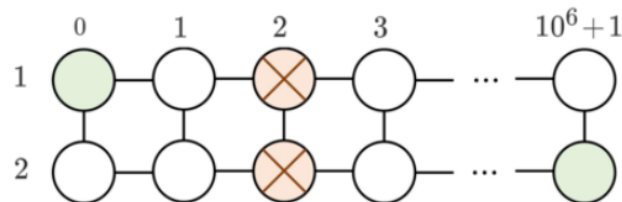
It can be shown that under the constraints of the problem there is always a way to make such a trip possible.

### Example

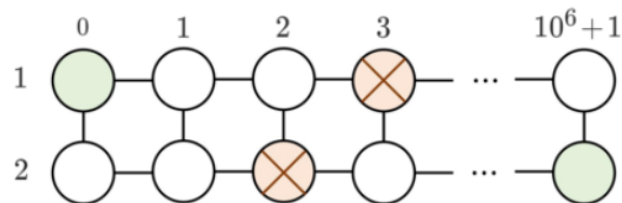
input	Copy
3 2 3 4 2 2 2 3 4 3 2 2 4 3 3 2	
output	Copy
7 3 3	

### Note

In the first sample, two obstacles are at  $(1, 2)$  and  $(2, 2)$ . You can move the obstacle on  $(2, 2)$  to  $(2, 3)$ , then to  $(1, 3)$ . The total cost is  $u + v = 7$  coins.



In the second sample, two obstacles are at  $(1, 3)$  and  $(2, 2)$ . You can move the obstacle on  $(1, 3)$  to  $(2, 3)$ . The cost is  $u = 3$  coins.



## Problem 7. Pythagorean Triples

<https://codeforces.com/problemset/problem/1487/D>

A Pythagorean triple is a triple of integer numbers  $(a, b, c)$  such that it is possible to form a right triangle with the lengths of the first cathetus, the second cathetus and the hypotenuse equal to  $a$ ,  $b$  and  $c$ , respectively. An example of the Pythagorean triple is  $(3, 4, 5)$ .

Vasya studies the properties of right triangles, and he uses a formula that determines if some triple of integers is Pythagorean. Unfortunately, he has forgotten the exact formula; he remembers only that the formula was some equation with squares. So, he came up with the following formula:  $c = a^2 - b$ .

Obviously, this is not the right formula to check if a triple of numbers is Pythagorean. But, to Vasya's surprise, it actually worked on the triple  $(3, 4, 5)$ :  $5 = 3^2 - 4$ , so, according to Vasya's formula, it is a Pythagorean triple.

When Vasya found the right formula (and understood that his formula is wrong), he wondered: how many are there triples of integers  $(a, b, c)$  with  $1 \leq a \leq b \leq c \leq n$  such that they are Pythagorean both according to his formula and the real definition? He asked you to count these triples.

### Input

The first line contains one integer  $t$  ( $1 \leq t \leq 10^4$ ) — the number of test cases.

Each test case consists of one line containing one integer  $n$  ( $1 \leq n \leq 10^9$ ).

### Output

For each test case, print one integer — the number of triples of integers  $(a, b, c)$  with  $1 \leq a \leq b \leq c \leq n$  such that they are Pythagorean according both to the real definition and to the formula Vasya came up with.



### Example

input	Copy
3 3 6 9	
output	Copy
0 1 1	

### Note

The only Pythagorean triple satisfying  $c = a^2 - b$  with  $1 \leq a \leq b \leq c \leq 9$  is  $(3, 4, 5)$ ; that's why the answer for  $n = 3$  is 0, and the answer for  $n = 6$  (and for  $n = 9$ ) is 1.

## Problem 8. Guessing the Greatest (easy version)

<https://codeforces.com/problemset/problem/1486/C1>

The only difference between the easy and the hard version is the limit to the number of queries.

This is an interactive problem.

There is an array  $a$  of  $n$  different numbers. In one query you can ask the position of the second maximum element in a subsegment  $a[l..r]$ . Find the position of the maximum element in the array in no more than 40 queries.

A subsegment  $a[l..r]$  is all the elements  $a_l, a_{l+1}, \dots, a_r$ . After asking this subsegment you will be given the position of the second maximum from this subsegment in the whole array.

### Input

The first line contains a single integer  $n$  ( $2 \leq n \leq 10^5$ ) — the number of elements in the array.

### Interaction

You can ask queries by printing "? l r" ( $1 \leq l < r \leq n$ ). The answer is the index of the second maximum of all elements  $a_l, a_{l+1}, \dots, a_r$ . Array  $a$  is fixed beforehand and can't be changed in time of interaction.

You can output the answer by printing "! p", where  $p$  is the index of the maximum element in the array.

You can ask no more than 40 queries. Printing the answer doesn't count as a query.

After printing a query do not forget to output end of line and flush the output. Otherwise, you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages

### Hacks

To make a hack, use the following test format.

In the first line output a single integer  $n$  ( $2 \leq n \leq 10^5$ ). In the second line output a permutation of  $n$  integers 1 to  $n$ . The position of  $n$  in the permutation is the position of the maximum

### Example

input	Copy
5 3 4	
output	Copy
? 1 5 ? ? ! 1	

### Note

In the sample suppose  $a$  is  $[5, 1, 4, 2, 3]$ . So after asking the  $[1..5]$  subsegment 4 is second to max value, and it's position is 3. After asking the  $[4..5]$  subsegment 2 is second to max value and it's position in the whole array is 4.

Note that there are other arrays  $a$  that would produce the same interaction, and the answer for them might be different. Example output is given in purpose of understanding the interaction.

## Problem 9. The Robot

<https://codeforces.com/problemset/problem/1468/K>

There is a robot on a checkered field that is endless in all directions. Initially, the robot is located in the cell with coordinates  $(0, 0)$ . He will execute commands which are described by a string of capital Latin letters 'L', 'R', 'D', 'U'. When a command is executed, the robot simply moves in the corresponding direction:

- 'L': one cell to the left (the  $x$ -coordinate of the current cell decreases by 1);
- 'R': one cell to the right (the  $x$ -coordinate of the current cell is increased by 1);
- 'D': one cell down (the  $y$ -coordinate of the current cell decreases by 1);
- 'U': one cell up (the  $y$ -coordinate of the current cell is increased by 1).

Your task is to put an obstacle in one cell of the field so that after executing the commands, the robot will return to the original cell of its path  $(0, 0)$ . Of course, an obstacle cannot be placed in the starting cell  $(0, 0)$ . It is guaranteed that if the obstacle is not placed, then the robot will not return to the starting cell.

An obstacle affects the movement of the robot in the following way: if it tries to go in a certain direction, and there is an obstacle, then it simply remains in place (the obstacle also remains, that is, it does not disappear).

Find any such cell of the field (other than  $(0, 0)$ ) that if you put an obstacle there, the robot will return to the cell  $(0, 0)$  after the execution of all commands. If there is no solution, then report it.

### Input

The first line contains one integer  $t$  ( $1 \leq t \leq 500$ ) — the number of test cases.

Each test case consists of a single line containing  $s$  — the sequence of commands, which are uppercase Latin letters 'L', 'R', 'D', 'U' only. The length of  $s$  is between 1 and 5000, inclusive. Additional constraint on  $s$ : executing this sequence of commands leads the robot to some cell other than  $(0, 0)$ , if there are no obstacles.

The sum of lengths of all  $s$  in a test doesn't exceed 5000.

### Output

For each test case print a single line:

- if there is a solution, print two integers  $x$  and  $y$  ( $-10^9 \leq x, y \leq 10^9$ ) such that an obstacle in  $(x, y)$  will force the robot to return back to the cell  $(0, 0)$ ;
- otherwise, print two zeroes (i. e. 0 0).

### Example

input	Copy
4 L RUUDL LLUU DDUUUUU	
output	Copy
-1 0 1 2 0 0 0 1	

## Problem 10. Prison Break

<https://codeforces.com/problemset/problem/1415/A>

There is a prison that can be represented as a rectangular matrix with  $n$  rows and  $m$  columns. Therefore, there are  $n \cdot m$  prison cells. There are also  $n \cdot m$  prisoners, one in each prison cell. Let's denote the cell in the  $i$ -th row and the  $j$ -th column as  $(i, j)$ .

There's a secret tunnel in the cell  $(r, c)$ , that the prisoners will use to escape! However, to avoid the risk of getting caught, they will escape at night.

Before the night, every prisoner is in his own cell. When night comes, they can start moving to adjacent cells. Formally, in one second, a prisoner located in cell  $(i, j)$  can move to cells  $(i - 1, j)$ ,  $(i + 1, j)$ ,  $(i, j - 1)$ , or  $(i, j + 1)$ , as long as the target cell is inside the prison. They can also choose to stay in cell  $(i, j)$ .

The prisoners want to know the minimum number of seconds needed so that every prisoner can arrive to cell  $(r, c)$  if they move optimally. Note that there can be any number of prisoners in the same cell at the same time.

### Input

The first line contains an integer  $t$  ( $1 \leq t \leq 10^4$ ), the number of test cases.

Each of the next  $t$  lines contains four space-separated integers  $n, m, r, c$  ( $1 \leq r \leq n \leq 10^9, 1 \leq c \leq m \leq 10^9$ ).

### Output

Print  $t$  lines, the answers for each test case.

### Example

<b>input</b>	Copy
<pre>3 10 10 1 1 3 5 2 4 10 2 5 1</pre>	
<b>output</b>	Copy
<pre>18 4 6</pre>	

--HẾT--