**Câu 1**: Giải bất phương trình **ax+b>0**, với **a** và **b** là 2 số nguyên nhập vào.

**Input Format**

Hai số nguyên, **a** và **b**, cách nhau 1 dấu cách.

**Constraints**

Không có.

**Output Format**

* Nếu bpt có vô số nghiệm, in ra **VSN**
* Nếu bpt vô nghiệm, in ra **VN**
* Nếu bpt có nghiệm, in ra nghiệm dạng **x>p** hoặc **x<p**

Trong đó, **p** là một giá trị thực với độ chính xác 2 chữ số sau dấu chấm.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| -2\_4 | -2 4 | x<2.00 |

#include <stdio.h>

int main(){

int a, b;

scanf("%d %d",&a, &b);

if(a==0){

if(b<=0) {

printf("VN");

}else {

printf("VSN");

}

}else if(a>0) {

printf("x>%.2f", (float)-b/a);

}else {

printf("x<%.2f", (float)-b/a);

}

return 0;

}

**Câu 3**:Viết chương trình giải phương trình (pt) ax+b=0.

Với **a**, **b** là hai số thực nhập từ bàn phím.

**Input Format**

* Hai số thực cách nhau dấu 1 cách.

**Constraints**

* Không có.

**Output Format**

* Nếu pt vô nghiệm, in ra **VN**.
* Nếu pt có 1 nghiệm, in ra nghiệm ở định dạng số thực với độ chính xác 2 chữ số sau dấu chấm thập phân.
* Nếu pt có vô số nghiệm, in ra **VSN**.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| 1x+2=0 | 1.00 2.00 | -2.00 |

#include <stdio.h>

int main() {

float a, b;

scanf("%f %f", &a, &b);

if (a == 0 && b == 0) {

printf("VSN");

} else if (a == 0) {

printf("VN");

} else {

printf("%.2f", -b/a);

}

return 0;

}

**Câu 4**: Viết chương trình giải phương trình bậc 2 ax2+bx+c =0. Với hệ số a, b, c được nhập từ bàn phím.

Yêu cầu: Xét tất cả các trường hợp.

Input:

a, b, c là các số thực được nhập từ bàn phím

Output:

+ Nếu vô nghiệm ghi VN

+ Nếu vô số nghiệm ghi VSN

+ Nếu có 1 nghiệm ghi giá trị nghiệm với độ chính xác 3 chữ số thập phân

+ Nếu có 2 nghiệm ghi giá trị nghiệm với độ chính xác 3 chữ số thập phân, các nghiệm cách nhau dấu cách. Nghiệm nhỏ hơn đứng trước.

Ví dụ:

Input: 3 5 -8

Output:

-2.670 1.000

Constrains:

a, b, c là các số thực

**For example:**

| **Input** | **Result** |
| --- | --- |
| -20.000 -1.000 4.000 | -0.473 0.423 |

#include <stdio.h>

#include <math.h>

int main() {

double a, b, c, delta, x1, x2;

scanf("%lf %lf %lf", &a, &b, &c);

if (a == 0) {

if (b == 0) {

if (c == 0) {

printf("VSN");

} else {

printf("VN");

}

} else {

printf("%.3lf", -c / b);

}

} else {

delta = b \* b - 4 \* a \* c;

if (delta < 0) {

printf("VN");

} else if (delta == 0) {

printf("%.3lf", -b / (2 \* a));

} else {

x1 = (-b + sqrt(delta)) / (2 \* a);

x2 = (-b - sqrt(delta)) / (2 \* a);

printf("%.3lf %.3lf", fmin(x1, x2), fmax(x1, x2));

}

}

return 0;

}

**Câu 5**:Viết chương trình thực hiện:

* Nhập hai số nguyên a, b cách nhau bởi dấu cách thể hiện tháng và năm
* In ra số ngày của tháng đó.

**Input Format**

* Hai số nguyên dương a, b cách nhau đúng một dấu cách.

**Constraints**

* 1 <= **a, b**

**Output Format**

* Nếu nhập vào a không thuộc đoạn [1, 12] hoặc b là số <=0 thì báo "Du lieu sai."
* Ngược lại in ra thông báo "Thang a nam b co X ngay."
* X là số ngày của tháng a trong năm b.
  + Lưu ý tháng 2 năm nhuận có 29 ngày
  + Năm nhuận là năm thỏa mãn a) hoặc b)

a) Nó lặp lại 4 năm 1 lần, nhưng không lặp lại mỗi 100 năm. (chia hết cho 4 nhưng không chia hết cho 100)

b) Năm nhuận cũng lặp lại mỗi 400 năm. (chia hết cho 400)

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 2022 | Thang 2 nam 2022 co 28 ngay. |

#include <stdio.h>

int main()

{

int a, b;

scanf("%d %d", &a, &b);

if(b>0){

switch(a){

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12: printf("Thang %d nam %d co 31 ngay.", a, b); break;

case 4:

case 6:

case 9:

case 11: printf("Thang %d nam %d co 30 ngay.", a, b); break;

case 2:

if((b%4==0 && b%100 != 0) || (b%400==0)) printf("Thang 2 nam %d co 29 ngay.", b);

else printf("Thang 2 nam %d co 28 ngay.", b); break;

default: printf("Du lieu sai.");

}

}else printf("Du lieu sai.");

}

**Câu 7**: Giả sử lịch học của một bạn sinh viên như sau: thứ 2, 3 học lý thuyết tại giảng đường, thứ 4, 5, 6 học thực hành tại phòng máy tính, riêng tối thứ 4 học bù môn Lập trình C, thứ 7 học ngoại ngữ tại trung tâm ngoại ngữ, chủ nhật được nghỉ.

Bạn hãy viết chương trình thực hiện công việc sau:

Nhập một số nguyên chỉ thứ trong tuần, số 2 ứng với thứ 2, số 3 ứng với thứ 3, ..., số 8 ứng với chủ nhật. In ra lịch học tương ứng của bạn sv trên.

Nếu người dùng nhập các số không đúng quy định thì in ra thông báo "Du lieu sai."

**Input Format**

* Một số nguyên chỉ thứ trong tuần.

**Constraints**

* Số nhập vào là số nguyên

**Output Format**

* In ra thông báo lịch học "Hoc ly thuyet.", "Hoc thuc hanh.", "Hoc ngoai ngu.", "Toi nay hoc bu Lap trinh C.", "Nghi.", "Du lieu sai." như yêu cầu của đề bài.
* Nếu có nhiều thông báo thì mỗi thông báo trên một dòng, thông báo nào đến trước thì ở dòng trên.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| -1\_Error | -1 | Du lieu sai. |
| 2\_Kem | 2 | Hoc ly thuyet. |
| 5\_TB | 5 | Hoc thuc hanh. |

#include <stdio.h>

int main(){

int a;

scanf("%d", &a);

if(a==2 || a==3){

printf("Hoc ly thuyet.");

}else if(a==5 || a==6){

printf("Hoc thuc hanh.");

}else if(a==4){

printf("Hoc thuc hanh.\nToi nay hoc bu Lap trinh C.");

}else if(a==7){

printf("Hoc ngoai ngu.");

}else if(a==8){

printf("Nghi.");

}

else printf("Du lieu sai.");

}

**Câu 8**: Nhập vào 3 số thực a, b, c.

Kiểm tra 3 số đó có tạo thành 3 cạnh của một tam giác không.

Nếu đúng thì phân loại xem tam giác đó là tam giác gì trong các loại: Cân, đều, vuông, thường.

Nếu không thì đưa ra thông báo "Khong phai tam giac." //có dấu chấm ở cuối thống bào

**Input Format**

* 3 số thực

**Constraints**

* a, b, c là các số thực

**Output Format**

* In ra một trong các phân loại: **"Tam giac can."; "Tam giac deu."; "Tam giac vuong."; "Tam giac thuong."**
* Hoặc in ra thông báo lỗi: **"Khong phai tam giac."**
* **//Chú ý có dấu chấm ở cuối các thông báo. Riêng trường hợp tam giác đều thì có cả thông báo "Tam giac can." ở dòng trên.**

**Example:**

**- Input: 4 4 4**

**- Output:**

**Tam giac can.**

**Tam giac deu.**

**For example:**

| **Input** | **Result** |
| --- | --- |
| -1 2 3 | Khong phai tam giac. |

#include <stdio.h>

int main()

{

float a, b, c;

scanf("%f%f%f", &a, &b, &c);

if(a<b+c && b<a+c && c<a+b)

{

if(a\*a==b\*b+c\*c || b\*b==a\*a+c\*c || c\*c==b\*b+a\*a)

printf("Tam giac vuong.");

else if(a==b || a==c || b==c)

{

printf("Tam giac can.");

if(a==b && b==c)

printf("\nTam giac deu.");

}

else

{

printf("Tam giac thuong.");

}

}

else

printf("Khong phai tam giac.");

}

**Câu 10**: Nhập một số nguyên chỉ điểm ở thang 10.

In ra xếp loại dựa trên điểm như sau:

* Nếu điểm từ 0~3 in ra **Kem**
* Nếu là 4 in ra **Yeu**
* Nếu từ 5~6 in ra **TB**
* Nếu từ 7~8 in ra **Kha**
* Nếu từ 9~10 in ra **Gioi**
* Nếu điểm không thuộc phạm vi nào bên trên thì in ra **Error**

**Input Format**

* Một số nguyên chỉ điểm (**d**) theo thang 10.

**Constraints**

* 0 <= **d** <= 10

**Output Format**

* In ra một trong các phân loại: **Kem**, **Yeu**, **TB**, **Kha**, **Gioi**
* Hoặc in ra thông báo lỗi: **Error**
* In đúng chữ hoa thường, không in thêm bất kì kí tự nào khác.

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| -1\_Error | -1 | Error |
| 2\_Kem | 2 | Kem |
| 5\_TB | 5 | TB |
| 8\_Kha | 8 | Kha |
| 10\_Gioi | 10 | Gioi |

#include <stdio.h>

int main(){

int a;

scanf("%d", &a);

if(a>=0 && a<=3){

printf("Kem");

}else if(a==4){

printf("Yeu");

}else if(a>=5 && a<=6){

printf("TB");

}else if(a>=7 && a<=8){

printf("Kha");

}else if(a>=9 && a<=10){

printf("Gioi");

}

else printf("Error");

}

**3 điểm thẳng hàng**

Nhập vào 6 điểm (x1, y1), (x2, y2), (x3, y3), (x4, y4), (x5, y5), (x6, y6). Với xi, yi là các số thực. Hãy kiểm tra 3 điểm (x1, y1), (x2, y2), (x3, y3) có thẳng hàng hay không? Sau đó kiểm tra 3 điểm (x4, y4), (x5, y5), (x6, y6) có thẳng hàng hay không?

Input:

Dòng 1: x1 y1 x2 y2 x3 y3 cách nhau dấu cách

Dòng 2: x4 y4 x5 y5 x6 y6 cách nhau dấu cách

Output:

Nếu 3 điểm thẳng hàng thì in ra chữ YES; ngược lại in ra chữ NO. Các kết quả in trên cùng dòng, cách nhau dấu cách.

Constraints:

xi, yi kiểu float

**For example:**

| **Input** | **Result** |
| --- | --- |
| 219.000000 892.000000  -664.000000 244.000000  -781.000000 194.000000  585.000000 298.000000  585.000000 298.000000  585.000000 298.000000 | NO YES |

#include <stdio.h>

int main() {

float x1, y1, x2, y2, x3, y3, x4, y4, x5, y5, x6, y6;

scanf("%f%f%f%f%f%f", &x1, &y1, &x2, &y2, &x3, &y3);

scanf("%f%f%f%f%f%f", &x4, &y4, &x5, &y5, &x6, &y6);

float check\_1 = (y2 - y1) \* (x3 - x2) - (y3 - y2) \* (x2 - x1);

float check\_2 = (y5 - y4) \* (x6 - x5) - (y6 - y5) \* (x5 - x4);

if (check\_1 == 0) {

printf("YES ");

} else {

printf("NO ");

}

if (check\_2 == 0) {

printf("YES\n");

} else {

printf("NO\n");

}

return 0;

}

Câu 1: Viết chương trình nhập và in ra ma trận gồm n hàng, m cột các số nguyên.

Input: Dòng thứ nhất nhập vào n và m là số hàng và số cột của ma trận.

Dòng tiếp theo nhập vào các phần tử của ma trận gồm n hàng, m cột

Output:

Dòng thứ nhất in ra thông báo "Ma tran ban dau"

Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

Constranins: 1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 18  13 14 2 10 21 18 21 19 2 1 4 20 20 10 19 14 7 19  15 19 8 0 21 5 17 5 13 5 18 15 4 9 12 23 20 24  24 5 22 18 7 0 17 12 23 15 19 13 21 13 17 3 8 24  1 8 0 6 9 20 18 16 23 23 12 10 6 5 22 21 6 1  21 24 8 19 20 15 18 17 8 1 6 3 5 7 11 24 19 13  6 2 2 4 22 3 5 14 14 22 16 18 1 12 7 9 20 24 | Ma tran ban dau  6 18  13 14 2 10 21 18 21 19 2 1 4 20 20 10 19 14 7 19  15 19 8 0 21 5 17 5 13 5 18 15 4 9 12 23 20 24  24 5 22 18 7 0 17 12 23 15 19 13 21 13 17 3 8 24  1 8 0 6 9 20 18 16 23 23 12 10 6 5 22 21 6 1  21 24 8 19 20 15 18 17 8 1 6 3 5 7 11 24 19 13  6 2 2 4 22 3 5 14 14 22 16 18 1 12 7 9 20 24 |

#include <stdio.h>

int main(){

int a[100][100];

int n,m;

scanf("%d %d", &n, &m);

for( int i=0;i<n;i++){

for( int j=0;j<m;j++){

scanf("%d", &a[i][j]);

}

}

printf("Ma tran ban dau\n");

printf("%d %d\n", n,m);

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

printf("%d ", a[i][j]);

}

printf("\n");

}

return 0;

}

Question **2**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập và in ra ma trận gồm n hàng, m cột các số thực.

Input: Dòng thứ nhất nhập vào n và m là số hàng và số cột của ma trận.

Dòng tiếp theo nhập vào các phần tử của ma trận gồm n hàng, m cột

Output:

Dòng thứ nhất in ra thông báo "Ma tran ban vua nhap"

Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

Constranins: 1<=n, m<=200, các phần tử trong ma trận là các số thực, có độ chính xác 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 18  13.20 14.40 2.50 10.00 21.00 18.00 21.00 19.00 2.00 1.00 4.00 20.00 20.00 10.00 19.00 14.00 7.00 19.00  15.00 19.00 8.00 0.00 21.00 5.00 17.00 5.00 13.00 5.00 18.00 15.00 4.00 9.00 12.00 23.00 20.00 24.00  24.00 5.00 22.00 18.00 7.00 0.00 17.00 12.00 23.00 15.00 19.00 13.00 21.00 13.00 17.00 3.00 8.00 24.00  1.00 8.00 0.00 6.00 9.00 20.00 18.00 16.00 23.00 23.00 12.00 10.00 6.00 5.00 22.00 21.00 6.00 1.00  21.00 24.00 8.00 19.00 20.00 15.00 18.00 17.00 8.00 1.00 6.00 3.00 5.00 7.00 11.00 24.00 19.00 13.00  6.00 2.00 2.00 4.00 22.00 3.00 5.00 14.00 14.00 22.00 16.00 18.00 1.00 12.00 7.00 9.00 20.00 24.00 | Ma tran ban vua nhap  6 18  13.20 14.40 2.50 10.00 21.00 18.00 21.00 19.00 2.00 1.00 4.00 20.00 20.00 10.00 19.00 14.00 7.00 19.00  15.00 19.00 8.00 0.00 21.00 5.00 17.00 5.00 13.00 5.00 18.00 15.00 4.00 9.00 12.00 23.00 20.00 24.00  24.00 5.00 22.00 18.00 7.00 0.00 17.00 12.00 23.00 15.00 19.00 13.00 21.00 13.00 17.00 3.00 8.00 24.00  1.00 8.00 0.00 6.00 9.00 20.00 18.00 16.00 23.00 23.00 12.00 10.00 6.00 5.00 22.00 21.00 6.00 1.00  21.00 24.00 8.00 19.00 20.00 15.00 18.00 17.00 8.00 1.00 6.00 3.00 5.00 7.00 11.00 24.00 19.00 13.00  6.00 2.00 2.00 4.00 22.00 3.00 5.00 14.00 14.00 22.00 16.00 18.00 1.00 12.00 7.00 9.00 20.00 24.00 |

#include <stdio.h>

int main(){

float a[200][200];

int n,m;

scanf("%d%d", &n, &m);

for( int i=0;i<n;i++){

for( int j=0;j<m;j++){

scanf("%f", &a[i][j]);

}

}

printf("Ma tran ban vua nhap\n");

printf("%d %d\n", n,m);

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

printf("%.2f ", (float) a[i][j]);

}

printf("\n");

}

}

Question **3**

Correct

Marked out of 1.00

Flag question

Question text

Viết hàm nhập vào mảng gồm n số nguyên, hàm in mảng đó ra màn hình.

Viết chương trình nhập và in ra một mảng các số nguyên có sử dụng hàm ở trên.

Input: Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

Dòng thứ nhất in ra thông báo "Mang ban dau"

Dòng thứ 2 in ra số phần tử của mảng (N)

Dòng thứ 3 in ra các phần tử của mảng cách nhau dấu cách.

Constrains: Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 12  20 11 -4 11 9 24 -15 9 12 1 13 0 | Mang ban dau  12  20 11 -4 11 9 24 -15 9 12 1 13 0 |

#include <stdio.h>

int main(){

int a[1000],n;

scanf("%d", &n);

for(int i=0;i<n;i++){

scanf("%d", &a[i]);

}

printf("Mang ban dau\n");

printf("%d \n",n);

for(int i=0;i<n;i++){

printf("%d ", a[i]);

}

printf("\n");

return 0;

}

Question **4**

Correct

Marked out of 1.00

Flag question

Question text

Viết hàm nhập vào mảng gồm n số thực, hàm in mảng đó ra màn hình.

Viết chương trình nhập và in ra một mảng các số thực có sử dụng hàm ở trên.

Input:

Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Output:

Dòng thứ nhất in ra thông báo "Mang ban dau"

Dòng thứ 2 in ra số phần tử của mảng (N)

Dòng thứ 3 in ra các phần tử của mảng cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Constrains:

N kiểu số nguyên

Mảng kiểu số thực

**For example:**

| **Input** | **Result** |
| --- | --- |
| 7  756.000 165.000 263.000 -33.000 548.000 689.000 350.000 | Mang ban dau  7  756.000 165.000 263.000 -33.000 548.000 689.000 350.000 |

#include<stdio.h>

int main()

{

float a[1000];

int i,n;

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%f",&a[i]);

}

printf("Mang ban dau\n");

printf("%d \n",n);

for(i=0;i<n;i++)

{

printf("%.3f ",a[i]);

}

printf("\n");

}

Question **5**

Correct

Marked out of 1.00

Flag question

Question text

Dãy số tăng là dãy số mà trong đó mọi số đứng sau đều lớn hơn số đứng trước.

Viết hàm kiểm tra một mảng số thực gồm n phần tử có tăng hay ko? Hàm nhận 2 tham số là mảng a và số phần tử. Hàm trả về 0 nếu dãy a ko tăng, 1 nếu ngược lại.

Viết chương trình (hàm main) nhập mảng **a** gồm **n** số thực và mảng b gồm m số thực. Với **n, m** là các số nguyên dương nhập từ bàn phím.

Kiểm tra phần tử mảng a, b có phải là một dãy số tăng hay không?

**Input Format**

Dữ liệu vào nằm trên 4 dòng:

* Dòng 1: một số nguyên dương chỉ số phần từ mảng a.
* Dòng 2: các phần tử mảng a, số thực, cách nhau bởi dấu cách.
* Dòng 3: một số nguyên dương chỉ số phần từ mảng b.
* Dòng 4: các phần tử mảng b, số thực, cách nhau bởi dấu cách.

**Constraints**

* Mảng nhập vào có ít nhất 1 phần tử và nhiều nhất là 500 phần tử. (1<=n, m<=500)
* Kiểu các phần tử mảng là số thực

**Output Format**

**Hai dòng: thông báo kết quả của mảng a và b tương ứng**

* Nếu dãy số tăng, in ra **TANG**
* Ngược lại, in ra **KHONG\_TANG**

**For example:**

| **Test** | **Input** | **Result** |
| --- | --- | --- |
| TC0 | 5  2.00 4.50 6.12 8.30 10.99  5  2.00 4.50 6.15 10.95 8.00 | TANG  KHONG\_TANG |
| TC1 | 5  2.00 4.50 6.15 10.95 8.00  5  2.00 4.50 6.12 8.30 10.99 |  |

#include<stdio.h>

int check (float a[], int n){

for(int i=1;i<n;i++){

if(a[i-1]>=a[i]){

return 0;

}

}

return 1;

}

int main(){

float a[500];

int n;

scanf("%d", &n);

for(int i=0;i<n;i++){

scanf("%f", &a[i]);

}

if(check(a,n)){

printf("TANG\n");

}else {

printf("KHONG\_TANG\n");

}

float b[500];

int m;

scanf("%d", &m);

for(int j=0;j<m;j++){

scanf("%f", &b[j]);

}

if(check(b,m)){

printf("TANG\n");

}else {

printf("KHONG\_TANG\n");

}

}

Question **6**

Correct

Marked out of 1.00

Flag question

Question text

Viết hàm kiểm tra một ma trận vuông có đối xứng qua đường chéo chính không.

Viết chương trình nhập vào hai ma trận vuông cấp N1, N2. Kiểm tra xem các ma trận có đối xứng qua đường chéo chính ko. In kết quả ra màn hình.

Input:

- Dòng thứ nhất nhập vào N1 (kích thước của ma trận thứ nhất)

- Các dòng tiếp theo nhập vào các phần tử của ma trận thứ nhất.

- Dòng tiếp theo nhập vào N2 (kích thước của ma trận thứ nhất)

- Các dòng tiếp theo nhập vào các phần tử của ma trận thứ hai.

Output:

- Dòng thứ nhất in ra thông báo "Ma tran 1: Co/Khong doi xung"

- Dòng thứ nhất in ra thông báo "Ma tran 2: Co/Khong doi xung"

Constrains: 1<=N, N2<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6  5 0 5 11 35 61  28 31 12 9 35 5  16 57 5 55 23 34  4 33 27 14 54 61  12 0 34 61 6 23  5 50 50 51 11 34  3  5 0 7  0 11 2  7 2 6 | Ma tran 1: Khong doi xung  Ma |

#include <stdio.h>

int check(int a[][100], int n){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(i!=j){

if(a[i][j]!=a[j][i]){

return 0;

}

}

}

}

return 1;

}

int main(){

int a[100][100];

int n;

scanf("%d", &n);

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

scanf("%d",&a[i][j]);

}

}

if(check(a,n)){

printf("Ma tran 1: Co doi xung\n");

}else printf("Ma tran 1: Khong doi xung\n");

int b[100][100];

int m;

scanf("%d",&m);

for(int j=0;j<m;j++){

for(int i=0;i<m;i++){

scanf("%d", &b[j][i]);

}

}

if(check(b,m)){

printf("Ma tran 2: Co doi xung\n");

}else printf("Ma tran 2: Khong doi xung\n");

return 0;

}

Question **7**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số nguyên. Sắp xếp từng hàng của ma trận tăng dần. In kết quả ra màn hình.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output:

- Dòng thứ nhất in ra n và m cách nhau dấu cách

- Dòng tiếp theo in ra ma trận kết quả

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 6  17 23 9 3 0 9  12 2 21 12 23 7  12 2 2 13 7 6  7 24 22 12 11 20  4 5 7 24 2 11  15 23 20 22 20 24 | 6 6  0 3 9 9 17 23  2 7 12 12 21 23  2 2 6 7 12 13  7 11 12 20 22 24  2 4 5 7 11 24  15 20 20 22 23 24 |

#include<stdio.h>

void sort(int a[] , int n){

for(int i = 0 ; i < n ; i++){

int pos = i;

for(int j = i + 1 ; j < n ; j++){

if(a[j] < a[pos]){

pos = j ;

}

}

int tmp = a[i];

a[i] = a[pos];

a[pos] = tmp;

}

}

void sapxep(int a[][100] , int n , int m){

for(int i = 0 ; i < n ; i++){

sort(a[i] , m);

}

}

int main(){

int a[100][100];

int n , m;

scanf("%d%d" , &n , &m);

for(int i = 0 ; i < n ; i++){

for(int j = 0 ; j < m ; j++){

scanf("%d" , &a[i][j]);

}

}

printf("%d %d\n" , n , m);

sapxep(a , n , m);

for(int i = 0 ; i < n ; i++){

for(int j = 0 ; j < m ; j++){

printf("%d " , a[i][j]);

}

printf("\n");

}

}

Question **8**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tích 2 ma trận X.Y và in kết quả ra màn hình. Nếu không tính được tích 2 ma trận thì in ra thông báo "Du lieu vao sai"

Input:

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

Output:

+ Nếu tính được tích:

- Dòng thứ nhất in ra thông báo "Ma tran tich"

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tích:

- In ra thông báo "Du lieu vao sai"

Constranins: 1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 8  4 0 9 4 8 8 2 4  5 5 1 7 1 1 5 2  8 8  6 1 4 2 3 2 2 1  6 8 5 7 6 1 8 9  2 7 9 5 4 3 1 2  3 3 4 1 1 3 8 7  4 2 7 7 9 3 1 9  8 6 5 0 2 8 6 0  2 4 8 6 5 0 9 0  0 6 1 3 8 9 3 4 | Ma tran tich  154 175 229 137 182 171 135 138  105 113 136 100 108 68 165 118 |

#include<stdio.h>

int main(){

int n1,m1,n2,m2;

scanf("%d%d",&n1,&m1);

int a[n1][m1];

for(int i=0;i<n1;i++){

for(int j=0;j<m1;j++){

scanf("%d", &a[i][j]);

}

}

scanf("%d%d", &n2,&m2);

int b[n2][m2];

for(int i=0;i<n2;i++){

for(int j=0;j<m2;j++){

scanf("%d", &b[i][j]);

}

}

if(m1==n2){

printf("Ma tran tich\n");

int c[n1][m2];

for(int i=0;i<n1;i++){

for(int j=0;j<m2;j++){

int sum=0;

for(int k=0;k<m1;k++){

sum+=(a[i][k]\*b[k][j]);

}

c[i][j]=sum;

}

}

for(int i=0;i<n1;i++){

for(int j=0;j<m2;j++){

printf("%d ", c[i][j]);

}

printf("\n");

}

}else {

printf("Du lieu vao sai");

}

return 0;

}

Question **9**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào 2 ma trận X (có n1 hàng, m1 cột) và Y (có n2 hàng, m2 cột) các số nguyên. Tính tổng 2 ma trận đó và in kết quả ra màn hình. Nếu không tính được tổng 2 ma trận thì in ra thông báo "Du lieu vao sai"

Input:

+ Dòng thứ nhất nhập vào n1 và m1 là số hàng và số cột của ma trận X.

+ Dòng tiếp theo nhập vào các phần tử của ma trận X gồm n1 hàng, m1 cột

+ Dòng tiếp theo nhập vào n2 và m2 là số hàng và số cột của ma trận Y.

+ Dòng tiếp theo nhập vào các phần tử của ma trận Y gồm n2 hàng, m2 cột

Output:

+ Nếu tính được tổng:

- Dòng thứ nhất in ra thông báo "Ma tran tong"

- Dòng thứ hai in ra số hàng và số cột cách nhau dấu cách

- Các dòng tiếp theo in ra các phần tử của ma trận cách nhau dấu cách.

+ Nếu không tính được tổng:

- In ra thông báo "Du lieu vao sai"

Constranins: 1<=n, m<=100, các phần tử trong ma trận là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 2 8  9 0 19 24 3 8 12 14  5 20 6 2 11 16 20 17  2 8  2 11 16 4 2 3 17 7  21 16 18 20 22 1 21 13 | Ma tran tong  11 11 35 28 5 11 29 21  26 36 24 22 33 17 41 30 |
| 5 6  10 6 16 1 7 1  12 17 21 10 16 23  12 9 9 4 22 7  23 0 13 23 17 4  9 5 8 0 23 4  6 3  23 6 0  13 16 5  5 17 12  11 2 24  3 2 18  21 23 15 | Du lieu vao sai |

#include<stdio.h>

int main(){

int n1 , m1 , n2 , m2;

scanf("%d%d" , &n1 , &m1);

int a[n1][m1];

for(int i = 0 ; i < n1 ; i++){

for(int j = 0 ; j < m1 ; j++){

scanf("%d" , &a[i][j]);

}

}

scanf("%d%d" , &n2 , &m2);

int b[n2][m2];

for(int i = 0 ; i < n2 ; i++){

for(int j = 0 ; j < m2 ; j++){

scanf("%d" , &b[i][j]);

}

}

if(n1 == n2 && m2 == m1){

printf("Ma tran tong\n");

int c[n1][m1];

for(int i = 0 ; i < n1 ; i++){

for(int j = 0 ; j < m1 ; j++){

c[i][j] = a[i][j] + b[i][j];

}

}

for(int i = 0 ; i < n1 ; i++){

for(int j = 0 ; j < m1 ; j++){

printf("%d " , c[i][j]);

}

printf("\n");

}

}

else{

printf("Du lieu vao sai");

}

}

Question **10**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số thực. In ra vị trí của phần tử lớn nhất trong ma trận. Nếu có nhiều phần tử lớn nhất thì in ra phần tử có chỉ số hàng, chỉ số cột nhỏ nhất.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output: In ra thông báo "Phan tu lon nhat co chi so hang x, chi so cot y, gia tri Z"

Trong đó: x là chỉ số hàng, y là chỉ số cột và Z là giá trị của phần tử lớn nhất tìm được.

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số thực có 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 4  7.67 -2.33 -1.33 -6.67  2.00 9.00 -9.67 -8.33  7.00 11.00 -6.67 -9.33  2.67 4.67 -2.67 6.00  11.00 -5.33 -6.00 8.67  -7.67 -1.67 -5.00 11.00 | Phan tu lon nhat co chi so hang 2, chi so cot 1, gia tri 11.00 |

#include<stdio.h>

int main(){

int n , m;

scanf("%d%d" , &n , &m);

float a[n][m];

for(int i = 0 ; i < n ; i++){

for(int j = 0 ; j < m ; j++){

scanf("%f" , &a[i][j]);

}

}

int hang = 0 , cot = 0 ;

float max = -1e9;

for(int i = 0 ; i < n ; i++){

for(int j = 0 ; j < m ; j++){

if(a[i][j] > max){

max = a[i][j];

hang = i ;

cot = j ;

}

}

}

printf("Phan tu lon nhat co chi so hang %d, chi so cot %d, gia tri %.2f" , hang , cot , max);

}

Question **11**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số nguyên. In ra vị trí của phần tử lớn nhất của từng hàng trong ma trận. Nếu trong hàng có nhiều phần tử lớn nhất thì in ra phần tử có chỉ số cột nhỏ nhất.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output: gồm n dòng, mỗi dòng in ra 3 giá trị x y Z cách nhau dấu cách

Trong đó: x là chỉ số hàng, y là chỉ số cột và Z là giá trị của phần tử lớn nhất tìm được.

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6 21  5 0 18 19 12 19 4 7 0 9 20 22 13 11 7 4 0 23 16 19 11  6 17 9 10 2 18 23 4 6 8 7 20 18 13 13 3 5 3 2 17 5  4 20 16 10 5 10 10 16 11 24 12 13 9 3 19 4 23 0 14 14 9  20 23 15 13 17 2 17 1 11 24 12 24 0 5 0 14 8 4 0 11 16  21 23 24 0 10 24 4 4 8 18 22 18 13 15 24 15 17 4 10 10 14  24 11 14 1 9 14 3 13 7 15 22 19 11 21 13 14 1 1 15 21 23 | 0 17 23  1 6 23  2 9 24  3 9 24  4 2 24  5 0 24 |

#include <stdio.h>

int main (){

int n,m,a[100][100],i,j;

scanf("%d%d",&n,&m);

for ( i=0;i<n;i++){

for ( j=0;j<m;j++){

scanf ("%d",&a[i][j]);

}

}

for ( i=0 ;i<n;i++){

int max=-1;

int temp = 0;

for ( j=0;j<m;j++){

if (a[i][j]>max){

max=a[i][j];

temp=j;

}

}

printf ("%d %d %d \n",i,temp,max);

}

return 0;

}

Question **12**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một ma trận gồm n hàng, m cột các số thực. In ra vị trí của phần tử nhỏ nhất trong ma trận. Nếu có nhiều phần tử nhỏ nhất thì in ra phần tử có chỉ số hàng, chỉ số cột nhỏ nhất.

Input:

- Dòng thứ nhất nhập vào n và m

- Dòng tiếp theo nhập vào các phần tử của ma trận

Output: In ra 3 số x y Z cách nhau dấu cách.

Trong đó: x là chỉ số hàng, y là chỉ số cột và Z là giá trị của phần tử nhỏ nhất tìm được.

Constrains: 1<=n, m<=100, các phần tử trong ma trận là số thực có 2 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 4  2.67 -4.33 -1.33 -10.00  -3.67 -4.67 -10.00 10.67  3.67 10.00 -1.33 -4.00  3.67 1.00 -6.67 -5.00  7.00 -4.00 -0.33 -0.67 | 0 3 -10.00 |

#include <stdio.h>

int main (){

int n,m;

scanf("%d %d", &n,&m);

float a[n][m];

float min=20;

int vitrihang=0;

int vitricot=0;

for ( int i=0;i<n;i++){

for ( int j=0;j<m;j++){

scanf ("%f",&a[i][j]);

if(a[i][j]<min){

min=a[i][j];

vitrihang =i;

vitricot=j;

}

}

}

printf("%d %d %.2f", vitrihang,vitricot,min);

return 0;

}

Question **13**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Liệt kê các phần tử lớn hơn hoặc bằng trung bình cộng của các phần tử trong mảng.

Input: Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào mảng gồm N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra các phần tử tìm được, cách nhau dấu cách.

+ Dòng thứ hai in ra thông báo "So phan tu thoa man: X". Trong đó, X là số lượng các phần tử tìm được theo yêu cầu bài toán.

Constrains: Các phần tử trong mảng là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 32  24 14 1 3 5 19 23 17 15 8 0 22 4 2 4 10 20 11 13 4 10 4 24 6 7 10 19 15 21 1 3 16 | 24 14 19 23 17 15 22 20 13 24 19 15 21 16  So phan tu thoa man: 14 |

#include <stdio.h>

int main(){

int n;

int a[1000];

int tong=0;

int dem=0;

scanf("%d", &n);

for(int i=0;i<n;i++){

scanf("%d", &a[i]);

tong+=a[i];

}

float tbc=(float)tong/n;

for(int i=0;i<n;i++){

if(a[i] >= tbc){

printf("%d ", a[i]);

dem++;

}

}

printf("\nSo phan tu thoa man: %d",dem);

return 0;

}

Question **14**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Tìm vị trí và giá trị của phần tử lẻ nhỏ nhất trong mảng. Nếu có nhiều số thỏa mãn thì kết quả là số có số thứ tự nhỏ nhất.

Input: Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào mảng gồm N số nguyên cách nhau dấu cách.

Output:

+ In ra thông báo "So le nho nhat co vi tri X gia tri Y"//X là vị trí, Y là giá trị của phần tử tìm được.

+ Nếu trong mảng không có số lẻ thì thông báo "Khong co so le trong mang".

Constrains: Các phần tử trong mảng là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10  171 90 222 241 139 86 157 118 184 206 | So le nho nhat co vi tri 4 gia tri 139 |
| 11  88 204 16 20 46 248 14 242 22 174 132 | Khong co so le trong mang |

#include <stdio.h>

int main() {

int n;

scanf("%d", &n);

// Khai báo mảng và biến lưu giá trị phần tử lẻ nhỏ nhất

int arr[n];

int min\_odd = 1000000;

int min\_odd\_index = -1;

// Nhập mảng và tìm phần tử lẻ nhỏ nhất

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

if (arr[i] % 2 == 1 && arr[i] < min\_odd) {

min\_odd = arr[i];

min\_odd\_index = i;

}

}

// In ra kết quả tìm được

if (min\_odd\_index == -1) {

printf("Khong co so le trong mang\n");

} else {

printf("So le nho nhat co vi tri %d gia tri %d\n", min\_odd\_index, min\_odd);

}

return 0;

}

Question **15**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Sắp xếp tăng dần các phần tử lẻ trong mảng. Các phần tử chẵn không thay đổi vị trí ban đầu.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng kết quả cách nhau dấu cách.

Constrains:

+ Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  169 228 108 212 214 205 145 31 77 211 241 | 11  31 228 108 212 214 77 145 169 205 211 241 |

#include <stdio.h>

int main() {

int n;

scanf("%d", &n); // Nhập số lượng phần tử của mảng

int a[n];

for (int i = 0; i < n; i++) {

scanf("%d", &a[i]); // Nhập giá trị từng phần tử của mảng

}

for (int i = 0; i < n-1; i++) { // Duyệt mảng

for (int j = i+1; j < n; j++) {

if (a[i]%2==1 && a[j]%2==1 && a[i] > a[j]) { // Nếu phần tử a[i] và a[j] là số lẻ và a[i] > a[j]

int temp = a[i];

a[i] = a[j]; // Hoán đổi giá trị của a[i] và a[j]

a[j] = temp;

}

}

}

printf("%d\n", n); // In ra số lượng phần tử của mảng

for (int i = 0; i < n; i++) {

printf("%d ", a[i]); // In ra giá trị từng phần tử của mảng sau khi đã sắp xếp

}

printf("\n");

return 0;

}

Question **16**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Sắp xếp mảng đó tăng dần.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng (đã được sắp xếp tăng dần) cách nhau dấu cách.

Constrains:

+ Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  169 224 228 108 212 214 205 145 77 211 241 | 11  77 108 145 169 205 211 212 214 224 228 241 |

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void bubble\_sort(int arr[], int n) {

int i, j;

for (i = 0; i < n-1; i++) {

for (j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1])

swap(&arr[j], &arr[j+1]);

}

}

}

int main() {

int n, i;

scanf("%d", &n);

int arr[n];

for (i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

bubble\_sort(arr, n);

printf("%d\n", n);

for (i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

Question **17**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Sắp xếp tăng dần các phần tử chẵn trong mảng. Các phần tử lẻ không thay đổi vị trí ban đầu.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng kết quả cách nhau dấu cách.

Constrains:

+ Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  169 224 228 108 212 205 214 145 77 211 241 | 11  169 108 212 214 224 205 228 145 77 211 241 |

#include <stdio.h>

void swap(int \*x, int \*y) {

int temp = \*x;

\*x = \*y;

\*y = temp;

}

void bubbleSort(int arr[], int n) {

int i, j;

for (i = 0; i < n-1; i++) {

for (j = 0; j < n-i-1; j++) {

if (arr[j] > arr[j+1]) {

swap(&arr[j], &arr[j+1]);

}

}

}

}

int main() {

int n;

scanf("%d", &n);

int arr[n], sorted\_arr[n], i, j = 0;

for (i = 0; i < n; i++) {

scanf("%d", &arr[i]);

if (arr[i] % 2 == 0) {

sorted\_arr[j++] = arr[i];

}

}

bubbleSort(sorted\_arr, j);

j = 0;

for (i = 0; i < n; i++) {

if (arr[i] % 2 == 0) {

arr[i] = sorted\_arr[j++];

}

}

printf("%d\n", n);

for (i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

printf("\n");

return 0;

}

Question **18**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Sắp xếp tăng dần các phần tử lẻ trong mảng. Các phần tử chẵn không thay đổi vị trí ban đầu.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số nguyên cách nhau dấu cách.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng kết quả cách nhau dấu cách.

Constrains:

+ Các phần tử trong mảng là các số nguyên

**For example:**

| **Input** | **Result** |
| --- | --- |
| 11  169 228 108 212 214 205 145 31 77 211 241 | 11  31 228 108 212 214 77 145 169 205 211 241 |

#include <stdio.h>

void swap(int \*a, int \*b) {

int temp = \*a;

\*a = \*b;

\*b = temp;

}

void oddSort(int arr[], int n) {

int oddArr[n];

int oddIdx = 0;

for (int i = 0; i < n; i++) {

if (arr[i] % 2 == 1) {

oddArr[oddIdx] = arr[i];

oddIdx++;

}

}

for (int i = 0; i < oddIdx - 1; i++) {

for (int j = i + 1; j < oddIdx; j++) {

if (oddArr[i] > oddArr[j]) {

swap(&oddArr[i], &oddArr[j]);

}

}

}

int oddPtr = 0;

for (int i = 0; i < n; i++) {

if (arr[i] % 2 == 1) {

arr[i] = oddArr[oddPtr];

oddPtr++;

}

}

}

int main() {

int n;

scanf("%d", &n);

int arr[n];

for (int i = 0; i < n; i++) {

scanf("%d", &arr[i]);

}

oddSort(arr, n);

printf("%d\n", n);

for (int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}

return 0;

}

Question **21**

Correct

Marked out of 1.00

Flag question

Question text

Viết các hàm nhập vào mảng gồm n số thực, hàm in mảng đó ra màn hình, sắp xếp mảng số thực tăng dần.

Sử dụng các hàm ở trên viết chương trình nhập vào một mảng gồm N số thực, sắp xếp và in ra kết quả ra màn hình.

Input:

Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng đã được sắp xếp, cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Constrains: Các phần tử trong mảng là các số thực, độ chính xác 3 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  302.000 744.000 -302.000 744.000 32.000 | 5  -302.000 32.000 302.000 744.000 744.000 |

#include <stdio.h>

#include <stdlib.h>

int soSanh(const void\* a, const void\* b) {

float fa = \*(const float\*) a;

float fb = \*(const float\*) b;

if (fa > fb) {

return 1;

} else if (fa < fb) {

return -1;

} else {

return 0;

}

}

void nhapMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

scanf("%f", &mang[i]);

}

}

void inMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

printf("%.3f ", mang[i]);

}

printf("\n");

}

void sapXepMang(float mang[], int n) {

qsort(mang, n, sizeof(float), soSanh);

}

int main() {

int n;

scanf("%d", &n);

float mang[n];

nhapMang(mang, n);

sapXepMang(mang, n);

printf("%d\n", n);

inMang(mang, n);

return 0;

}

Question **10**

Correct

Marked out of 1.00

Flag question

Question text

Viết hàm kiểm tra một số nguyên có là số hoàn hảo hay không? Biết số hoàn hảo là một số nguyên dương mà tổng các ước nguyên dương chính thức của nó (số nguyên dương bị nó chia hết ngoại trừ nó) bằng chính nó. Ví dụ số 6 là số hoàn hảo vì 6=1+2+3.

Viết chương trình nhập vào hai số nguyên a, b. Tìm các số hoàn hảo trong đoạn [a, b] nếu a<=b, hoặc ngược lại. Nếu trong đoạn trên không có số hoàn hảo thì thông báo "Khong co".

Input: Số nguyên a, b kiểu int được nhập từ bàn phím cách nhau dấu cách. Ví dụ: -30 80

Output: In ra danh sách các số hoàn hảo tìm được cách nhau dấu cách hoặc thông báo "Khong co".

Contrain: Các biến kiểu int

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 200 | 6 28 |

#include <stdio.h>

#include <math.h>

int shh(int n){

if(n<=2){

return 0;

}

int sum=0;

for(int i=1;i<n;i++){

if(n%i==0){

sum+=i;

}

}

return sum == n;

}

int main(){

int a,b;

scanf("%d %d",&a,&b);

if(a>b){

int temp=a;

a=b;

b=temp;

}

int z=0;

for(int i=a;i<=b;i++){

if(shh(i)){

printf("%d ",i);

z=1;

}

}

if(!z){

printf("Khong co");

}

return 0;

}

Question **9**

Correct

Marked out of 1.00

Flag question

Question text

**Dãy Fibonacci** là dãy vô hạn các số tự nhiên được định nghĩa như sau:

Bạn hãy xây dựng hàm với đầu vào là số nguyên n (n>=1) đầu ra là số Fibonaci thứ n.

Viết chương trình (có sử dụng hàm ở trên) in ra M số Fibonaci đầu tiên.

Input: Số nguyên M >=1

Output: M số Fibonaci, cách nhau dấu cách.

Constrains: 1<= M <=100

**For example:**

| **Input** | **Result** |
| --- | --- |
| 9 | 1 1 2 3 5 8 13 21 34 |

#include <stdio.h>

int fibonacci(int n) {

if (n <= 1) {

return n;

}

int fibNMinus2 = 0;

int fibNMinus1 = 1;

int fibN = 0;

for (int i = 2; i <= n; i++) {

fibN = fibNMinus1 + fibNMinus2;

fibNMinus2 = fibNMinus1;

fibNMinus1 = fibN;

}

return fibN;

}

int main() {

int m;

scanf("%d", &m);

for (int i = 1; i <= m; i++) {

printf("%d ", fibonacci(i));

}

return 0;

}

Question **8**

Not complete

Marked out of 1.00

Flag question

Question text

**Bạn hãy viết hàm:**

**1) Đếm số chữ số của một số nguyên dương n.**

**2) TÍnh tổng các chữ số của số nguyên dương n.**

**Viết chương trình nhập vào một số n. Nếu n là số âm thì in ra thông báo "Du lieu sai.". Ngược lại in ra số chữ số và tổng các chữ số của nó trên cùng một dòng, cách nhau dấu cách.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190**

**+ Output: 3 10**

**Example 2:**

**+ Input: -190**

**+ Output: Du lieu sai.**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 | 3 10 |

#include <stdio.h>

int countDigits(int n) {

int count = 0;

while (n != 0) {

n /= 10;

count++;

}

return count;

}

int sumDigits(int n) {

int sum = 0;

while (n != 0) {

sum += n % 10;

n /= 10;

}

return sum;

}

int main() {

int n, numDigits, digitSum;

scanf("%d", &n);

if (n < 0) {

printf("Du lieu sai.");

} else {

numDigits = countDigits(n);

digitSum = sumDigits(n);

printf("%d %d", numDigits, digitSum);

}

return 0;

}

Question **7**

Correct

Marked out of 1.00

Flag question

Question text

Viết hàm kiểm tra một số nguyên có là số nguyên tố không. Biết số nguyên tố là số nguyên dương >=2 và chỉ chia hết cho 1 và chính nó.

Viết chương trình nhập vào 2 số nguyên a, b. In ra các số nguyên tố trong đoạn [a, b], nếu a<=b, hoặc ngược lại. Nếu trong đoạn này không có số nguyên tố thì in ra thông báo "Khong co".

Input: a, b là số nguyên cách nhau dấu cách.

Output: các số nguyên tố tìm được viết trên một dòng, cách nhau dấu cách. Hoặc thông báo "Khong co".

Constrains: các số kiểu int

Ví dụ 1:

+ Input: 12 18

+ Output: 13 17

Ví dụ 2:

+ Input: 18 12

+ Output: 17 13

Ví dụ 3:

+ Input: 14 16

+ Output: Khong co

**For example:**

| **Input** | **Result** |
| --- | --- |
| -9 -33 | Khong co |
| 28 18 | 23 19 |

#include <stdio.h>

#include <math.h>

int main(){

int a, b;

scanf("%d %d", &a, &b);

if(a == b)

{

if(a < 2)

{

printf("Khong co");

}

else

{

int lasonguyento = 1;

int i;

for( i = 2; i <= a - 1; i = i + 1)

{

if(a % i == 0)

{

lasonguyento = 0;

break;

}

}

if(lasonguyento == 1)

{

printf("%d", a);

}

else

{

printf("Khong co");

}

}

}

else if(a > b)

{

int daykhongcosonguyento = 1;

int i;

for(i = a; i >= b; i = i - 1)

{

if(i < 2)

{

}

else

{

int lasonguyento = 1;

int j;

for( j = 2; j <= i - 1; j = j + 1)

{

if(i % j == 0)

{

lasonguyento = 0;

break;

}

}

if(lasonguyento == 1)

{

daykhongcosonguyento = 0;

printf("%d ", i);

}

else

{

}

}

}

if(daykhongcosonguyento == 1)

{

printf("Khong co");

}

}

else //a < b

{

int daykhongcosonguyento = 1;

int i;

for(i = a; i <= b; i = i + 1)

{

if(i < 2)

{

}

else

{

int lasonguyento = 1;

int j;

for( j = 2; j <= i - 1; j = j + 1)

{

if(i % j == 0)

{

lasonguyento = 0;

break;

}

}

if(lasonguyento == 1)

{

daykhongcosonguyento = 0;

printf("%d ", i);

}

else

{

}

}

}

if(daykhongcosonguyento == 1)

{

printf("Khong co");

}

}

}

Question **9**

Correct

Marked out of 1.00

Flag question

Question text

**Bạn hãy viết hàm: Kiểm tra các chữ số của một số nguyên dương n có toàn là số chẵn không.**

**Viết chương trình nhập vào ba số nguyên x, y, z. In ra "Yes" nếu các chữ số của x (hoặc y, z) toàn là số chẵn. In ra "No" nếu các chữ số của x (hoặc y, z) có ít nhất một số lẻ. In ra "Error" nếu x (hoặc y, z) là số âm. Thông báo trên cùng một dòng, cách nhau dấu cách.**

**Input: 3 số x, y, z là các số nguyên được nhập từ bàn phím.**

**Output: như yêu cầu.**

**Constrain: các số kiểu int.**

**Example 1:**

**+ Input: 190 -2 88**

**+ Output: No Error Yes**

**Example 2:**

**+ Input: 190 172 111**

**+ Output: No No No**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 109 -2 88 | No Error Yes |

#include <stdio.h>

void kiemtra(int n){

if(n<0){

printf("Error");

}else {

while(n!=0){

int a =n%10;

if(a%2!=0){

printf("No");

return ;

}

n/=10;

}

printf("Yes");

}

}

int main(){

int x,y,z;

scanf("%d%d%d",&x,&y,&z);

kiemtra(x);

printf(" ");

kiemtra(y);

printf(" ");

kiemtra(z);

printf(" ");

return 0;

}

**Code số nguyên tố:**

**int check(int n){**

**int dem;**

**for(int i=2;i<=sqrt(n);i++){**

**if(n%i==0) return 0;**

**}**

**return n>1;**

**}**

Question **15**

Correct

Marked out of 1.00

Flag question

Question text

Viết chương trình nhập vào một mảng các số nguyên. Tìm vị trí và giá trị của phần tử chẵn lớn nhất trong mảng. Nếu có nhiều số thỏa mãn thì kết quả là số có số thứ tự nhỏ nhất.

Input: Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào mảng gồm N số nguyên cách nhau dấu cách.

Output:

+ In ra thông báo "So chan lon nhat co vi tri X gia tri Y"//X là vị trí, Y là giá trị của phần tử tìm được.

+ Nếu trong mảng không có số chẵn thì thông báo "Khong co so chan trong mang".

Constrains: Các phần tử trong mảng là các số nguyên.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 13  163 195 75 79 148 148 237 41 175 147 24 122 6 | So chan lon nhat co vi tri 4 gia tri 148 |
| 11  881 2041 1671 2011 465 2483 141 2421 2371 1741 131 | Khong co so chan trong mang |

**#include <stdio.h>**

**int main() {**

**int n, a[1000], max\_even = -1, pos = -1;**

**scanf("%d", &n);**

**for (int i = 0; i < n; i++) {**

**scanf("%d", &a[i]);**

**if (a[i] % 2 == 0 && a[i] > max\_even) {**

**max\_even = a[i];**

**pos = i;**

**}**

**}**

**if (pos == -1) {**

**printf("Khong co so chan trong mang");**

**} else {**

**printf("So chan lon nhat co vi tri %d gia tri %d", pos, max\_even);**

**}**

**return 0;**

**}**

Question **16**

Correct

Marked out of 1.00

Flag question

Question text

Viết các hàm nhập vào mảng gồm n số thực, in ra mảng, sắp xếp mảng số thực tăng dần.

Viết chương trình (sử dụng các hàm trên) thực hiện nhập, sắp xếp mảng và in kết quả ra màn hình.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Output:

+ Dòng thứ nhất in ra thông báo "Mang sap xep"

+ Dòng thứ hai in ra các phần tử của mảng đã được sắp xếp, cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Constrains: Các phần tử trong mảng là các số thực, độ chính xác 3 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  302.000 744.000 -302.000 744.000 32.000 | Mang sap xep  -302.000 32.000 302.000 744.000 744.000 |

#include <stdio.h>

#include <stdlib.h>

void nhapMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

scanf("%f", &mang[i]);

}

}

void inMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

printf("%.3f ", mang[i]);

}

printf("\n");

}

int cmp(const void\* a, const void\* b) {

float fa = \*(const float\*)a;

float fb = \*(const float\*)b;

return (fa > fb) - (fa < fb);

}

void sapXepMang(float mang[], int n) {

qsort(mang, n, sizeof(float), cmp);

}

int main() {

int n;

scanf("%d", &n);

float\* mang = (float\*)malloc(n \* sizeof(float));

nhapMang(mang, n);

printf("Mang sap xep\n");

sapXepMang(mang, n);

inMang(mang, n);

free(mang);

return 0;

}

Question **17**

Correct

Marked out of 1.00

Flag question

Question text

Viết các hàm nhập vào mảng gồm n số thực, hàm in mảng đó ra màn hình, sắp xếp mảng số thực tăng dần.

Sử dụng các hàm ở trên viết chương trình nhập vào một mảng gồm N số thực, sắp xếp và in ra kết quả ra màn hình.

Input:

Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Output:

+ Dòng thứ nhất in ra số phần tử của mảng (N)

+ Dòng thứ hai in ra các phần tử của mảng đã được sắp xếp, cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Constrains: Các phần tử trong mảng là các số thực, độ chính xác 3 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  302.000 744.000 -302.000 744.000 32.000 | 5  -302.000 32.000 302.000 744.000 744.000 |

#include <stdio.h>

#include <stdlib.h>

void nhapMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

scanf("%f", &mang[i]);

}

}

void inMang(float mang[], int n) {

for (int i = 0; i < n; i++) {

printf("%.3f ", mang[i]);

}

printf("\n");

}

int cmp(const void\* a, const void\* b) {

float fa = \*(const float\*)a;

float fb = \*(const float\*)b;

return (fa > fb) - (fa < fb);

}

void sapXepMang(float mang[], int n) {

qsort(mang, n, sizeof(float), cmp);

}

int main() {

int n;

scanf("%d", &n);

float\* mang = (float\*)malloc(n \* sizeof(float));

nhapMang(mang, n);

printf("%d\n", n);

sapXepMang(mang, n);

inMang(mang, n);

free(mang);

return 0;

}

Question **18**

Correct

Marked out of 1.00

Flag question

Question text

Viết các hàm nhập vào mảng gồm n số thực, hàm tìm vị trí phần tử lớn nhất trong mảng. Nếu có nhiều hơn một phần tử lớn nhất thì lấy phần tử có vị trí nhỏ nhất.

Sử dụng các hàm ở trên viết chương trình nhập một mảng các số thực. Sau đó in ra vị trí và giá trị phần tử lớn nhất trong mảng.

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

Output:

+ In ra thông báo "Phan tu lon nhat co vi tri x, co gia tri y"

Trong đó: x, y là vị trí và giá trị của phần tử lớn nhất trong mảng.

Constrains: Các phần tử trong mảng là các số thực, độ chính xác 3 chữ số thập phân.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  302.000 744.000 -302.000 744.000 32.000 | Phan tu lon nhat co vi tri 1, co gia tri 744.000 |

#include <stdio.h>

void nhapMang(float arr[], int n) {

int i;

for (i = 0; i < n; i++) {

scanf("%f", &arr[i]);

}

}

int timViTriMax(float arr[], int n) {

int i, viTriMax = 0;

for (i = 1; i < n; i++) {

if (arr[i] > arr[viTriMax]) {

viTriMax = i;

}

}

return viTriMax;

}

int main() {

int n, viTriMax;

float arr[1000], max;

scanf("%d", &n);

nhapMang(arr, n);

viTriMax = timViTriMax(arr, n);

max = arr[viTriMax];

printf("Phan tu lon nhat co vi tri %d, co gia tri %.3f", viTriMax, max);

return 0;

}

Question **19**

Correct

Marked out of 1.00

Flag question

Question text

Viết các hàm nhập vào mảng gồm n số thực, tìm vị trí khóa key trong mảng. Nếu có nhiều hơn một phần bằng key thì lấy vị trí nhỏ nhất. Nếu không có khóa key trong mảng thì hàm trả về giá trị -1.

Sử dụng các hàm ở trên viết chương trình nhập một mảng các số thực. Sau đó nhập vào một số thực X, in ra vị trí của X trong mảng nếu tìm thấy. Ngược lại in ra thông báo " X not found".

Input:

+ Dòng thứ nhất nhập vào số nguyên N, 1 <= N <= 1000

+ Dòng thứ hai nhập vào N số thực cách nhau dấu cách, độ chính xác 3 chữ số thập phân.

+ Dòng thứ ba nhập vào số thực X.

Output:

+ In ra thông báo "Phan tu X co vi tri y" hoặc "X not found"

Trong đó: X là khóa cần tìm, y là vị trí của X trong mảng.

Constrains: Các phần tử trong mảng là các số thực, độ chính xác 3 chữ số thập phân.

Examples:

+ Input:

5 //N

302.000 744.000 -302.000 744.000 32.000 //Mảng

320.000 //khóa X

+ Output: 320.000 not found

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  302.000 744.000 -302.000 744.000 32.000  32.000 | Phan tu 32.000 co vi tri 4 |

#include <stdio.h>

void nhapMang(float arr[], int n) {

int i;

for (i = 0; i < n; i++) {

scanf("%f", &arr[i]);

}

}

int timViTri(float arr[], int n, float key) {

int i, viTri = -1;

for (i = 0; i < n; i++) {

if (arr[i] == key) {

viTri = i;

break;

}

}

return viTri;

}

int main() {

int n, viTri;

float arr[1000], key;

scanf("%d", &n);

nhapMang(arr, n);

scanf("%f", &key);

viTri = timViTri(arr, n, key);

if (viTri == -1) {

printf("%.3f not found", key);

} else {

printf("Phan tu %.3f co vi tri %d", key, viTri);

}

return 0;

}