

Hướng Dẫn Kỳ Thi Olympic Tin học Sinh Viên Toàn Quốc & ICPC 2025

Đặng Phúc An Khang*

Ngày 12 tháng 3 năm 2025

Tóm tắt nội dung

Báo cáo về quá trình ôn luyện lập trình thi đấu hướng đến kỳ thi OLP Tin học Sinh viên toàn quốc & ICPC của tôi - Đặng Phúc An Khang. Làm theo chỉ đạo của thầy Nguyễn Quân Bá Hồng.

Code:

- C/C++: <https://github.com/GrootTheDeveloper/OLP-ICPC/tree/master/2025/C%2B%2B>.
- Python:

Tài khoản trên các Online Judge:

- Codeforces: <https://codeforces.com/profile/vuivethoima>.
- VNOI: oj.vnoi.info/user/Groot.
- IUHCoder: oj.iuhcoder.com/user/ankhang2111.
- MarisaOJ: <https://marisaoj.com/user/grootsiuvip/submissions>.
- CSES: <https://cses.fi/user/212174>.
- UTOJ: sot.umtoj.edu.vn/user/grootsiuvip.
- SPOJ: www.spoj.com/users/grootsiuvip/.
- POJ: http://poj.org/userstatus?user_id=vuivethoima.
- ATCoder: <https://atcoder.jp/users/grootsiuvip>
- OnlineJudge.org: [vuivethoima](https://onlinejudge.org/players/vuivethoima)
- updating...

Mục lục

1 Preliminaries – Kiến thức chuẩn bị	2
2 WANG & WU 2016	2
2.1 SECTION 1: FUNDAMENTAL PROGRAMMING SKILLS	2
2.1.1 Financial Management	2
2.1.2 Doubles	3
2.1.3 Sum of Consecutive Prime Numbers	4
2.1.4 I Think I Need a Houseboat	6
2.1.5 Hangover	8
2.1.6 Humidex	9
2.1.7 Sum	10
2.1.8 Specialized Four-Digit Numbers	11
2.1.9 Quicksum	12
2.1.10 A Contesting Decision	13
2.1.11 Dirichlet's Theorem on Arithmetic Progressions	15
2.1.12 The Circumference of the Circle	17
2.1.13 Vertical Histogram	18
2.1.14 Ugly Numbers	20
2.1.15 Number Sequence	21
2.1.16 Đánh Giá Chủ Quan Chap 1 - Section 1	22
2.1.17 Speed Limit	22
2.1.18 Ride to School	23
2.1.19 Self-Numbers	24
2.1.20 Bee	25
2.1.21 Gold Coins	27
2.1.22 The $3n + 1$ Problem	28
2.1.23 Pascal Library	29

*E-mail: ankhangluonvuituoi@gmail.com. Tây Ninh, Việt Nam.

2.1.24	Calendar	30
2.1.25	Manager	32
2.1.26	Đánh Giá Chủ Quan Chap 2 - Section 1	34
3	Linear Algebra – Chuyển bài thi OLP Toán học môn ĐSTT về lập trình	34
3.1	Dại số 2023 - B.4.	34
4	Miscellaneous	36
4.1	Contributors	36

1 Preliminaries – Kiến thức chuẩn bị

Resources – Tài nguyên.

1. [WANG_WU2016]. WANG & Wu. *Data Structure Practice for Collegiate Programming Contests and Education*. https://github.com/GrootTheDeveloper/OLP-ICPC/blob/master/Wu_Wang2016.pdf
2. [WANG_WU2019]. WANG & Wu. *Algorithm Design Practice for Collegiate Programming Contests and Education*. https://github.com/GrootTheDeveloper/OLP-ICPC/blob/master/Wu_Wang2019.pdf

Personal Critical-thinking Questions:

Question 1 (Expanding my problem-solving approach). *What common problem-solving techniques do I tend to use, and how can I generalize them for a broader range of problems? How can I extract key ideas from problems I have solved and apply them to unfamiliar ones?*

Question 2 (Connecting different concepts). *Can I find patterns between problems I have solved and those I struggle with? Are there underlying principles between different algorithmic topics (e.g., dynamic programming, graph theory, number theory) that I can use to approach problems more creatively?*

Question 3 (Innovation). *How can I break away from my usual way of thinking and explore new approaches?*

Question 4 (Improving my learning process). *Which learning methods (e.g., solving harder problems, reading editorials, discussing with others) have the most impact on enhancing my creative problem-solving skills? How can I track whether my thinking process is improving?*

2 WANG & WU 2016

Resources – Tài nguyên.

1. Link: https://github.com/GrootTheDeveloper/OLP-ICPC/blob/master/Wu_Wang2016.pdf
2. Online Judge:

2.1 SECTION 1: FUNDAMENTAL PROGRAMMING SKILLS

2.1.1 Financial Management

Larry graduated this year and finally has a job. He's making a lot of money, but somehow never seems to have enough. Larry has decided that he needs to get a hold of his financial portfolio and solve his financial problems. The first step is to figure out what's been going on with his money. Larry has his bank account statements and wants to see how much money he has. Help Larry by writing a program to take his closing balance from each of the past 12 months and calculate his average account balance.

Input

The input will be 12 lines. Each line will contain the closing balance of his bank account for a particular month. Each number will be positive and displayed to the penny. No dollar sign will be included.

Output

The output will be a single number, the average (mean) of the closing balances for the 12 months. It will be rounded to the nearest penny, preceded immediately by a dollar sign, and followed by the end of the line. There will be no other spaces or characters in the output.

Example

Source: ACM Mid-Atlantic United States 2001.

IDs for online judges: POJ 1004, ZOJ 1048, UVA 2362.

Sample Input	Sample Output
100.00 489.12 12454.12 1234.10 823.05 109.20 5.27 1542.25 839.18 83.99 1295.01 1.75	\$1581.42

Analysis – Phân tích bài toán

Đề bài cho nhập 12 số thực - Dùng kiểu float/double, yêu cầu tính trung bình cộng của 12 số thực đó, làm tròn số đến 2 chữ số thập phân gần nhất (the nearest penny). Lưu ý có dấu \$ trước kết quả.

Program – Chương trình

```

1 #include <iostream>
2 #include <iomanip>
3 using namespace std;
4 int main() {
5
6     double ans = 0.0;
7     for (int i = 1; i <= 12; i++) {
8         double x; cin >> x;
9         ans += x / 12.0;
10    }
11
12    cout << "$" << fixed << setprecision(2) << ans;
13    return 0;
14 }
```

2.1.2 Doubles

As part of an arithmetic competency program, your students will be given randomly generated lists of 2–15 unique positive integers and asked to determine how many items in each list are twice some other item in the same list. You will need a program to help you with the grading. This program should be able to scan the lists and output the correct answer for each one. For example, given the list

1 4 3 2 9 7 18 22

your program should answer 3, as 2 is twice 1, 4 is twice 2, and 18 is twice 9.

Input

The input file will consist of one or more lists of numbers. There will be one list of numbers per line. Each list will contain from 2 to 15 unique positive integers. No integer will be larger than 99. Each line will be terminated with the integer 0, which is not considered part of the list. A line with the single number –1 will mark the end of the file. The example input below shows three separate lists. Some lists may not contain any doubles.

Output

The output will consist of one line per input list, containing a count of the items that are double some other item.

Example

Sample Input	Sample Output
1 4 3 2 9 7 18 22 0	3
2 4 8 10 0	2
7 5 11 13 1 3 0	0
-1	

Source: ACM Mid-Central United States 2003.

IDs for online judges: POJ 1552, ZOJ 1760, UVA 2787.

Analysis – Phân tích bài toán

Tiếp cận phần thuật toán, nhận xét rằng nếu biến đang nhập có giá trị lẻ (tức chia lấy dư cho 2 dư 1) thì không cần xét. Vì không thể lấy 2 lần một số nguyên nào mà ra kết quả có giá trị lẻ. Ngược lại, nếu biến mang giá trị chẵn thì lưu vào CTDL cây nhị phân map. Vậy khi tìm kiếm ta chỉ cần tìm nửa giá trị của biến, và đếm số lượng tìm được.

Bài toán có dạng multi-testcase, mỗi bộ test kết thúc bằng số 0, và bài toán chỉ kết thúc khi gặp số 1.

Vậy chỉ cần vòng lặp vô hạn, đặt điều kiện:

- Nếu biến đang nhập mang giá trị 0, Xử lý và xuất kết quả bài toán, sau đó reset trạng thái mọi biến/CTDL ta dùng.
- Nếu biến đang nhập mang giá trị -1, break chương trình

Program – Chương trình

```
1
2 #include <iostream>
3 #include <map>
4 #include <vector>
5
6 using namespace std;
7
8 int main() {
9     map<int, int> ma;
10    vector<int> a;
11    int n; int ans = 0;
12    while (cin >> n) {
13        if (n == -1) {
14            break;
15        }
16        if (n == 0) {
17            for (int i = 0; i < a.size(); i++) {
18                if (a[i] % 2 == 1) continue;
19                ans += ma[a[i] / 2];
20            }
21            cout << ans << endl;
22            ans = 0;
23            ma.clear();
24            a.clear();
25            continue;
26        }
27        a.push_back(n);
28        ma[n]++;
29    }
30 }
```

2.1.3 Sum of Consecutive Prime Numbers

Some positive integers can be represented by a sum of one or more consecutive prime numbers. How many such representations does a given positive integer have? For example, the integer 53 has two representations $5 + 7 + 11 + 13 + 17$ and 53. The integer 41 has three representations: $2 + 3 + 5 + 7 + 11 + 13$, $11 + 13 + 17$, and 41. The integer 3 has only one representation, which is 3. The integer 20 has no such representations. Note that summands must be consecutive prime numbers, so neither $7 + 13$ nor $3 + 5 + 5 + 7$ is a valid representation for the integer 20. Your mission is to write a program that reports the number of representations for the given positive integer.

Input

The input is a sequence of positive integers, each in a separate line. The integers are between 2 and 10,000, inclusive. The end of the input is indicated by a zero.

Output

The output should be composed of lines each corresponding to an input line, except the last zero. An output line includes the number of representations for the input integer as the sum of one or more consecutive prime numbers. No other characters should be inserted in the output.

Example

Sample Input	Sample Output
2	1
3	1
17	2
41	3
20	0
666	0
12	1
53	2
0	

Source: ACM Japan 2005.

IDs for online judges: POJ 2739, UVA 3399.

Analysis – Phân tích bài toán

Bài toán có dạng multi-testcase với điều kiện là nhập đến khi gặp giá trị 0.

Tiếp cận phần thuật toán, bài toán yêu cầu với số nguyên N (với $2 \leq N \leq 10000$), liệu có thể phân tích được bao nhiêu dạng tổng các số nguyên tố liên tiếp nhau.

Vậy, công việc đầu tiên là tạo mảng *prime* với ý nghĩa là lưu trữ các số nguyên tố trong đoạn $[2..10000]$. Ở bước này, ta thực hiện kiểm tra số nguyên tố bằng cách thử chia từ 2 đến \sqrt{i} , nếu i chia hết cho bất kỳ số nào trong khoảng này thì i không phải số nguyên tố. (Có thể tối ưu hơn với Sàng nguyên tố *Eratosthenes*, tuy nhiên không cần thiết lắm với bài này)

Tiếp theo, thay vì dùng 2 vòng for lồng nhau để xét mọi cặp chỉ số, ta sẽ sử dụng thuật toán **Two-Pointer** để đảm bảo kiểm tra hết tất cả các đoạn tổng liên tiếp một cách nhanh chóng.

Cách tiếp cận **Two-Pointer** được triển khai như sau:

- Đặt hai con trỏ *left* và *right*, bắt đầu từ đầu mảng *prime*.
- Dùng biến *sum* để lưu tổng các số nguyên tố trong đoạn $[left, right]$.
- Nếu *sum* < *N*, tăng *right* để mở rộng đoạn.
- Nếu *sum* > *N*, tăng *left* để thu hẹp đoạn.
- Nếu *sum* == *N*, tăng biến đếm *count*, sau đó tiếp tục dịch *left* để tìm các đoạn khác.

Độ phức tạp: $O(T(N^{3/2} + M))$ với $N = 10000$, T là số testcase, M là số lượng số nguyên tố trong đoạn $[2..10000]$. Trong đó $O(N^{3/2})$ là thời gian sinh số nguyên tố và $O(M)$ là thời gian kiểm tra các tổng liên tiếp.

Program – Chương trình

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5
6  vector<int> primes;
7
8  int main() {
9      for (int i = 2; i <= 10000; i++) {
10         bool check = true;
11
12         for (int j = 2; j <= sqrt((double)i); j++) {
13             if (i % j == 0) {
14                 check = false;
15                 break;
16             }
17         }
18         if (check) {
19             primes.push_back(i);
20         }
21     }
22 }
```

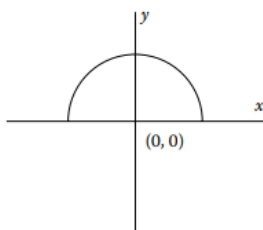
```

23     int N;
24     while (cin >> N && N != 0) {
25         int count = 0;
26         int left = 0, right = 0, sum = 0;
27
28         while (right < primes.size()) {
29             if (sum < N) {
30                 sum += primes[right++];
31             }
32             else if (sum > N) {
33                 sum -= primes[left++];
34             }
35             else {
36                 count++;
37                 sum -= primes[left++];
38             }
39         }
40
41         cout << count << endl;
42     }
43 }
44
45

```

2.1.4 I Think I Need a Houseboat

Fred Mapper is considering purchasing some land in Louisiana to build his house on. In the process of investigating the land, he learned that the state of Louisiana is actually shrinking by 50 square miles each year, due to erosion caused by the Mississippi River. Since Fred is hoping to live in this house for the rest of his life, he needs to know if his land is going to be lost to erosion. After doing more research, Fred has learned that the land that is being lost forms a semicircle. This semicircle is part of a circle centered at $(0, 0)$, with the line that bisects the circle being the X axis. Locations below the X axis are in the water. The semicircle has an area of 0 at the beginning of year 1.



Input

The first line of input will be a positive integer indicating how many data sets will be included (N). Each of the next N lines will contain the X and Y Cartesian coordinates of the land Fred is considering. These will be floating-point numbers measured in miles. The Y coordinate will be nonnegative. $(0, 0)$ will not be given.

Output

For each data set, a single line of output should appear. This line should take the form of

Property N : This property will begin eroding in year Z .

where N is the data set (counting from 1) and Z is the first year (start from 1) this property will be within the semicircle AT THE END OF YEAR Z . Z must be an integer. After the last data set, this should print out "END OF OUTPUT."

Example

Sample Input	Sample Output
2	Property 1: This property will begin eroding in year 1.
1.0 1.0	Property 1: This property will begin eroding in year 20.
25.0 0.0	END OF OUTPUT.

Source: ACM Mid-Atlantic United States 2001.

Note: No property will appear exactly on the semicircle boundary: it will be either inside or outside. This problem will be judged automatically. Your answer must match exactly, including the capitalization, punctuation, and white space. This includes the periods at the ends of the lines. All locations are given in miles.

IDs for online judges: POJ 1005, ZOJ 1049, UVA 2363.

Analysis – Phân tích bài toán

Diện tích đất bị xói mòn 50 dặm vuông mỗi năm, và xói mòn theo dạng nửa hình tròn. Ta có công thức:

$$S = \frac{1}{2}\pi r^2 \Rightarrow r = \sqrt{\frac{2S}{\pi}}$$

Với S tăng đều mỗi năm 50 dặm vuông, r cũng tăng theo, ta chỉ cần so sánh xem độ dài từ tâm O đến điểm (x, y) cần xét (bán kính $R' = \sqrt{x^2 + y^2}$) có nhỏ hơn r không, nếu nhỏ hơn, chỉ cần tăng S (số năm tăng) đến khi $R' < r$. Đến khi đó, ta xác định được số năm cần tìm.

Có thể sử dụng kỹ thuật tìm kiếm nhị phân để tìm nhanh hơn, trong đoạn $[L..R]$, ta tìm điểm giữa $mid = (l + r)/2$ với ý nghĩa mid là số năm đã bị xói mòn, từ đó ta có được S mới (diện tích nửa hình tròn sau mid year), từ S mới ta suy ra được r .

Nếu $r < R'$ có nghĩa là điểm (x, y) vẫn nằm ngoài nửa hình tròn sau mid năm. Vậy ta tiếp tục kiểm tra đoạn $[mid + 1, R]$. Nếu $r \geq R'$ có nghĩa là điểm (x, y) đã nằm trong đường tròn sau mid năm. Ta lưu kết quả $yearEnd$ tốt nhất hiện tại, và tiếp tục xét đoạn $[L..mid - 1]$ xem có năm nào sớm hơn mà vẫn thỏa mãn điểm (x, y) nằm trong nửa hình tròn không.

Program – Chương trình

```
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  const double PI = 3.1415926;
7
8  double calcR(double x, double y) {
9      return sqrt(x * x + y * y);
10 }
11
12 int main() {
13     int properties; cin >> properties;
14
15     for (int property = 1; property <= properties; property++) {
16
17         double x, y; cin >> x >> y;
18
19         int yearEnd = 1;
20         double S = 50.0;
21         double R = sqrt(2 * S / PI);
22
23         int l = 1;
24         int r = 1e9 + 7;
25
26         while (l <= r) {
27             int mid = l + r >> 1;
28             S = 50.0 * mid;
29             R = sqrt(2 * S / PI);
30             if (calcR(x, y) >= R) {
31                 l = mid + 1;
32             }
33             else {
34                 yearEnd = mid;
35                 r = mid - 1;
36             }
37         }
38
39         cout << "Property_" << property << ":_This_property_will_begin_eroding_in_year_" <<
40             yearEnd << "." << endl;
41     }
42     cout << "END_OF_OUTPUT.";
```

2.1.5 Hangover

How far can you make a stack of cards overhang a table? If you have one card, you can create a maximum overhang of half a card length. (We're assuming that the cards must be perpendicular to the table.) With two cards, you can make the top card overhang the bottom one by half a card length, and the bottom one overhang the table by a third of a card length, for a total maximum overhang of $1/2 + 1/3 = 5/6$ card lengths. In general, you can make n cards overhang by $1/2 + 1/3 + 1/4 + \dots + 1/(n+1)$ card lengths, where the top card overhangs the second by $1/2$, the second overhangs the third by $1/3$, the third overhangs the fourth by $1/4$, and so on, and the bottom card overhangs the table by $1/(n+1)$. This is illustrated in Figure 1.2.

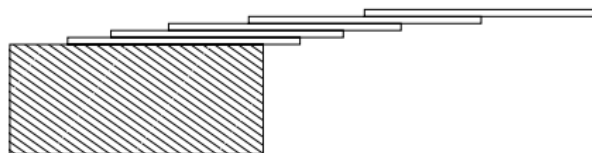


Figure 1.2 : A stack of cards overhangs a table.

Input

The input consists of one or more test cases, followed by a line containing the number 0.00 that signals the end of the input. Each test case is a single line containing a positive floating-point number c whose value is at least 0.01 and at most 5.20; c will contain exactly three digits.

Output

For each test case, output the minimum number of cards necessary to achieve an overhang of at least c card lengths. Use the exact output format shown in the examples.

Example

Sample Input	Sample Output
1.00	3 card(s)
3.71	61 card(s)
0.04	1 card(s)
5.19	273 card(s)
0.00	

Source:

IDs for online judges:

Analysis – Phân tích bài toán

Trước hết, tôi thử nháp xem rằng với bao nhiêu lá bài thì c vượt qua giá trị 5.20.

```
1 double ans = 0.00;
2 for (int i = 1; i <= 10000; i++) {
3     ans += 1 / (1.0 * (i + 1));
4     cout << ans << " ";
5     if (ans > 5.2) {
6         cout << "-" << i << "-";
7         break;
8     }
9 }
```

Kết quả cho thấy: có tối đa 276 lá bài.

Vậy nên, tôi tạo một mảng $f[277]$ với ý nghĩa $f[i]$ lưu tổng giá trị của $1/2 + 1/3 + 1/4 + \dots + 1/(i+1)$. Tiếp đến, ta cần tìm chỉ số i nhỏ nhất sao cho $c \leq f[i]$. Sau khi tìm được, kết quả bài toán là i . Vì kích thước bài toán nhỏ, ta có thể duyệt for chạy hết mảng, nếu gặp $f[i] \geq c$, ta lập tức biết được kết quả cần tìm là i .

Ở đây tôi dùng tìm kiếm nhị phân để trông có vẻ "pro" hơn.

Program – Chương trình


```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  int main() {
7
8      double n;
9      vector<double>f(277, 0.00);
10
11     for (int i = 1; i <= 276; i++) {
12         f[i] = f[i - 1] + (1.0 / (1.0*i + 1));
13     }
14
15     while (cin >> n && n != 0.00) {
16         int ans = 0;
17
18         int l = 1, r = 276;
19         while (l <= r) {
20             int mid = l + r >> 1;
21             if (f[mid] == n) {
22                 ans = mid;
23                 break;
24             }
25             if (f[mid] > n) {
26                 ans = mid;
27                 r = mid - 1;
28             }
29             else {
30                 l = mid + 1;
31             }
32         }
33
34         cout << ans << "┐card(s)" << endl;
35     }
36 }

```

2.1.6 Humidex

The humidex is a measurement used by Canadian meteorologists to reflect the combined effect of heat and humidity. It differs from the heat index used in the United States in using dew point rather than relative humidity.

When the temperature is 30°C (86°F) and the dew point is 15°C (59°F), the humidex is 34 (note that humidex is a dimensionless number, but the number indicates an approximate temperature in Celsius). If the temperature remains 30°C and the dew point rises to 25°C (77°F), the humidex rises to 42.3.

The humidex tends to be higher than the U.S. heat index at equal temperature and relative humidity.

The current formula for determining the humidex was developed by J.M. Masterton and F.A. Richardson of Canada’s Atmospheric Environment Service in 1979.

According to the Meteorological Service of Canada, a humidex of at least 40 causes “great discomfort” and above 45 is “dangerous.” When the humidex hits 54, heat stroke is imminent.

The record humidex in Canada occurred on June 20, 1953, when Windsor, Ontario, hit 52.1. (The residents of Windsor would not have known this at the time, since the humidex had yet to be invented.) More recently, the humidex reached 50 on July 14, 1995, in both Windsor and Toronto.

The humidex formula is as follows:

$$\begin{aligned}
 \text{humidex} &= \text{temperature} + h \\
 h &= (0.5555) \cdot (e - 10.0) \\
 e &= 6.11 \times \exp \left(5417.7530 \left(\frac{1}{273.16} - \frac{1}{\text{dewpoint} + 273.16} \right) \right)
 \end{aligned}$$

where $\exp(x)$ is 2.718281828 raised to the exponent x .

While humidex is just a number, radio announcers often announce it as if it were the temperature, for example, “It’s 47° out there ... with the humidex.” Sometimes weather reports give the temperature and dew point, or the temperature and humidex, but rarely do they report all three measurements. Write a program that, given any two of the measurements, will calculate the third.

You may assume that for all inputs, the temperature, dew point, and humidex are all between −100°C and 100°C.

Input

Input will consist of a number of lines. Each line except the last will consist of four items separated by spaces: a letter, a number, a second letter, and a second number. Each letter specifies the meaning of the number that follows it and will be either T, indicating temperature; D, indicating dew point; or H, indicating humidex. The last line of input will consist of the single letter E

Output

For each line of input except the last, produce one line of output. Each line of output should have the form:

T number D number H number

where the three numbers are replaced with the temperature, dew point, and humidex. Each value should be expressed rounded to the nearest tenth of a degree, with exactly one digit after the decimal point. All temperatures are in degrees Celsius

Example

Sample Input	Sample Output
T 30 D 15	T 30.0 D 15.0 H 34.0
T 30.0 D 25.0	T 30.0 D 25.0 H 42.3
E	

Source: Waterloo Local Contest, July 14, 2007

IDs for online judges: POJ 3299.

Analysis – Phân tích bài toán

Program – Chương trình

2.1.7 Sum

Your task is to find the sum of all integer numbers lying between 1 and N inclusive.

Input

The input consists of a single integer N that is not greater than 10,000 by its absolute value.

Output

Write a single integer number that is the sum of all integer numbers lying between 1 and N inclusive.

Example

Sample Input	Sample Output
-3	-5

Source: ACM 2000, Northeastern European Regional Programming Contest (test tour).

IDs for online judges: Ural 1068.

Hint: Based on the summation formula of arithmetic progression $s = 1 + 2 + \dots + N$, if N is an integer larger than 0, then $s = [(1 + N)/2]*N$; otherwise, $s = [(1 - N)/2]*N + 1$.

Analysis – Phân tích bài toán

Công thức tính tổng dãy số cách đều 1, 2, 3, ... , N là $S = N * (N + 1) / 2$.

Nếu $N \leq 0$, ta cần tính S từ 1 đến $-N$, sau đó kết quả sẽ là $-S + 1$.

Nếu $N > 0$, ta đơn giản là tính S dựa theo công thức thôi.

Program – Chương trình

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5
6     int n; cin >> n;
7     if (n <= 0) {
8         n = -n;
9         cout << -n * (n + 1) / 2 + 1;
10    }
11    else {
12        cout << n * (n + 1) / 2;
13    }
14 }
```

2.1.8 Specialized Four-Digit Numbers

Find and list all four-digit numbers in decimal notation that have the property that the sum of their four digits equals the sum of their digits when represented in hexadecimal (base 16) notation and also equals the sum of their digits when represented in duodecimal (base 12) notation.

For example, the number 2991 has the sum of (decimal) digits $2 + 9 + 9 + 1 = 21$. Since $2991 = 1*1728 + 8*144 + 9*12 + 3$, its duodecimal representation is 189312, and these digits also sum up to 21. But in hexadecimal, 2991 is BAF16, and $11 + 10 + 15 = 36$, so 2991 should be rejected by your program.

The next number (2992), however, has digits that sum to 22 in all three representations (including BB016), so 2992 should be on the listed output. (We don't want decimal numbers with fewer than four digits—excluding leading zeros—so that 2992 is the first correct answer.)

Input

There is no input for this problem.

Output

Your output is to be 2992 and all larger four-digit numbers that satisfy the requirements (in strictly increasing order), each on a separate line, with no leading or trailing blanks, ending with a new-line character. There are to be no blank lines in the output. The first few lines of the output are shown below:

Example

Sample Input	Sample Output
	2992
	2993
	2994
	2995
	2996
	...

Source: ACM Pacific Northwest 2004.

IDs for online judges: POJ 2196, ZOJ 2405, UVA 3199.

Analysis – Phân tích bài toán

Với hệ cơ số N bất kì, ta có thể tính tổng các chữ số bằng cách:

```
1
2 int calc(int n, int base) {
3     int ans = 0;
4     while (n) {
5         ans += n % base;
6         n /= base;
7     }
8     return ans;
9 }
```

Trong đó, $base$ là hệ cơ số, n là số nguyên cần tính.

Program – Chương trình

```
1
2 #include <iostream>
3 using namespace std;
4
5 int calc(int n, int base) {
6     int ans = 0;
7     while (n) {
8         ans += n % base;
9         n /= base;
10    }
11    return ans;
12 }
13
14 int main() {
15
16     for (int i = 2991; i <= 9999; i++) {
17         if (calc(i, 10) == calc(i, 16) && calc(i, 10) == calc(i, 12)) {
18             cout << i << endl;
19         }
20     }
21 }
```

2.1.9 Quicksum

A checksum is an algorithm that scans a packet of data and returns a single number. The idea is that if the packet is changed, the checksum will also change, so checksums are often used for detecting transmission errors, validating document contents, and in many other situations where it is necessary to detect undesirable changes in data.

For this problem, you will implement a checksum algorithm called quicksum. A quicksum packet allows only uppercase letters and spaces. It always begins and ends with an uppercase letter. Otherwise, spaces and letters can occur in any combination, including consecutive spaces.

A quicksum is the sum of the products of each character's position in the packet times the character's value. A space has a value of zero, while letters have a value equal to their position in the alphabet. So, A = 1, B = 2, and so on, through Z = 26. Here are example quicksum calculations for the packets "ACM" and "MID CENTRAL":

ACM: $1*1 + 2*3 + 3*13 = 46$

MID CENTRAL: $1*13 + 2*9 + 3*4 + 4*0 + 5*3 + 6*5 + 7*14 + 8*20 + 9*18 + 10*1 + 11*12 = 650$

Input

The input consists of one or more packets followed by a line containing only # that signals the end of the input. Each packet is on a line by itself, does not begin or end with a space, and contains from 1 to 255 characters. .

Output

For each packet, output its quicksum on a separate line in the output.

Sample Input	Sample Output
ACM	46
MID CENTRAL	650
REGIONAL PROGRAMMING CONTEST	4690
CONTEST	49
ACN	75
A C M	14
BBC	15
#	

Example

Source: ACM Mid-Central United States 2006.

IDs for online judges: POJ 3094, ZOJ 2812, UVA 3594.

Analysis – Phân tích bài toán

Với các ký tự ['A'..'Z'] ta chuyển về [1..26]. Để thực hiện, ta cần xét kí tự c thuộc đoạn ['A'..'Z'], sau đó ta trừ 'A' (mã ASCII) cho c để dịch về [0..25] và cộng 1 để trở thành đoạn [1..26].

Sample Input trong Example bị thiếu:

Gốc: REGIONAL PROGRAMMING = 4690 (thực tế là = 2240).

Sửa lại: REGIONAL PROGRAMMING CONTEST = 4960 (chuẩn với ví dụ).

Program – Chương trình

```

1
2 #include <iostream>
3 #include <string>
4 using namespace std;
5
6 int main() {
7
8     string s;
9     while (getline(cin, s) && s != "#") {
10         int ans = 0;
11         for (int i = 0; i < s.size(); i++) {
12             if (s[i] == '\n') continue;
13             ans += (i + 1) * (s[i] - 'A' + 1);
14         }
15         cout << ans << endl;
16     }
17 }
```

2.1.10 A Contesting Decision

Judging a programming contest is hard work, with demanding contestants, tedious decisions, and monotonous work—not to mention the nutritional problems of spending 12 hours with only donuts, pizza, and soda for food. Still, it can be a lot of fun.

Software that automates the judging process is a great help, but the notorious unreliability of some contest software makes people wish that something better were available. You are part of a group trying to develop better, open-source, contest management software, based on the principle of modular design.

Your component is to be used for calculating the scores of programming contest teams and determining a winner. You will be given the results from several teams and must determine the winner.

Scoring

There are two components to a team's score. The first is the number of problems solved. The second is penalty points, which reflect the amount of time and incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved.

So if a team solved problem 1 on their second submission at 20 minutes, they are charged 40 penalty points. If they submit problem 2 three times, but do not solve it, they are charged no penalty points. If they submit problem 3 once and solve it at 120 minutes, they are charged 120 penalty points. Their total score is two problems solved with 160 penalty points.

The winner is the team that solves the most problems. If teams tie for solving the most problems, then the winner is the team with the fewest penalty points.

Input

For the programming contest your program is judging, there are four problems. You are guaranteed that the input will not result in a tie between teams after counting penalty points.

Line 1: $\langle nTeams \rangle$

Line 2: $n+1 \langle Name \rangle \langle p1Sub \rangle \langle p1Time \rangle \langle p2Sub \rangle \langle p2Time \rangle \dots \langle p4Time \rangle$

The first element on the line is the team name, which contains no white space. Following that, for each of the four problems, is the number of times the team submitted a run for that problem and the time at which it was solved correctly (both integers). If a team did not solve a problem, the time will be zero. The number of submissions will be at least one if the problem was solved.

Output

The output consists of a single line listing the name of the team that won, the number of problems they solved, and their penalty points.

Example

Sample Input	Sample Output
4 Stars 2 20 5 0 4 190 3 220 Rockets 5 180 1 0 2 0 3 100 Penguins 1 15 3 120 1 300 4 0 Marsupials 9 0 3 100 2 220 3 80	Penguins 3 475

Source: ACM Mid-Atlantic 2003.

IDs for online judges: POJ 1581, ZOJ 1764, UVA 2832.

Analysis – Phân tích bài toán

Mấu chốt quyết định đầu tiên là số bài giải được, có nghĩa là $Time > 0$ là nhiều nhất. Sau đó mới xét đến *Penalty*.

Bài toán chỉ để kiểm tra khả năng quản lý biến chứ không có gì khó khăn.

Program – Chương trình

```
1
2 #include <iostream>
3 #include <string>
4 #include <vector>
5 using namespace std;
6
7
8 struct problem {
9     int sub, time;
10 };
11 struct team {
12     string name;
13     problem prob[4];
14 };
15 int main() {
16
17     int n; cin >> n;
18
19     vector<team>a(n);
20     for (int i = 0; i < n; i++) {
21         cin >> a[i].name;
22         for (int j = 0; j < 4; j++) {
23             cin >> a[i].prob[j].sub >> a[i].prob[j].time;
24         }
25     }
```

```

25     }
26
27
28     int maxSolve = 0;
29     int maxPenalty = 0;
30     string maxName;
31
32
33     for (int i = 0; i < n; i++) {
34         int currentPenalty = 0;
35         int currentSolve = 0;
36         string currentName = a[i].name;
37         for (int j = 0; j < 4; j++) {
38             currentPenalty += (a[i].prob[j].time > 0 ? a[i].prob[j].time + (20 * (a[i].prob[j].
39                 sub <= 1 ? 0 : a[i].prob[j].sub - 1)) : 0);
40             currentSolve += (a[i].prob[j].time > 0 ? 1 : 0);
41         }
42
43         if (currentSolve > maxSolve) {
44             maxSolve = currentSolve;
45             maxPenalty = currentPenalty;
46             maxName = currentName;
47         }
48         else if (currentSolve == maxSolve) {
49             if (currentPenalty < maxPenalty) {
50                 maxSolve = currentSolve;
51                 maxPenalty = currentPenalty;
52                 maxName = currentName;
53             }
54         }
55     }
56     cout << maxName << "␣" << maxSolve << "␣" << maxPenalty;
57 }

```

2.1.11 Dirichlet's Theorem on Arithmetic Progressions

If a and d are relatively prime positive integers, the arithmetic sequence beginning with a and increasing by d , that is, $a, a + d, a + 2d, a + 3d, a + 4d, \dots$, contains infinitely many prime numbers. This fact is known as Dirichlet's theorem on arithmetic progressions, which had been conjectured by Johann Carl Friedrich Gauss (1777–1855) and was proved by Johann Peter Gustav Lejeune Dirichlet (1805–1859) in 1837.

For example, the arithmetic sequence beginning with 2 and increasing by 3, that is,

2, 5, 8, 11, 14, 17, 20, 23, 26, 29, 32, 35, 38, 41, 44, 47, 50, 53, 56, 59, 62, 65, 68, 71, 74, 77, 80, 83, 86, 89, 92, 95, 98, \dots

contains infinitely many prime numbers:

2, 5, 11, 17, 23, 29, 41, 47, 53, 59, 71, 83, 89, \dots

Your mission, should you choose to accept it, is to write a program to find the n th prime number in this arithmetic sequence for given positive integers a, d , and n .

Input

The input is a sequence of data sets. A data set is a line containing three positive integers a, d , and n separated by a space. a and d are relatively prime. You may assume $a \leq 9307$, $d \leq 346$, and $n \leq 210$.

The end of the input is indicated by a line containing three zeros separated by a space. It is not a data set.

Output

The output should be composed of as many lines as the number of the input data sets. Each line should contain a single integer and should never contain extra characters.

The output integer corresponding to a data set a, d, n should be the n th prime number among those contained in the arithmetic sequence beginning with a and increasing by d .

For your information, it is known that the result is always less than 106 (1 million) under this input condition.

Example

Source: ACM Japan 2006, Domestic.

IDs for online judges: POJ 3006.

Sample Input	Sample Output
367 186 151	92809
179 10 203	6709
271 37 39	12037
103 230 1	103
27 104 185	93523
253 50 85	14503
1 1 1	2
9075 337 210	899429
307 24 79	5107
331 221 177	412717
259 170 40	22699
269 58 102	25673
0 0 0	

Analysis – Phân tích bài toán

Với $a \leq 9307, d \leq 346, n \leq 210$, số nguyên tố lớn nhất có thể xuất hiện trong giới hạn này là: 469487.

Kiểm tra bằng đoạn code:

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  bool check(int n) {
5      if (n == 2 || n == 3) return true;
6      for (int i = 2; i * i <= n; i++) {
7          if (n % i == 0) return false;
8      }
9      return true;
10 }
11
12 int main(){
13     int a = 9307;
14     int d = 346;
15     int cnt = 0;
16     int i = 0;
17     for (; cnt < 210;) {
18         if (check(a + i * d)) {
19             cout << a + i * d << " ";
20             cnt++;
21         }
22         i++;
23     }
24 }
```

đễ dàng thấy được đúng là vậy. Vậy để tối ưu tốc độ, ta dùng sàng nguyên tố Eratosthenes với giới hạn là $469487 + 1$ (0-index-based).

Việc còn lại chỉ cần kiểm tra số nguyên tố thứ n trong dãy số được cho.

Phía trên là những nhận định sai lầm nhất cuộc đời tôi

Sau khi nộp lần đầu và bị sai, tôi thử tăng giới hạn số nguyên tố lên 1.000.000 thì đã AC.

Program – Chương trình

```

1
2  #include <iostream>
3  #include <vector>
4
5  using namespace std;
6
7  int maxPrimeNumber = 1000000;
8  vector<bool> prime(maxPrimeNumber + 1, true);
9
10 void sieve() {
11     prime[0] = prime[1] = false;
12     for (int i = 2; i * i <= maxPrimeNumber; i++) {
13         if (prime[i]) {
14             for (int j = i * i; j <= maxPrimeNumber; j += i) {
15                 prime[j] = false;
16             }
17         }
18     }
19 }
```



```

16         }
17     }
18 }
19 }
20
21 int main(){
22     sieve();
23     int a, d, n;
24     while (cin >> a && a != 0 && cin >> d >> n) {
25         int number = 0;
26         int index = 0;
27         int result = 0;
28         while (number < n) {
29             if (prime[a + index * d] == true) {
30                 number++;
31                 result = a + index * d;
32             }
33             index++;
34         }
35
36         cout << result << endl;
37     }
38 }

```

2.1.12 The Circumference of the Circle

To calculate the circumference of a circle seems to be an easy task—provided you know its diameter. But what if you don't? You are given the Cartesian coordinates of three noncollinear points in the plane. Your job is to calculate the circumference of the unique circle that intersects all three points.

Input

The input file will contain one or more test cases. Each test case consists of one line containing six real numbers, $x_1, y_1, x_2, y_2, x_3, y_3$, representing the coordinates of the three points. The diameter of the circle determined by the three points will never exceed 1 million. Input is terminated by the end of the file.

Output

For each test case, print one line containing one real number telling the circumference of the circle determined by the three points. The circumference is to be printed accurately rounded to two decimals. The value of π is approximately 3.141592653589793.

Example

Sample Input	Sample Output
0.0 -0.5 0.5 0.0 0.0 0.5	3.14
0.0 0.0 0.0 1.0 1.0 1.0	4.44
5.0 5.0 5.0 7.0 4.0 6.0	6.28
0.0 0.0 -1.0 7.0 7.0 7.0	31.42
50.0 50.0 50.0 70.0 40.0 60.0	62.83
0.0 0.0 10.0 0.0 20.0 1.0	632.24
0.0 -500000.0 500000.0 0.0 0.0 500000.0	3141592.65

Source: : Ulm Local Contest 1996.

IDs for online judges: POJ 2242, ZOJ 1090.

Analysis – Phân tích bài toán

Chu vi của đường tròn: $C = 2\pi R$

Trong đó:

- R là bán kính đường tròn
- $\pi = 3.141592653589793$

Ta cần tìm bán kính của đường tròn ngoại tiếp tam giác 3 điểm $(x_1, y_1), (x_2, y_2), (x_3, y_3)$. Công thức tính diện tích của đường tròn ngoại tiếp tam giác là:

$$A = \frac{1}{2}|x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)|.$$

Sau đó, tìm bán kính của đường tròn ngoại tiếp bằng công thức:

$$R = \frac{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \cdot \sqrt{(x_3-x_1)^2+(y_3-y_1)^2} \cdot \sqrt{(x_2-x_3)^2+(y_2-y_3)^2}}{4A}$$

Cuối cùng là tìm chu vi của đường tròn.

Program – Chương trình

```

1
2 #include <iostream>
3 #include <cmath>
4 #include <iomanip>
5 using namespace std;
6
7 struct point {
8     double x, y;
9 };
10
11 double pw(double x) {
12     return x * x;
13 }
14
15 double triangle(point p1, point p2, point p3) {
16     return 0.5 * abs(p1.x * (p2.y - p3.y) + p2.x * (p3.y - p1.y) + p3.x * (p1.y - p2.y));
17 }
18
19 double R(point p1, point p2, point p3) {
20     double ans = (sqrt(pw(p1.x - p2.x) + pw(p1.y - p2.y)) * sqrt(pw(p1.x - p3.x) + pw(p1.y - p3.y)) *
21                 sqrt(pw(p2.x - p3.x) + pw(p2.y - p3.y)));
22     ans /= 4.0 * triangle(p1, p2, p3);
23     return ans;
24 }
25
26 const double pi = 3.141592653589793;
27
28 int main(){
29     point p1, p2, p3;
30     while (cin >> p1.x >> p1.y >> p2.x >> p2.y >> p3.x >> p3.y) {
31         double r = R(p1, p2, p3);
32         double c = 2 * pi * r;
33         cout << fixed << setprecision(2) << c << endl;
34     }
35 }
```

2.1.13 Vertical Histogram

Write a program to read four lines of uppercase (i.e., all CAPITAL LETTERS) text input (no more than 72 characters per line) from the input file and print a vertical histogram that shows how many times each letter (but not blanks, digits, or punctuation) appears in the all-uppercase input. Format your output exactly as shown.

Input

Lines 1–4: Four lines of uppercase text, no more than 72 characters per line.

Output

Lines 1–?: Several lines with asterisks and spaces followed by one line with the uppercase alphabet separated by spaces. Do not print unneeded blanks at the end of any line. Do not print any leading blank lines.

Example

Sample Input	Sample Output
THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG. THIS IS AN EXAMPLE TO TEST FOR YOUR HISTOGRAM PROGRAM. HELLO!	<pre> * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * A B C D E F G H I J K L M N O P Q R S T U V W X Y Z </pre>

Source: USACO, February 2003, Orange.
IDs for online judges: POJ 2136.

Analysis – Phân tích bài toán

Nhìn vào **Example**, ta thấy được tần suất ký tự xuất hiện ở các dòng tăng dần về dưới. Vậy cần *Priority Queue* quản lý tần suất các ký tự xuất hiện.

Thuật toán như sau: với các ký tự có lần xuất hiện cao nhất hiện tại, ta cho chúng nằm cùng 1 hàng. Sau đó đưa nó vào lại Priority Queue với lần xuất hiện hiện tại trừ đi 1, tất nhiên là nếu lần xuất hiện hiện tại > 1.

Program – Chương trình

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main(){
5      string s[4];
6      for (int i = 0; i < 4; i++) {
7          cin >> s[i];
8      }
9      vector<int>character(26, 0);
10     priority_queue<pair<int, char>> pq;
11
12     for (int i = 0; i < 4; i++) {
13         for (auto c : s[i]) {
14             character[c - 'A']++;
15         }
16     }
17     int maxSize = *max_element(character.begin(), character.end());
18
19     vector<vector<char>> show(maxSize, vector<char>(26, '␣'));
20
21     for (int i = 0; i < 26; i++) {
22         if (character[i] == 0) continue;
23         pq.push({ character[i], i});
24     }
25     int idx = 0;
26
27     while (pq.empty() == false) {
28         auto [Len, C] = pq.top();
29         pq.pop();
30         show[idx][C] = '*';
31         while (pq.empty() == false && pq.top().first == Len) {
32             auto [sLen, sC] = pq.top();
33             show[idx][sC] = '*';
34             pq.pop();
35             if (sLen == 1) continue;
36             pq.push({ sLen - 1, sC });
37         }
38         idx++;

```

```

39         if (Len == 1) continue;
40         pq.push({ Len - 1, C });
41     }
42
43
44     for (int i = 0; i < maxSize; i++) {
45         for (int j = 0; j < 26; j++) {
46             cout << show[i][j] << "  ";
47         }
48         cout << endl << endl;
49     }
50     for (char C = 'A'; C <= 'Z'; C++) {
51         cout << C << "  ";
52     }
53 }

```

2.1.14 Ugly Numbers

Ugly numbers are numbers whose only prime factors are 2, 3, or 5. The sequence 1, 2, 3, 4, 5, 6, 8, 9, 10, 12, ... shows the first 10 ugly numbers. By convention, 1 is included.

Given the integer n , write a program to find and print the n th ugly number.

Input

Each line of the input contains a positive integer n ($n \leq 1500$). Input is terminated by a line with $n = 0$.

Output

For each line, output the n th ugly number. Don't deal with the line with $n = 0$.

Example

Sample Input	Sample Output
1	1
2	2
9	10
0	

Source: New Zealand 1990, Division I.

IDs for online judges: POJ 1338, UVA 136.

Analysis – Phân tích bài toán

Số xấu quắc là số khi phân tích thừa số nguyên tố, các thừa số nguyên tố chỉ thuộc $[2, 3, 5]$. Vậy chỉ cần chia nó cho 2, 3, 5 (nếu chia được), sau khi chia xong, nếu số $= 1$ thì đương nhiên nó được tạo chỉ từ các ước số $[2, 3, 5]$.

Số 1 là ngoại lệ.

Program – Chương trình

```

1
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5
6 int main(){
7     int n;
8
9     vector<int> uglyNumber;
10    int i;
11    for (i = 1; uglyNumber.size() < 1500; i++) {
12        int number = i;
13        while (number % 2 == 0) {
14            number /= 2;
15        }
16        while (number % 3 == 0) {
17            number /= 3;
18        }
19

```

```

20         while (number % 5 == 0) {
21             number /= 5;
22         }
23         if (number == 1) uglyNumber.push_back(i);
24     }
25     while (cin >> n && n != 0) {
26         n--;
27         cout << uglyNumber[n] << endl;
28     }
29 }

```

2.1.15 Number Sequence

A single positive integer i is given. Write a program to find the digit located in the position i in the sequence of number groups S_1, S_2, \dots, S_k . Each group S_k consists of a sequence of positive integer numbers ranging from 1 to k , written one after another. For example, the first 80 digits of the sequence are as follows:
11212312341234512345612345671234567812345678910123456789101112345678910

Input

The first line of the input file contains a single integer t ($1 \leq t \leq 10$), the number of test cases, followed by one line for each test case. The line for a test case contains the single integer i ($1 \leq i \leq 2147483647$).

Output

There should be one output line per test case containing the digit located in the position i .

Example

Sample Input	Sample Output
2	2
8	2
3	

Source: ACM Tehran 2002, First Iran Nationwide Internet Programming Contest.
IDs for online judges: POJ 1019, ZOJ 1410

Analysis – Phân tích bài toán

Thuật toán ngây thơ là cho 1 chuỗi S và T ban đầu rỗng, duyệt $idx = 1$ tăng dần, cộng dồn idx hiện tại vào chuỗi T (tất nhiên là idx đã chuyển sang dạng chuỗi), sau đó cộng dồn T vào S , lặp đến khi nào độ dài chuỗi lớn hơn vị trí i cần tìm. Kết quả là: $S[n-1]$

Có nghĩa là:

```

1     int T;
2     cin >> T;
3     while (T--) {
4         int n; cin >> n;
5         string s, t;
6         for (int i = 1; ; i++) {
7             if (s.size() + to_string(i).size() > n) {
8                 break;
9             }
10            t += to_string(i);
11            s += t;
12        }
13        cout << s[n - 1] << endl;
14    }

```

Tôi gọi đây là thuật toán ngây thơ vì giới hạn bài toán lên đến $2^{63} - 1$ quá lớn, chuỗi trong C++ không thể chứa nổi.

Thuật toán chuẩn thì chịu.

Program – Chương trình

2.1.16 Đánh Giá Chủ Quan Chap 1 - Section 1

Toán ơi là toán, toàn là toán. Chap này có lẽ là món khoái khẩu của thầy Nguyễn Quân Bá Hồng đây.

2.1.17 Speed Limit

Bill and Ted are taking a road trip. But the odometer in their car is broken, so they don't know how many miles they have driven. Fortunately, Bill has a working stopwatch, so they can record their speed and the total time they have driven. Unfortunately, their record-keeping strategy is a little odd, so they need help computing the total distance driven. You are to write a program to do this computation. For example, if their log shows

Speed in Miles per Hour	Total Elapsed Time in Hours
20	2
30	6
10	7

this means they drove 2 hours at 20 miles per hour, then $6 - 2 = 4$ hours at 30 miles per hour, then $7 - 6 = 1$ hour at 10 miles per hour. The distance driven is then $(2)(20) + (4)(30) + (1)(10) = 40 + 120 + 10 = 170$ miles. Note that the total elapsed time is always since the beginning of the trip, not since the previous entry in their log.

Input

The input consists of one or more data sets. Each set starts with a line containing an integer n , $1 \leq n \leq 10$, followed by n pairs of values, one pair per line. The first value in a pair, s , is the speed in miles per hour, and the second value, t , is the total elapsed time. Both s and t are integers, $1 \leq s \leq 90$ and $1 \leq t \leq 12$. The values for t are always in strictly increasing order. A value of -1 for n signals the end of the input.

Output

For each input set, print the distance driven, followed by a space, followed by the word miles.

Example

Sample Input	Sample Output
3	170 miles
20 2	180 miles
30 6	90 miles
10 7	
2	
60 1	
30 5	
4	
15 1	
25 2	
30 3	
10 5	
-1	

Source: ACM Mid-Central United States 2004.

IDs for online judges: POJ 2177, ZOJ 2176, UVA 3059.

Analysis – Phân tích bài toán

Làm theo y chang đề bài mô tả luôn, không biết phân tích gì....

Program – Chương trình

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 int main() {
```

```

6
7     int n;
8     while (cin >> n && n != -1) {
9         vector<pair<int, int>> a(n);
10        for (int i = 0; i < n; i++) {
11            int speed, time;
12            cin >> speed >> time;
13            a[i].first = speed;
14            a[i].second = time;
15        }
16        int distance = a[0].first * a[0].second;
17        for (int i = 1; i < n; i++) {
18            distance += a[i].first * (a[i].second - a[i - 1].second);
19        }
20        cout << distance << " miles" << endl;
21    }
22 }

```

2.1.18 Ride to School

Many graduate students of Peking University are living on Wanliu Campus, which is 4.5 kilometers from the main campus—Yanyuan. Students in Wanliu have to either take a bus or ride a bike to go to school. Due to the bad traffic in Beijing, many students choose to ride a bike.

We may assume that all the students except “Charley” ride from Wanliu to Yanyuan at a fixed speed. Charley is a student with a different riding habit—he always tries to follow another rider to avoid riding alone. When Charley gets to the gate of Wanliu, he will look for someone who is setting off to Yanyuan. If he finds someone, he will follow that rider, or if not, he will wait for someone to follow. On the way from Wanliu to Yanyuan, at any time if a faster student surpasses Charley, he will leave the rider he is following and speed up to follow the faster one.

We assume the time that Charley gets to the gate of Wanliu is zero. Given the set-off time and speed of the other students, your task is to give the time when Charley arrives at Yanyuan.

Input

There are several test cases. The first line of each case is N ($1 \leq N \leq 10,000$) representing the number of riders (excluding Charley). $N = 0$ ends the input. The following N lines are information of N different riders, in such format:

$$V_i[TAB]T_i$$

V_i is a positive integer ≤ 40 , indicating the speed of the i th rider (kilometers per hour). T_i is the set-off time of the i th rider, which is an integer and counted in seconds. In any case, it is ensured that there always exists a nonnegative T_i .

Output

The output is one line for each case: the arrival time of Charley. Round up (ceiling) the value when dealing with a fraction.

Example

Sample Input	Sample Output
4	780
20 0	771
25 -155	
27 190	
30 240	
2	
21 0	
22 34	
0	

Source: ACM Beijing 2004, Preliminary.

IDs for online judges: POJ 1922, ZOJ 2229.

Analysis – Phân tích bài toán

Vì Charley chỉ bám theo người vượt hẳn, nên với những người xuất phát trước thì không cần xét.

Vì Charley chỉ bám theo, nên ta chỉ cần xét người tới đích đầu tiên mà không cần xét bám theo ai và ai vượt lên cho phức tạp.

Program – Chương trình

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4
5 struct driver {
6     double v;
7     double t;
8 };
9
10 int main(){
11     int n;
12     const double distance = 4500;
13     while(cin >> n && n != 0) {
14         vector<driver> a(n);
15         for (int i = 0; i < n; i++) {
16             cin >> a[i].v >> a[i].t;
17             a[i].v = a[i].v * 1000 / 3600;
18         }
19         double min_dist = 1e9 + 7;
20         for (int i = 0; i < n; i++) {
21             if (a[i].t >= 0) {
22                 min_dist = min(min_dist, distance / a[i].v + a[i].t);
23             }
24         }
25         int ans = (int)min_dist + (min_dist == (int)min_dist ? 0 : 1);
26         cout << ans << endl;
27     }
28 }
```

2.1.19 Self-Numbers

In 1949, the Indian mathematician D.R. Kaprekar discovered a class of numbers called self- numbers. For any positive integer n , define $d(n)$ to be n plus the sum of the digits of n . (The d stands for digit addition, a term coined by Kaprekar.) For example, $d(75) = 75 + 7 + 5 = 87$. Given any positive integer n as a starting point, you can construct the infinite increasing sequence of integers $n, d(n), d(d(n)), d(d(d(n))), \dots$. For example, if you start with 33, the next number is $33 + 3 + 3 = 39$, the next is $39 + 3 + 9 = 51$, the next is $51 + 5 + 1 = 57$, and so on, and you generate the sequence

33, 39, 51, 57, 69, 84, 96, 111, 114, 120, 123, 129, 141, ...

The number n is called a generator of $d(n)$. In the sequence above, 33 is a generator of 39, 39 is a generator of 51, 51 is a generator of 57, and so on. Some numbers have more than one generator; for example, 101 has two generators, 91 and 100. A number with no generators is a self-number. There are 13 self-numbers less than 100: 1, 3, 5, 7, 9, 20, 31, 42, 53, 64, 75, 86, and 97.

Input

There is no input for this problem.

Output

Write a program to output all positive self-numbers less than 10,000 in increasing order, one per line.

Example

Source: ACM Mid-Central United States 1998.

IDs for online judges: POJ 1316, ZOJ 1180, UVA 640.

Analysis – Phân tích bài toán

Ý tưởng tương tự như Sàng nguyên tố, đầu tiên ta cho tất cả các số $[1..10000]$ đều là self-number. Sau đó duyệt từ $i = 1$ đến 10000, nếu số hiện tại là self-number, thì $d(n)$ của nó không phải là self-number, và mọi $d(d(n))$, $d(d(d(n)))$... đều không phải là self-number.

Program – Chương trình

Sample Input	Sample Output
	1
	3
	5
	7
	9
	20
	31
	42
	53
	64
	a lot more numbers
	9903
	9914
	9925
	9927
	9938
	9949
	9960
	9971
	9982
	9993

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  const int limit = 10000;
6
7  int d(int n) {
8      int digit = 0;
9      int number = n;
10     while(number != 0) {
11         digit += number % 10;
12         number /= 10;
13     }
14     return n + digit;
15 }
16
17 int main(){
18     vector<bool>selfNumber(limit + 1, true);
19     for (int i = 1; i <= limit; i++) {
20         int num = i;
21         if (selfNumber[num] == true) {
22             while(d(num) <= limit) {
23                 selfNumber[d(num)] = false;
24                 num = d(num);
25             }
26         }
27     }
28     for (int i = 1; i <= limit; i++) {
29         if (selfNumber[i] == true) {
30             cout << i << endl;
31         }
32     }
33 }

```

2.1.20 Bee

In Africa, there is a very special species of bee. Every year, the female bees of the species give birth to one male bee, while the male bees give birth to one male bee and one female bee, and then they die.

Now scientists have accidentally found one “magical” female bee of the special species to the effect that she is immortal, but still able to give birth once a year as all the other female bees. The scientists would like to know how many bees there will be after N years. Write a program that helps them find the number of male bees and the total number of all bees after N years.

Input

Each line of input contains an integer N ($N \geq 0$). Input ends with a case where $N = -1$. (This case should not be processed.)

Output

Each line of output should have two numbers, the first one the number of male bees after N years and the second one the total number of bees after N years. (The two numbers will not exceed 2^{32} .)

Example

Sample Input	Sample Output
1	1 2
3	4 7
-1	

Source:

IDs for online judges: UVA 11000.

Analysis – Phân tích bài toán

Ban đầu, có 1 con ong cái "ma thuật", 0 ong đực, 0 ong cái.

1 ong cái sinh ra 1 ong đực, 1 ong đực sinh ra 1 ong cái và 1 ong đực. Sau khi sinh thì chết, trừ ong ma thuật.

Ta gọi $T(n)$ là số ong sinh ra sau n năm.

$F(n)$ là số ong cái sinh ra sau n năm.

$M(n)$ là số ong đực sinh ra sau n năm.

Ta có công thức truy hồi như sau:

$$F(n) = M(n-1)$$

$$M(n) = F(n-1) + M(n-1) + \text{magicBee}$$

$$T(n) = F(n) + M(n)$$

Với $M(0) = 0, T(0) = 0, \text{magic bee} = 1$

Lưu ý, dùng kiểu dữ liệu long long để phù hợp với bài này.

Program – Chương trình

```
1 #include <iostream>
2 #define int long long
3 using namespace std;
4
5 const int magicBee = 1;
6
7 signed main(){
8
9     int year;
10    while(cin >> year && year != -1) {
11        int femaleBee = 0;
12        int maleBee = 0;
13        for (int y = 1; y <= year; y++) {
14            int newFemaleBee = maleBee;
15            int newMaleBee = maleBee + femaleBee + magicBee;
16            maleBee = newMaleBee;
17            femaleBee = newFemaleBee;
18        }
19        cout << maleBee << " " << maleBee + femaleBee + magicBee << endl;
20    }
21 }
```

2.1.21 Gold Coins

The king pays his loyal knight in gold coins. On the first day of his service, the knight receives one gold coin. On each of the next two days (the second and third days of service), the knight receives two gold coins. On each of the next three days (the fourth, fifth, and sixth days of service), the knight receives three gold coins. On each of the next four days (the seventh, eighth, ninth, and tenth days of service), the knight receives four gold coins. This pattern of payments will continue indefinitely: after receiving N gold coins on each of N consecutive days, the knight will receive $N + 1$ gold coins on each of the next $N + 1$ consecutive days, where N is any positive integer.

Your program will determine the total number of gold coins paid to the knight in any given number of days (starting from day 1).

Input

The input contains at least 1, but no more than 21 lines. Each line of the input file (except the last one) contains data for one test case of the problem, consisting of exactly one integer (in the range 1 .. 10000), representing the number of days. The end of the input is signaled by a line containing the number 0.

Output

There is exactly one line of output for each test case. This line contains the number of days from the corresponding line of input, followed by one blank space and the total number of gold coins paid to the knight in the given number of days, starting with day 1.

Example

Sample Input	Sample Output
10	10 30
6	6 14
7	7 18
11	11 35
15	15 55
16	16 61
100	100 945
10000	10000 942820
1000	1000 29820
21	21 91
22	22 98
0	

Source: ACM Rocky Mountain 2004.

IDs for online judges: POJ 2000, ZOJ 2345, UVA 3045.

Analysis – Phân tích bài toán

Ta có thể thấy số ngày được chia ra từng block với ý nghĩa block i là gồm i ngày liên tiếp, và mỗi ngày trong block i được nhận i đồng tiền.

Vậy cứ trừ dần từng block đến khi nó bằng 0, hoặc còn dư thì cộng số ngày còn lại trong block nhân với số tiền tương ứng trong block đang xét.

Program – Chương trình

```
1 #include <iostream>
2 using namespace std;
3
4 int main(){
5     int n;
6     while (cin >> n && n != 0) {
7         int d = n;
8         int day = 1;
9         int coin = 0;
10        while (n - day >= 0) {
11            n -= day;
12            coin += day * day;
```

```

13         day++;
14     }
15     if (n > 0) {
16         coin += (n * day);
17     }
18     cout << d << " " << coin << endl;
19 }
20 }

```

2.1.22 The $3n + 1$ Problem

Problems in computer science are often classified as belonging to a certain class of problems (e.g., NP, unsolvable, recursive). In this problem you will analyze a property of an algorithm whose classification is not known for all possible inputs.

Consider the following algorithm:

1. input n
2. print n
3. if $n = 1$ then STOP
4. if n is odd then $n \leftarrow 3n + 1$
5. else $n \leftarrow n/2$
6. GOTO 2

Given the input 22, the following sequence of numbers will be printed: 22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1.

It is conjectured that the algorithm above will terminate (when a 1 is printed) for any integral input value. Despite the simplicity of the algorithm, it is unknown whether this conjecture is true. It has been verified, however, for all integers n such that $0 < n < 1,000,000$ (and, in fact, for many more numbers than this).

Given an input n , it is possible to determine the number of numbers printed before the 1 is printed. For a given n this is called the cycle length of n . In the example above, the cycle length of 22 is 16.

For any two numbers i and j you are to determine the maximum cycle length over all numbers between i and j .

Input

The input will consist of a series of pairs of integers i and j , one pair of integers per line. All integers will be less than 10,000 and greater than 0.

You should process all pairs of integers and for each pair determine the maximum cycle length over all integers between and including i and j .

Output

For each pair of input integers i and j you should output i , j , and the maximum cycle length for integers between and including i and j . These three numbers should be separated by at least one space with all three numbers on one line and with one line of output for each line of input. The integers i and j must appear in the output in the same order in which they appeared in the input and should be followed by the maximum cycle length (on the same line).

Example

Sample Input	Sample Output
1 10	1 10 20
100 200	100 200 125
201 210	201 210 89
900 1000	900 1000 174

Source: Duke Internet Programming Contest 1990.

IDs for online judges: POJ 1207, UVA 100.

Analysis – Phân tích bài toán

Làm theo yêu cầu bài toán thôi, tuy nhiên điểm cần lưu ý ở bài này là không cho ($i < j$), tức là i có thể $> j$. Vậy thay vì duyệt từ i đến j , ta duyệt từ $\min(i,j)$ đến $\max(i,j)$.

Program – Chương trình

```

1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4

```

```

5 int main(){
6     int l, r;
7     while(cin >> l >> r) {
8         int maxCycle = -1;
9         for (int i = min(l,r) ; i <= max(l,r) ; i++) {
10             int num = i;
11             int currentCycle = 0;
12             while(num != 1) {
13                 if (num % 2 == 1) {
14                     num = num * 3 + 1;
15                 }
16                 else {
17                     num = num /2;
18                 }
19                 currentCycle++;
20             }
21             maxCycle = max(maxCycle, currentCycle + 1);
22         }
23         cout << l << " " << r << " " << maxCycle << endl;
24     }
25 }

```

2.1.23 Pascal Library

Pascal University, one of the oldest in the country, needs to renovate its library building, because after all these centuries the building has started to show the effects of supporting the weight of the enormous amount of books it houses.

To help in the renovation, the alumni association of the university decided to organize a series of fund-raising dinners, for which all alumni were invited. These events proved to be a huge success, and several were organized during the past year. (One of the reasons for the success of this initiative seems to be the fact that students that went through the Pascal system of education have fond memories of that time and would love to see a renovated Pascal library.)

The organizers maintained a spreadsheet indicating which alumni participated in each dinner. Now they want your help to determine whether any alumnus or alumna took part in all of the dinners.

Input

The input contains several test cases. The first line of a test case contains two integers, N and D , indicating, respectively, the number of alumni and the number of dinners organized ($1 \leq N \leq 100$ and $1 \leq D \leq 500$). Alumni are identified by integers from 1 to N . Each of the next D lines describes the attendees of a dinner and contains N integers X_i indicating if the alumnus or alumna i attended that dinner ($X_i = 1$) or not ($X_i = 0$). The end of input is indicated by $N = D = 0$.

Output

For each test case in the input your program must produce one line of output, containing either the word “yes,” in case there exists at least one alumnus or alumna that attended all dinners, or the word “no” otherwise.

Example

Sample Input	Sample Output
3 3	Yes
1 1 1	No
0 1 1	
1 1 1	
7 2	
1 0 1 0 1 0 1	
0 1 0 1 0 1 0	
0 0	

Source: ACM South America 2005.

IDs for online judges: : POJ 2864, UVA 3470.

Analysis – Phân tích bài toán

Program – Chương trình

Ta chỉ cần duyệt theo từng cột, nếu toàn bộ đều là 1 thì xuất "yes", ngược lại nếu sau khi duyệt tất cả các cột mà vẫn không tồn tại điều kiện thỏa mãn thì xuất "no".

```

1  #include <iostream>
2  #include <vector>
3  using namespace std;
4
5  int main(){
6      int n, d;
7      while (cin >> n >> d && n != 0 && d != 0) {
8          vector<vector<int>>> a(d, vector<int>(n));
9          for (int i = 0; i < d; i++) {
10             for (int j = 0; j < n; j++) {
11                 cin >> a[i][j];
12             }
13         }
14         bool check = false;
15         for (int i = 0; i < n; i++) {
16             bool attended_check = true;
17             for (int j = 0; j < d; j++) {
18                 if (a[j][i] == 0) {
19                     attended_check = false;
20                     break;
21                 }
22             }
23             if (attended_check) {
24                 check = true;
25                 break;
26             }
27         }
28         if (check) cout << "yes";
29         else cout << "no";
30         cout << endl;
31     }
32 }

```

2.1.24 Calendar

Most of us have a calendar on which we scribble details of important events in our lives—visits to the dentist, the Regent 24-hour book sale, programming contests, and so on. However, there are also the fixed dates—partner’s birthdays, wedding anniversaries, and the like—and we also need to keep track of these. Typically we need to be reminded of when these important dates are approaching—the more important the event, the further in advance we wish to have our memories jogged.

Write a program that will provide such a service. The input will specify the year for which the calendar is relevant (in the range 1901–1999). Bear in mind that, within the range specified, all years that are divisible by 4 are leap years and hence have an extra day (February 29) added. The output will specify “today’s” date, a list of forthcoming events and an indication of their relative importance.

Input

The first line of input will contain an integer representing the year (in the range 1901–1999). This will be followed by a series of lines representing anniversaries or days for which the service is requested.

An anniversary line will consist of the letter ‘A’; three integer numbers (D, M, P) representing the date, the month, and the importance of the event; and a string describing the event, all separated by one or more spaces. P will be a number between 1 and 7 (both inclusive) and represents the number of days before the event that the reminder service should start. The string describing the event will always be present and will start at the first nonblank character after the priority. A date line will consist of the letter ‘D’ and the date and month as above.

All anniversary lines will precede any date lines. No line will be longer than 255 characters in total. The file will be terminated by a line consisting of a single #.

Output

Output will consist of a series of blocks of lines, one for each date line in the input. Each block will consist of the requested date followed by the list of events for that day and as many following days as necessary.

The output should specify the date of the event (D and M), right justified in fields of width 3, and the relative importance of the event. Events that happen today should be flagged as shown below, events that happen tomorrow should have P stars, events that happen the day after tomorrow should have P – 1 stars, and so on. If several events are scheduled for the same day, order them by relative importance (number of stars).

If there is still a conflict, order them by their appearance in the input stream. Follow the format used in the example below. Leave one blank line between blocks

Example

Source: New Zealand Contest 1993.

Sample Input	Sample Output
1993	
A 23 12 5 Partner's birthday	Today is: 20 12
A 25 12 7 Christmas	20 12 *TODAY* Unspecified anniversary
A 20 12 1 Unspecified anniversary	text23 12 *** Partner's birthday
D 20 12	25 12 *** Christmas
#	

IDs for online judges: UVA 158.

Analysis – Phân tích bài toán

Nếu số ngày cách ngày khảo sát vượt quá 7 ngày thì chẳng cần xét.

Tính số ngày cách ngày khảo sát bằng công thức: như trong code (lười quá tại bị Wrong Answer mà chẳng biết tại sao lại thế?).

Program – Chương trình

```

1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <algorithm>
5  using namespace std;
6
7  struct event{
8      int day;
9      int month;
10     int important;
11     string name;
12     int idxInput;
13 };
14
15 bool compare(const event &x, const event &y) {
16     if (x.month != y.month) {
17         return x.month < y.month;
18     }
19     if (x.day != y.day) {
20         return x.day < y.day;
21     }
22     if (x.important != y.important) {
23         return x.important > y.important;
24     }
25     return x.idxInput < y.idxInput;
26 }
27
28 int get_day_in_month(int month, bool leap) {
29     if (month == 1 || month == 3 || month == 5 || month == 7 || month == 8 || month == 10 || month == 12)
30         return 31;
31     }
32     else if (month == 2) {
33         if (leap) {
34             return 29;
35         }
36         else return 28;
37     }
38     else {
39         return 30;
40     }
41 }
42
43 int main(){
44     vector<event> a;
45     int year; cin >> year;
46     char c;
47
48     bool leap = (year % 4 == 0);
49
50     int indexInput = 0;
51     while (cin >> c && c != '#') {
52         if (c == 'A') {
53             int day, month, important;
54             string name;
55             cin >> day >> month >> important;
56             cin.ignore();

```

```

57         getline(cin, name);
58         a.push_back({day, month, important, name, indexInput});
59         indexInput++;
60     }
61     else if (c == 'D') {
62         int day, month;
63         cin >> day >> month;
64         cout << "Today is: " << day << " " << month << endl;
65         sort(a.begin(), a.end(), compare);
66         int idx = 0;
67         while (idx < a.size()) {
68             if (a[idx].day == day && a[idx].month == month) {
69                 break;
70             }
71             idx++;
72         }
73         if (idx == a.size()) idx = 0;
74
75         for (int i = idx; i < a.size(); i++) {
76
77             if (a[i].day == day && a[i].month == month) {
78                 cout << a[i].day << " " << a[i].month << " ";
79                 cout << "*TODAY*" << a[i].name << endl;
80             }
81             else {
82                 int day_diff = 0;
83
84                 for (int j = month; j < a[i].month; j++) {
85                     day_diff += get_day_in_month(j, leap);
86                 }
87                 day_diff += a[i].day;
88                 day_diff -= day;
89
90                 if (day_diff >= 1 && day_diff <= 7) {
91                     cout << a[i].day << " " << a[i].month << " ";
92                     for (int j = day_diff; j <= a[i].important; j++){
93                         cout << "*";
94                     }
95                     cout << " " << a[i].name << endl;
96                 }
97             }
98         }
99     }
100 }
101

```

2.1.25 Manager

One of the programming paradigms in parallel processing is the producer-consumer paradigm that can be implemented using a system with a “manager” process and several “client” processes. The clients can be producers, consumers, and so on. The manager keeps a trace of client processes.

Each process is identified by its cost, which is a strictly positive integer in the range 1 .. 10,000. The number of processes with the same cost cannot exceed 10,000. The queue is managed according to three types of requests, as follows:

a x: Add to the queue the process with the cost x.

r: Remove a process, if possible, from the queue according to the current manager policy.

p i: Enforce the policy i of the manager, where i is 1 or 2. The default manager policy is 1.

e: Ends the list of requests.

There are two manager policies:

1. Remove the minimum cost process.
2. Remove the maximum cost process.

The manager will print the cost of a removed process only if the ordinal number of the removed process is in the removal list. Your job is to write a program that simulates the manager process.

Input

The input is from the standard input. Each data set in the input has the following format:

The maximum cost of the processes.

The length of the removal list.

The removal list—the list of ordinal numbers of the removed processes that will be displayed. For example, 1 4 means that the cost of the first and fourth removed processes will be displayed.

The list of requests, each on a separate line

Each data set ends with an e request. The data sets are separated by empty lines.

Output

The program prints on standard output the cost of each process that is removed, provided that the ordinal number of the remove request is in the list and the queue is not empty at that moment. If the queue is empty, the program prints -1 . The results are printed on separate lines. An empty line separates the results of different data sets.

An example is given in the following:

Example

Sample Input	Sample Output
5	2
2	5
1 3	
a 2	
a 3	
r	
a 4	
p 2	
r	
a 5	
r	
e	

Source: ACM Southeastern Europe 2002.

IDs for online judges: POJ 1281, UVA 2514.

Analysis – Phân tích bài toán

Mấu chốt bài này là dùng **mảng đánh dấu** để tìm phần tử lớn nhất hoặc nhỏ nhất.

Nếu $p = 1$, ta duyệt từ 0 lên maximum cost, nếu $p = 2$ ta duyệt từ maximum cost xuống 0. Nếu tìm thấy phần tử $\text{maximum_cost}[i] > 0$ thì giảm giá trị $\text{maximum_cost}[i]$ đi 1 đơn vị, và xét nếu lần loại bỏ hiện tại có nằm trong danh sách **removal list** hay không, nếu có thì in ra i vừa giảm. Ngược lại, nếu không tìm thấy thì và lần loại bỏ nằm trong **removal list** thì in ra -1 theo yêu cầu đề.

Program – Chương trình

```
1 #include <iostream>
2 #include <vector>
3
4 using namespace std;
5
6
7 int main(){
8
9     int maximum_cost;
10    int n;
11    while(cin >> maximum_cost >> n){
12        vector<int>costs(maximum_cost + 1, 0);
13        vector<int>removal_list(n);
14        for (int i = 0; i < n; i++) {
15            cin >> removal_list[i];
16        }
17
18        char c;
19        int cnt = 0;
20        int policy = 1;
21        while (cin >> c && c != 'e') {
```

```

22     if (c == 'a') {
23         int cost; cin >> cost;
24         costs[cost]++;
25     }
26     else if (c == 'p') {
27         int p; cin >> p;
28         policy = p;
29     }
30     else if (c == 'r'){
31         cnt++;
32         bool check = false;
33         for (int j = 0; j < removal_list.size(); j++) {
34             if (removal_list[j] == cnt) {
35                 check = true;
36             }
37         }
38
39         int val = -1;
40
41         if (policy == 1) {
42             for (int j = 0; j <= maximum_cost; j++) {
43                 if (costs[j] == 0) continue;
44                 costs[j]--;
45                 val = j;
46                 break;
47             }
48         }
49         else {
50             for (int j = maximum_cost; j >= 0; j--) {
51                 if (costs[j] == 0) continue;
52                 costs[j]--;
53                 val = j;
54                 break;
55             }
56         }
57         if (check) {
58             cout << val << endl;
59         }
60     }
61 }
62 cout << endl;
63 }
64 }
65 }

```

2.1.26 Đánh Giá Chủ Quan Chap 2 - Section 1

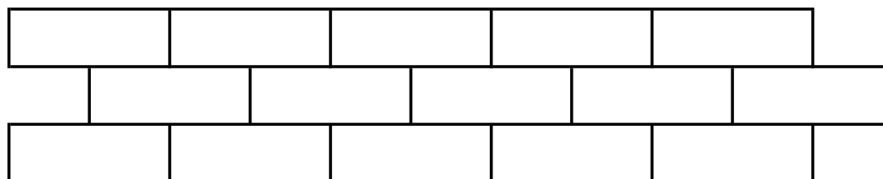
Chap này không quá yêu cầu về mặt thuật toán, mà chủ yếu ở phần cài đặt code, làm xong là tay to hơn rõ về phần cài đặt code.

Cái hay là vậy, nhưng chap này cũng khiến tôi khá ức chế vì yêu cầu quá nhiều về định dạng, phải đọc rất kỹ mới thấy được những yêu cầu đó mà theo tôi là không hề cần thiết.

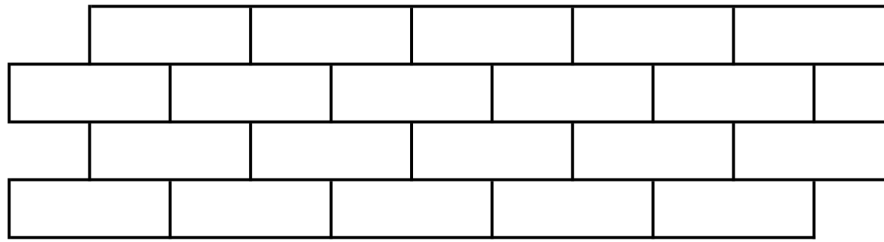
3 Linear Algebra – Chuyển bài thi OLP Toán học môn ĐSTT về lập trình

3.1 Đại số 2023 - B.4.

- a. Có bao nhiêu cách chọn ra 3 viên gạch, mỗi viên từ một hàng trong hình sau đây, sao cho không có hai viên gạch nào được lấy ra nằm kề nhau (hai viên gạch được gọi là kề nhau nếu có chung một phần của một cạnh)?



- b. Có bao nhiêu cách chọn ra 4 viên gạch, mỗi viên từ một hàng trong hình sau đây, sao cho không có hai viên gạch nào được lấy ra nằm kề nhau?



Analysis – Phân tích bài toán

Nhìn qua, ta thấy được đây là một bài tổ hợp, có thể tách ra thành bài toán con để giải quyết, rút ra nhận xét rằng bài toán đang hướng đến kỹ thuật quy hoạch động.

Theo đề bài, với 3 hàng ta cần đếm số cách chọn ra 3 viên gạch, với 4 hàng ta cần đếm số cách chọn ra 4 viên gạch. Vậy từ những dữ kiện trên, ta rút ra được bài toán tổng quát là xét n hàng, mỗi hàng có c viên gạch, đếm số cách chọn n viên (mỗi hàng 1 viên) sao cho không có hai viên gạch nào được lấy ra nằm kề nhau.

Gọi $dp[n][c]$ là số cách chọn khi xét đến hàng thứ n , đang chọn viên gạch thứ c . Bài toán cơ sở là $dp[1][i] = 1$ - số cách chọn viên gạch thứ i ở hàng 1 luôn là 1.

Do các hàng gạch chồng so-le nhau, để ý nếu xét hàng đầu là hàng thụt vào thì ở hàng chẵn, nếu ta chọn viên đầu tiên thì xét hàng trước đó ta luôn chọn được tất cả các viên ngoại trừ viên đầu hàng. Xét các viên gạch còn lại, chọn được tất cả các viên ở hàng trước ngoại trừ 2 viên: viên cùng chỉ số và viên có chỉ số ngay trước nó.

Ý tưởng cũng tương tự với hàng lẻ.

Kết quả bài toán: $\sum_{i=1}^c dp[n][i]$ - Số cách chọn là tổng số cách khi xét đến hàng n , ta chọn viên gạch thứ i .

Program – Chương trình

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  int main() {
5      int n, c;
6      cin >> n >> c;
7      vector<vector<int>>> dp(n + 1, vector<int>(c + 1, 0));
8      for (int i = 1; i <= c; i++) {
9          dp[1][i] = 1;
10     }
11     for (int i = 2; i <= n; i++) {
12         for (int j = 1; j <= c; j++) {
13             if (i % 2 == 1) {
14                 for (int brickInPreviousRow = 1; brickInPreviousRow <= c; brickInPreviousRow
15                     ++){
16                     if (brickInPreviousRow == j || brickInPreviousRow + 1 == j) {
17                         continue;
18                     }
19                     dp[i][j] += dp[i - 1][brickInPreviousRow];
20                 }
21             }
22             else {
23                 for (int brickInPreviousRow = 1; brickInPreviousRow <= c; brickInPreviousRow
24                     ++){
25                     if (brickInPreviousRow == j || brickInPreviousRow - 1 == j) {
26                         continue;
27                     }
28                     dp[i][j] += dp[i - 1][brickInPreviousRow];
29                 }
30             }
31         }
32         int ans = 0;
33         for (int i = 1; i <= c; i++) {
34             ans += dp[n][i];
35         }
36         cout << ans;
37     }
38 }
```

4 Miscellaneous

4.1 Contributors

1. NGUYỄN QUẢN BÁ HỒNG - TEMPLATE: https://github.com/NQBH/advanced_STEM_beyond/blob/main/VMC/NQBH_VMC.tex.