BÀI TẬP VẬN DỤNG

Tên file:sapxep.cpp/sapxep.py

**Bài 1:** Cho một dãy số thực gồm N phần tử (n<=109) Bạn hãy sắp các phần tử theo thứ tự giảm dần sau đó in danh sách n phần tử sau khi đã sắp xếp ra màn hình kèm với đó là vị trí của nó trong dãy ban đầu.

Dữ liệu vào: Dòng 1 là số nguyên N, các dòng còn lại là dãy số thực.

Kết quả: Dòng 1 là danh sách n số thực đã sắp xếp, dòng 2 là vị trí của từng phần tử mới trong dãy ban đầu.

Ví dụ:

|  |  |
| --- | --- |
| Sapxep4.inp | Sapxep4.out |
| 5  4 2 6 4 9 | 9 6 4 4 2  5 3 1 4 2 |

#include <bits/stdc++.h>

using namespace std;

struct sx

{

    int id, dl;

};

bool cmp(sx *x*, sx *y*)

{

    return x.dl > y.dl;

}

int main()

{

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    int n;

    cin >> n;

    sx a[n];

    for (int i = 0; i < n; i++)

    {

        cin >> a[i].dl;

        a[i].id = i + 1;

    }

    sort(a, a + n, cmp);

    for (int i = 0; i < n; i++)

    {

        cout << a[i].dl << " ";

    }

    cout << endl;

    for (int i = 0; i < n; i++)

    {

        cout << a[i].id << " ";

    }

*// vector<int> res(n);*

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

*// {*

*//     res[i] = a[i].id;*

*// }*

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

*// {*

*//     cout << res[i] << " ";*

*// }*

*// cout << endl;*

    return 0;

}

Bài 2: Sửa máy tên file suamay.cpp

Một cơ sở sửa chữa ô tô có nhận ***n*** chiếc xe để sửa. Do các nhân viên làm việc quá lười nhác nên đã đến hạn trả cho khách hàng mà vẫn chưa tiến hành sửa được chiếc xe nào. Theo hợp đồng đã ký kết từ trước, nếu bàn giao xe thứ ***i*** quá hạn ngày nào thì sẽ phải trả thêm một khoản tiền phạt là ***a[i] (i=1..n)***.

Ông chủ cơ sở sửa chữa quyết định sa thải toàn bộ công nhân và thuê nhân công mới. Với lực lượng mới này, ông ta dự định rằng để sửa chiếc xe thứ ***i*** sẽ cần ***b[i] (i=1..n)*** ngày. Vấn đề đặt ra đối với ông là phải lập lịch sửa tuần tự các chiếc xe sao cho tổng số tiền bị phạt là ít nhất.

**Dữ liệu:** file văn bản **SUAMAY.INP**

- Dòng đầu ghi số nguyên ***n*** 

- Dòng thứ hai ghi ***n*** số nguyên ***a[1], a[2],..,a[n]***

- Dòng thứ ba ghi ***n*** số nguyên ***b[1], b[2],..,b[n]***

**Kết quả:** file văn bản **SUAMAY.OUT**

- Dòng đầu ghi một số duy nhất chi phí tối thiểu.

- Dòng thứ hai ghi lần lượt số hiệu các xe cần sửa

*\* Các số trong file cách nhau một dâu cách*

Ví dụ:

|  |  |
| --- | --- |
| **SUAMAY.INP** | **SUAMAY.OUT** |
| 4  1 3 4 2  3 2 3 1 | 44  4 2 3 1 |

Giải thích:

* Xong công việc 4 vào cuối ngày 1 => phải trả 2 \* 1 = 2 .
* Xong công việc 2 vào cuối ngày 3 => phải trả 3 \* 3 = 9.
* Xong công việc 3 vào cuối ngày 6 => phải trả 6 \* 4 = 24 .
* Xong công việc 1 vào cuối ngày 9 => phải trả 1 \* 9 = 9 .
* Vậy tổng cộng phải trả 44 .

#include <bits/stdc++.h>

using namespace std;

struct Xe

{

    int phat;

    int time;

    int id;

};

bool cmp(Xe *x*, Xe *y*)

{

    long long tilex = x.phat / x.time;

    long long tiley = y.phat / y.time;

    return tilex > tiley;

}

int main()

{

    freopen("in.inp", "r", stdin);

    freopen("out.out", "w", stdout);

    int n;

    cin >> n;

    vector<Xe> a(n);

    for (int i = 0; i < n; i++)

    {

        cin >> a[i].phat;

        a[i].id = i + 1;

    }

    for (int i = 0; i < n; i++)

    {

        cin >> a[i].time;

    }

    sort(a.begin(), a.end(), cmp);

    long long tong\_phat = 0;

    long long thoi\_gian = 0;

    vector<int> res(n);

    for (int i = 0; i < n; i++)

    {

        res[i] = a[i].id;

        thoi\_gian += a[i].time;

        tong\_phat += thoi\_gian \* a[i].phat;

    }

    cout << tong\_phat << endl;

    for (int i = 0; i < n; i++)

    {

        cout << res[i] << " ";

    }

    return 0;

}

**Giới hạn:**  ***1<=n<=1000; 1<=a[i]<=100; 1<=b[j]<=100;***

### Bài 3: Bí kíp luyện rồng Tên file: dragon.cpp hoặc dragon.pas

Sau khi không còn gì để đố 2 thằng em của mình nữa Tý quay ra chơi điện tử. Tý rất mê trò chơi điều khiển nhân vật Hiccup của mình chiến đấu cùng với những con rồng, vượt qua tất cả sẽ nhận được chứng chỉ dũng sĩ dragon. Hiccup chỉ có thể vượt qua 1 con rồng nếu như chỉ số chiến đấu của cậu lớn hơn chỉ số chiến đấu của con rồng đó, và sau khi giành chiến thắng chỉ số chiến đấu của cậu sẽ tăng thêm một mức nhất định tùy vào con rồng mà cậu đánh bại.

Tý muốn biết liệu Hiccup có thể nhận được chứng chỉ không? Bạn hãy giúp Tý với.

INPUT:

* Dòng đầu tiên gồm 2 số nguyên s và n, trong đó n là số con rồng Hiccup cần phải vượt qua (1 ≤ n ≤ 103) và s là chỉ số chiến đấu ban đầu của Hiccup (1 ≤ s ≤ 104).
* n dòng tiếp theo, dòng thứ i chứa cặp số (x[i], y[i]). Trong đó x[i] là chỉ số chiến đấu của con rồng thứ i (1 ≤ x[i] ≤ 104), y[i] là mức tăng chỉ số cho Hiccup nếu cậu đánh bại nó (0 ≤ y[i] ≤ 104).

OUTPUT:

* In ra “YES” nếu Hiccup nhận được chứng chỉ dũng sĩ dragon.
* Trường hợp ngược lại thì:
* Dòng 1 ghi ”NO” nếu Hiccup không nhận được chứng chỉ dũng sỹ dragon.
* Dòng 2 đưa ra số lượng rồng còn lại chưa bị tiêu diệt.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
| Test 1 | | Test 2 | |
| INPUT | OUTPUT | INPUT | OUTPUT |
| 2 2  1 99  100 0 | YES | 10 1  100 100 | NO  1 |
|  |  |  |  |

### Bài 4: Bài tập về nhà Tên file: homework.cpp hoặc homework.pas

Bạn An được thầy giáo giao cho N bài tập về nhà, mỗi bài tập có mức độ khó khác nhau được đánh giá bởi giá trị a[i].

Để làm được bài tập thứ i thì bạn An cần phải có kỹ năng lớn hơn hoặc bằng mức độ khó a[i]. Mỗi khi làm được bài độ khó là a[i] thì kỹ năng giải bài của cậu được tăng thêm một lượng là a[i]. Ban đầu kỹ năng làm bài của bạn An là S.

Hãy cho biết bạn An có thể làm được tối đa bao nhiêu bài tập.

INPUT: HOMEWORK.INP

* Dòng 1 chứa N và S (1 <= N <= 100, S <= 1000)
* Dòng 2 chứa N số nguyên a[1], a[2], ..., a[N]

Output: HOMEWORK.OUT

* Một số nguyên duy nhất là số lượng bài tập lớn nhất làm được.

Ví dụ

|  |  |
| --- | --- |
| HOMEWORK.INP | HOMEWORK.OUT |
| 5 2  18 2 5 14 3 | 3 |

### Bài 5: Vắt sữa bò Tên file: milk2.cpp hoặc milk2.pas

Vào một buổi sáng nông dân John sắp một đàn bò gồm ***n*** con bò để vắt sữa. Ông dự kiến là vào sáng hôm đó, con bò thứ ***i*** có khả năng sẽ vắt được ***ai*** lít sữa. Tuy nhiên đàn bò của ông có đặc điểm là cứ mỗi lần vắt sữa một con, những con còn lại trông thấy sợ quá nên sẽ bị giảm sản lượng mỗi con **01** lít sữa. Nếu vắt sữa con bò thứ nhất, ***n-1*** con còn lại bị giảm sản lượng. Sau đó vắt sữa con bò thứ hai thì ***n-2*** con còn lại bị giảm sản lượng....

Bạn hãy giúp nông dân John tính xem thứ tự vắt sữa bò như thế nào để số lượng sữa vắt được là nhiều nhất nhé.

Dữ liệu vào: gồm 2 dòng

- Dòng thứ nhất là số nguyên n (1 ≤ n ≤ 100) là số lượng con bò.

- Dòng thứ hai gồm n số nguyên a1, a2,..., an (1 ≤ ai ≤ 1000) là sản lượng sữa của các con bò.

Dữ liệu xuất:

- Là một số nguyên xác định số lít sữa nhiều nhất mà anh Bo có thể vắt được.

Ví dụ

|  |  |
| --- | --- |
| Milk2.INP | Milk2.OUT |
| 4 4 4 4 4 | 10 |
| 4 2 1 4 3 | 6 |

### Bài 6: Xếp gạch Tên file: tile.cpp hoặc tile.pas

Nam có ***n*** viên gạch được đánh số từ ***1*** đến ***n***. Các viên gạch có độ cứng lần lượt là ***a1***, ***a2***,..., ***an***. Một viên gạch có độ cứng ***x*** nghĩa là Nam có thể chồng lên trên viên gạch đó tối đa ***x*** viên gạch khác, nếu chồng nhiều hơn thì viên gạch đó bị vỡ. Hỏi Nam có thể sắp được chồng gạch cao nhất là bao nhiêu?

Dữ liệu nhập:

- Dòng đầu tiên là số nguyên ***n*** (1 ≤ ***n*** ≤ 100) - là số viên gạch.

- Dòng tiếp theo gồm ***n*** số nguyên ***a1***, ***a2***,..., ***an*** (0 ≤ ***ai*** ≤ 100) mỗi số cách nhau một khoảng trắng.

Dữ liệu xuất

- Là số nguyên xác định chiều cao cao nhất của chồng gạch mà Nam sắp được.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 3  1 2 1 | 3 |
| 6  0 0 0 0 0 0 | 1 |

**Bài 7: Thu mua sữa bò**  tên file: milk.cpp hoặc milk.pas

Một nhà máy chế biến sữa cần mua hàng ngày N lít sữa của M nông dân. Mỗi nông dân có thể cung cấp một lượng sữa và đưa ra giá bán sữa khác nhau.

Hãy tìm cách mua sữa sao cho đủ số lượng sữa cần cho hàng ngày với số tiền bỏ ra là ít nhất. Biết rằng tổng lượng sữa của các nông dân luôn đủ cung cấp cho nhà máy.

Dữ liệu vào:

* Dòng đầu tiên chứa hai số N (0 <= N <= 2.000.000) và M (0 <= M <= 5,000)
* M dòng tiếp theo, mỗi dòng chưa hai số nguyên Pi (0 <= Pi <= 1,000) và Ai (0 <= Ai <= 2,000,000), trong đó Pi là số tiền mà nông dân bán 1 lít, Ai là số lượng sữa mà nông dân thứ i có thể cung cấp

Kết quả:

* Một số nguyên duy nhất là số tiền ít nhất để mua sữa

Ví dụ

|  |  |  |
| --- | --- | --- |
| Milk.inp | Milk.out | Giải thích |
| 100 5  5 20  9 40  3 10  8 80  6 30 | 630 | Mua 20 lít sữa của nông dân 1 => 5\*20 = 100  Không mua sữa của nông dân 2  Mua 10 lít sữa của nông dân 3 => 3\*10 = 30  Mua 40 lít sữa của nông dân 4 => 8\*40 = 320  Mua 30 lít sữa của nông dân 5 => 6\*30 = 180  Tổng số tiền mua = 100 + 30 + 320 + 180 = 630 |

### Bài 8: Cắm điện Tên file: election.cpp hoặc election.pas

Trong nhà Nam hiện đang có ***n*** ổ cắm điện rời. Số lượng chỗ cắm trên mỗi ổ cắm điện này lần lượt là ***a1, a2, a3,…, an***  chỗ cắm. Trên tường nhà Nam có một chỗ cắm cố định đang có điện. Vậy để cho một ổ cắm điện rời có điện thì phải cắm ổ cắm đó vào chỗ cắm cố định trên tường. Chúng ta cũng có thể cắm ổ cắm điện rời này vào một ổ cắm điện rời khác đang có điện.

Nam có ***m*** thiết bị sử dụng điện, để sử dụng thì các thiết bị này cần được cắm vào ổ cắm trên tường hoặc ổ cắm rời đang có điện.

Bạn hãy giúp Nam tìm ra số ổ cắm rời ít nhất cần dùng để có thể sử dụng tất cả ***m*** thiết bị điện này.

Dữ liệu vào:

- Dòng thứ nhất gồm 2 số nguyên ***n, m*** cách nhau một khoảng trắng, dữ liệu vào đảm bảo 1 ≤ ***n, m*** ≤ 50, ***n*** là số lượng ổ cắm và ***m*** là số lượng thiết bị.

- Dòng thứ hai gồm n số nguyên ***a1, a2, …, an*** là số chỗ cắm trên các ổ cắm rời tương ứng, mỗi số cách nhau một khoảng trắng, dữ liệu vào đảm bảo 1 ≤ ***ai*** ≤ 50.

Dữ liệu ra:

- Là số nguyên cho biết số ổ cắm rời ít nhất cần sử dụng là bao nhiêu. Nếu đã sử dụng hết tất cả ổ cắm rời mà vẫn không đủ, in ra -1.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 4 7  3 3 2 4 | 3 |
| 5 5  1 3 1 2 1 | -1 |

### Bài 9: Đội cờ Tên file: chess.cpp hoặc chess.pas

Có hai đội cờ vua A và B thi đấu với nhau. Mỗi đội cờ cử ra ***n*** kỳ thủ, mỗi kỳ thủ của đội B chỉ dấu một trận và chỉ đấu với một kỳ thủ của đội A và ngược lại. Vậy có tất cả ***n*** trận đấu. Đội nào thắng được 2 điểm, hoà được 1 điểm và thua được 0 điểm.

Cho đội B được quyền chọn cặp thi đấu.

**Yêu cầu:** Lập trình để đội B chọn dược các cặp thi đấu sao cho tổng số điểm của đội B là cao nhất. Cho biết trình đội của kỳ thủ thứ i của đội A và B lần lượt là a[i] và b[i] (i=1,2,…,n) và giả sử trong thi đấu, hai kỳ thủ có trình độ ngang nhau sẽ hoà và kỳ thủ nào có trình độ cao hơn sẽ thắng.

INPUT:

- Dòng đầu tiên ghi số nguyên dương n, 1≤n≤1000

- Dòng thứ i+1 ghi 2 số nguyên a[i], b[i] (1 ≤ a[i], b[i]≤100) cách nhau ít nhất một khoảng trắng.

OUTPUT

- Dòng đầu ghi số nguyên T là tổng số điểm cao nhất mà đội B có thể đạt được.

Ví dụ:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| 4  7 8  5 6  4 3  9 4 | 5 |

\* Giải thích:

Các sắp xếp cặp thi đấu của đội A và B như sau:

1 – 1, 2 – 2, 3 – 4, 4 – 3 trong đó số thứ nhất thuộc đội A, số thứ hai thuộc đội B.

2.4.4.

### Taxi Tên file: taxi.cpp hoặc taxi.pas

**Bài 10:** Sau buổi học, có ***n*** nhóm bạn rủ nhau đến chúc mừng sinh nhật An. Nhóm bạn thứ ***i*** có ***ai*** người (1 ≤ ***ai*** ≤ 4). Họ quyết định gọi những chiếc xe taxi để đi. Những chiếc taxi chỉ chở tối đa được 4 khách mà thôi.

Hãy cho biết các nhóm bạn này cần gọi tối thiểu bao nhiêu chiếc taxi, biết rằng những người trong cùng một nhóm luôn đi cùng trên một chiếc taxi, và một chiếc taxi có thể chở nhiều nhóm khác nhau.

Input:

- Dòng 1: chứa số nguyên dương ***n*** (1 ≤ ***n*** ≤ 105)

- Dòng 2: chứa ***n*** số nguyên dương ***a1, a2, … an*** – là số lượng người của mỗi nhóm bạn.

Output

- Một số nguyên dương duy nhất là số lượng taxi.

Ví dụ:

|  |  |
| --- | --- |
| Input | Output |
| 5  1 2 4 3 3 | 4 |
| 8  2 3 4 4 2 1 3 1 | 5 |

 Đọc vào số lượng nhóm n và số lượng người trong mỗi nhóm groups.

 Đếm số lượng nhóm có 1, 2, 3, và 4 người.

 Ban đầu, số lượng taxi tối thiểu bằng số lượng nhóm có 4 người (count4).

 Ghép các nhóm có 3 người với các nhóm có 1 người. Nếu số nhóm có 1 người lớn hơn hoặc bằng số nhóm có 3 người, trừ đi số lượng nhóm có 1 người bằng với số nhóm có 3 người. Nếu không, đặt số nhóm có 1 người về 0.

 Ghép các nhóm có 2 người với nhau. Nếu còn dư một nhóm có 2 người, cần thêm một taxi nữa và cố gắng ghép thêm 2 nhóm có 1 người vào taxi này.

 Cuối cùng, ghép các nhóm có 1 người với nhau, mỗi taxi chở tối đa 4 người. Nếu còn dư nhóm có 1 người, tính số lượng taxi cần thêm bằng cách làm tròn lên (sử dụng (count1 + 3) / 4).