Input Format

Input will contain four integers - a, b, c, d , one per line.

Output Format

Return the greatest of the four integers.

PS: I/O will be automatically handled.

Sample Input

```
3
4
6
5
```

Sample Output

```
6
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1 3
2 4
3 6
4 5
```

Your Output (stdout)

```
1 6
```

Expected Output Download

```
1 6
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) **Download**

```
1 4
2 5
```

Your Output (stdout)

```
1 9
2 1
```

Expected Output **Download**

```
1 9
2 1
```

- vector<int>: a vector of the parsed integers.

Note You can learn to push elements onto a vector by solving the first problem in the STL chapter.

Input Format

There is one line of n integers separated by commas.

Constraints

The length of str is less than 8×10^5.

Sample Input

```
23,4,56
```

Sample Output

```
23
4
56
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1 23,4,56
```

Your Output (stdout)

```
1 23
2 4
3 56
```

Expected Output Download

```
1 23
2 4
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1 4
2 5
```

Your Output (stdout)

```
1 9
2 1
```

Expected Output Download

```
1 9
2 1
```

Sample Output 1

```
eight
```

Explanation 1

`eight` is the English word for the number **8**.

Sample Input 2

```
44
```

Sample Output 2

```
Greater than 9
```

Explanation 2

$n = 44$ is greater than 9, so we print `Greater than 9`.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) Download

```
1 5
```

Your Output (stdout)

```
1 five
```

Expected Output Download

```
1 five
```

The first line of the input contains N,where N is the number of integers.The next line contains N space-separated integers.

Constraints

$1 <= N <= 1000$

$1 <= A[i] <= 10000$, where $A[i]$ is the i^{th} integer in the array.

Output Format

Print the N integers of the array in the reverse order, space-separated on a single line.

Sample Input

```
4
1 4 3 2
```

Sample Output

```
2 3 4 1
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1  4
2  1 4 3 2
```

Your Output (stdout)

```
1  2 3 4 1
```

Expected Output Download

```
1  2 3 4 1
```

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

You are required to extend the existing code so that it handles std::invalid_argument exception properly. More specifically, you have to extend the implementation of `process_input` function. It takes integer n as an argument and has to work as follows:

1. It calls function `largest_proper_divisor(n)`.
2. If this call returns a value without raising an exception, it should print in a single line `result=d` where d is the returned value.
3. Otherwise, if the call raises a `std::invalid_argument` exception, it has to print in a single line the string representation of the raised exception, i.e. its message.
4. Finally, no matter if the exception is raised or not, it should print in a single line `returning control flow to caller` after any other previously printed output.

To keep the code quality high, you are advised to have exactly one line printing `returning control flow to caller` in the body of `process_input` function.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template. In the only line of the input, there is a single integer n, which is going to be the argument passed to function `process_input`.

Constraints

- $0 \leq n \leq 100$

Upload Code as File | Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Sample Test case 1

Input (stdin) Download

```
0
```

Your Output (stdout)

```
largest proper divisor is not defined for n=0
returning control flow to caller
```

Expected Output Download

```
largest proper divisor is not defined for n=0
returning control flow to caller
```

In this challenge, the task is to debug the existing code to successfully execute all provided test files.

The given code defines two classes `HotelRoom` and `HotelApartment` denoting respectively a standard hotel room and a hotel apartment. An instance of any of these classes has two parameters: `bedrooms` and `bathrooms` denoting respectively the number of bedrooms and the number of bathrooms in the room.

The prices of a standard hotel room and hotel apartment are given as:

- Hotel Room: $50 \cdot bedrooms + 100 \cdot bathrooms$.
- Hotel Apartment: The price of a standard room with the same number bedrooms and bathrooms plus 100.

For example, if a standard room costs 200, then an apartment with the same number of bedrooms and bathrooms costs 300.

In hotel's codebase, there is a piece of code reading the list of rooms booked for today and calculates the total profit for the hotel. However, sometimes calculated profits are lower than they should be.

Debug the existing `HotelRoom` and `HotelApartment` classes' implementations so that the existing code computing the total profit returns a correct profit.

Your function will be tested against several cases by the locked template code.

Input Format

The input is read by the provided locked code template.

In the first line, there is a single integer n denoting the number of rooms booked for

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) — Download

```
2
standard 3 1
apartment 1 1
```

Your Output (stdout)

```
500
```

Expected Output — Download

```
500
```

$1 <= A_i <= 10000$, where A_i is the i^{th} element in the array A.

Output Format

For each of the contiguous subarrays of size K of each array, you have to print the maximum integer.

Sample Input

```
2
5 2
3 4 6 3 4
7 4
3 4 5 8 1 4 10
```

Sample Output

```
4 6 6 4
8 8 8 10
```

Explanation

For the first case, the contiguous subarrays of size 2 are {3,4},{4,6},{6,3} and {3,4}. The 4 maximum elements of subarray of size 2 are: 4 6 6 4.

For the second case,the contiguous subarrays of size 4 are {3,4,5,8},{4,5,8,1},{5,8,1,4} and {8,1,4,10}. The 4 maximum element of subarray of size 4 are: 8 8 8 10.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
2
5 2
3 4 6 3 4
7 4
3 4 5 8 1 4 10
```

Your Output (stdout)

```
4 6 6 4
8 8 8 10
```

Expected Output

Download

```
4 6 6 4
```

Consider an n-element array, a, where each index i in the array contains a reference to an array of k_i integers (where the value of k_i varies from array to array). See the Explanation section below for a diagram.

Given a, you must answer q queries. Each query is in the format `i j`, where i denotes an index in array a and j denotes an index in the array located at $a[i]$. For each query, find and print the value of element j in the array at location $a[i]$ on a new line.

Click here to know more about how to create variable sized arrays in C++.

Input Format

The first line contains two space-separated integers denoting the respective values of n (the number of variable-length arrays) and q (the number of queries).
Each line i of the n subsequent lines contains a space-separated sequence in the format k $a[i]_0$ $a[i]_1$ ... $a[i]_{k-1}$ describing the k-element array located at $a[i]$.
Each of the q subsequent lines contains two space-separated integers describing the respective values of i (an index in array a) and j (an index in the array referenced by $a[i]$) for a query.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq q \leq 10^5$
- $1 \leq k \leq 3 \cdot 10^5$
- $n \leq \sum k \leq 3 \cdot 10^5$
- $0 \leq i < n$

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
2 2
3 1 5 4
5 1 2 8 9 3
0 1
1 3
```

Your Output (stdout)

```
5
9
```

Expected Output Download

```
5
```

C++ provides a nice alternative data type to manipulate strings, and the data type is conveniently called string. Some of its widely used features are the following:

- Declaration:

```
string a = "abc";
```

- Size:

```
int len = a.size();
```

- Concatenate two strings:

```
string a = "abc";
string b = "def";
string c = a + b; // c = "abcdef".
```

- Accessing i^{th} element:

```
string s = "abc";
char   c0 = s[0];   // c0 = 'a'
char   c1 = s[1];   // c1 = 'b'
char   c2 = s[2];   // c2 = 'c'

s[0] = 'z';         // s = "zbc"
```

P.S.: We will use cin/cout to read/write a string.

Upload Code as File

Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to r

Sample Test case 0

Input (stdin)

```
1  abcd
2  ef
```

Your Output (stdout)

```
1  4 2
2  abcdef
3  ebcd af
```

Expected Output

```
1  4 2
2  abcdef
3  ebcd af
```

The task is to overload the << operator for Person class in such a way that for **p** being an instance of class Person the result of:

```
std::cout << p << " " << <some_string_value> << std::endl;
```

produces the following output:

```
first_name=<first_name>,last_name=<last_name> <some_string_value>
```

where:

- `<first_name>` is the value of p's `first_name_`
- `<last_name>` is the value of p's `last_name_`
- `<some_string_value>` is an arbitrary `std::string value`

Input Format

The input is read by the provided locked code template. In the only line of the input there are 3 space-separated strings `first_name`, `last_name`, `event`. The values of `first_name` and `last_name` will be used to create an object **p** of type Person. The value of `event` will be used by the provided code to produce the output.

Constraints

- Each word in the input contains only English letters and is no longer than 15 characters

Output Format

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1  john doe registered
```

Your Output (stdout)

```
1  first_name=john,last_name=doe registered
```

Expected Output Download

```
1  first_name=john,last_name=doe registered
```

While playing a video game, you are battling a powerful dark wizard. He casts his spells from a distance, giving you only a few seconds to react and conjure your counterspells. For a counterspell to be effective, you must first identify what kind of spell you are dealing with.

The wizard uses scrolls to conjure his spells, and sometimes he uses some of his generic spells that restore his stamina. In that case, you will be able to extract the name of the scroll from the spell. Then you need to find out how similar this new spell is to the spell formulas written in your spell journal.

Spend some time reviewing the locked code in your editor, and complete the body of the counterspell function.

Check Dynamic cast to get an idea of how to solve this challenge.

Input Format

The wizard will read t scrolls, which are hidden from you.

Every time he casts a spell, it's passed as an argument to your counterspell function.

Constraints

- $1 \leq t \leq 100$
- $1 \leq |s| \leq 1000$, where s is a scroll name.
- Each scroll name, s, consists of uppercase and lowercase letters.

Output Format

After identifying the given spell, print its name and power.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin)

```
3
fire 5
AquaVitae 999 AruTaVae
frost 7
```

Your Output (stdout)

```
Fireball: 5
6
Frostbite: 7
```

Expected Output

Download

```
Fireball: 5
```

You are given three classes A, B and C. All three classes implement their own version of func.

In class A, func multiplies the value passed as a parameter by **2**:

```
class A
{
    public:
        A(){
            callA = 0;
        }
    private:
        int callA;
        void inc(){
            callA++;
        }

    protected:
        void func(int & a)
        {
            a = a * 2;
            inc();
        }
    public:
        int getA(){
            return callA;
        }
};
```

In class B, func multiplies the value passed as a parameter by 3:

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
30
```

Your Output (stdout)

```
Value = 30
A's func called 1 times
B's func called 1 times
C's func called 1 times
```

Expected Output Download

```
Value = 30
A's func called 1 times
B's func called 1 times
```

$1 <= N <= 500000$

$1 <= M <= 1000$

$1 <= key <= 20$

$1 <= value <= 2000$

Output Format

The code provided in the editor will use your derived class LRUCache to output the value whenever a get command is encountered.

Sample Input

```
3 1
set 1 2
get 1
get 2
```

Sample Output

```
2
-1
```

Explanation

Since, the capacity of the cache is 1, the first set results in setting up the key 1 with it's value 2. The first get results in a cache hit of key 1, so 2 is printed as the value for the first get. The second get is a cache miss, so -1 is printed.

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1

Input (stdin) Download

```
3 1
set 1 2
get 1
get 2
```

Your Output (stdout)

```
2
-1
```

Expected Output Download

```
2
-1
```

21 November 202

A pointer in C++ is used to share a memory address among different contexts (primarily functions). They are used whenever a function needs to modify the content of a variable, but it does not have ownership.

In order to access the memory address of a variable, ***val***, prepend it with **&** sign. For example. `&val` returns the memory address of ***val***.

This memory address is assigned to a pointer and can be shared among functions. For example, ***int* p = &val*** assigns the memory address of ***val*** to pointer ***p***. To access the content of the memory pointed to, prepend the variable name with a *. For example, `*p` will return the value stored in ***val*** and any modification to it will be performed on ***val***.

```
void increment(int *v) {
    (*v)++;
}

int main() {
    int a;
    scanf("%d", &a);
    increment(&a);
    printf("%d", a);
    return 0;
}
```

Function Description

Complete the update function in the editor below.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0 Download

Input (stdin)

```
4
5
```

Your Output (stdout)

```
9
1
```

Expected Output Download

```
9
1
```

Problem

Objective

This is a simple challenge to help you practice printing to stdout. You may also want to complete Solve Me First in C++ before attempting this challenge.

We're starting out by printing the most famous computing phrase of all time! In the editor below, use either printf or cout to print the string **Hello, World!** to stdout.

The more popular command form is **cout**. It has the following basic form:

```
cout<<value_to_print<<value_to_print;
```

Any number of values can be printed using one command as shown.

The **printf** command comes from C language. It accepts an optional format specification and a list of variables. Two examples for printing a string are:

```
printf("%s", string); printf(string);
```

Note that neither method adds a newline. It only prints what you tell it to.

Output Format

Print **Hello, World!** to stdout.

Sample Output

```
Hello, World!
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) — Download

```
1
```

Your Output (stdout)

```
1  Hello, World!
```

Expected Output — Download

```
1  Hello, World!
```

In real life applications and systems, a common component is a messaging system. Thea idea is that a sender sends messages to the recipient. The messages might be sent for example over the network. However, some network protocols don't guarantee to preserve the order of sent messages while they are received by the recipient. For example, if someone sends a text messages `hello, hi` and `what's up`, they might be received in the order `what's up, hello, hi`. In many systems the expected behavior is to preserve the order, so the order of sent messages is the same as the order of received messages.

In this problem, the task is to implement a software layer over the top of a network protocol sending messages in arbitrary order, in such a way that the sent messages are printed by the recipient in the order they were sent.

In the template code below, there are implementations of classes `Recipient` and `Network`.

Your task is to implement classes `Message` and `MessageFactory` according to the below specification:

Class `Message` is required to store a text value of type `std::string` and provide a public getter `const string& get_text()` which is expected to return this text value. Besides that, it should implement the < operator that will be used in `fix_order()` method of the recipient to fix the order of received messages. Feel free to implement any other methods and class/instance variables. In particular, you can implement any additional constructors, but make sure that you provide an empty constructor, i.e. the one without arguments.

Class `MessageFactory` is required to have an empty constructor, and implement a

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1 Alex
2 Hello Monique!
3 What'up?
4 Not much :(
```

Your Output (stdout)

```
1 Alex
2 Hello Monique!
3 What'up?
4 Not much :(
```

Expected Output Download

Output Format

There are two types of output depending on the object.

If the object is of type Professor, print the space separated name, age, publications and id on a new line.

If the object is of the Student class, print the space separated name, age, the sum of the marks in **6** subjects and id on a new line.

Sample Input

```
4
1
Walter 56 99
2
Jesse 18 50 48 97 76 34 98
2
Pinkman 22 10 12 0 18 45 50
1
White 58 87
```

Sample Output

```
Walter 56 99 1
Jesse 18 403 1
Pinkman 22 135 2
White 58 87 2
```

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
4
1
Walter 56 99
2
Jesse 18 50 48 97 76 34 98
2
Pinkman 22 10 12 0 18 45 50
1
White 58 87
```

Your Output (stdout)

```
Walter 56 99 1
```

A template parameter pack is a template parameter that accepts zero or more template arguments (non-types, types, or templates). To read more about parameter pack, click here.

Create a template function named reversed_binary_value. It must take an arbitrary number of bool values as template parameters. These booleans represent binary digits in reverse order. Your function must return an integer corresponding to the binary value of the digits represented by the booleans. For example: reversed_binary_value<0,0,1>() should return 4.

Input Format

The first line contains an integer, t, the number of test cases. Each of the t subsequent lines contains a test case. A test case is described as 2 space-separated integers, x and y, respectively.

- x is the value to compare against.
- y represents the range to compare: $64 \times y$ to $64 \times y + 63$.

Constraints

- $0 \leq x \leq 65535$
- $0 \leq y \leq 1023$
- The number of template parameters passed to reversed_binary_value will be ≤ 16.

Output Format

Each line of output contains 64 binary characters (i.e., 0's and 1's). Each character

Upload Code as File Test against custom input Run Code Submit

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
2
65 1
10 0
```

Your Output (stdout)

```
0100000000000000000000000000000000000000000000000000000000000000
0000000000100000000000000000000000000000000000000000000000000000
```

Expected Output Download

```
0100000000000000000000000000000000000000000000000000000000000000
```

Constraints

$1 \leq T \leq 10^3$

$0 \leq A, B \leq 2^{60}$

Output Format

For each test case, print a single line containing whichever message described in the Problem Statement above is appropriate. After all messages have been printed, the locked stub code in your editor prints the server load.

Sample Input

```
2
-8 5
1435434255433 5
```

Sample Output

```
Exception: A is negative
Not enough memory
2
```

Explanation

-8 is negative, hence 'Exception: A is negative' is thrown. Since the second input is too large, 'not enough memory' is displayed. 2 is the server load.

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
2
-8 5
1435434255433 5
```

Your Output (stdout)

```
Exception: A is negative
Not enough memory
2
```

Expected Output Download

```
Exception: A is negative
Not enough memory
```

In this challenge, we work with string streams.

stringstream is a stream class to operate on strings. It implements input/output operations on memory (string) based streams. stringstream can be helpful in different type of parsing. The following operators/functions are commonly used here

- Operator >> Extracts formatted data.
- Operator << Inserts formatted data.
- Method str() Gets the contents of underlying string device object.
- Method str(string) Sets the contents of underlying string device object.

Its header file is sstream.

One common use of this class is to parse comma-separated integers from a string (e.g., "23,4,56").

```
stringstream ss("23,4,56");
char ch;
int a, b, c;
ss >> a >> ch >> b >> ch >> c;  // a = 23, b = 4, c = 56
```

Here ch is a storage area for the discarded commas.

If the >> operator returns a value, that is a true value for a conditional. Failure to return a value is false.

Given a string of comma delimited integers, return a vector of integers.

Function Description

Complete the parseInts function in the editor below.

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
1  23,4,56
```

Your Output (stdout)

```
1  23
2  4
3  56
```

Expected Output Download

```
1  23
2  4
3  56
```

You inherited a piece of code that performs username validation for your company's website. The existing function works reasonably well, but it throws an exception when the username is too short. Upon review, you realize that nobody ever defined the exception.

The inherited code is provided for you in the locked section of your editor. Complete the code so that, when an exception is thrown, it prints `Too short: n` (where n is the length of the given username).

Input Format

The first line contains an integer, t, the number of test cases.
Each of the t subsequent lines describes a test case as a single username string, u.

Constraints

- $1 \leq t \leq 1000$
- $1 \leq |u| \leq 100$
- The username consists only of uppercase and lowercase letters.

Output Format

You are not responsible for directly printing anything to stdout. If your code is correct, the locked stub code in your editor will print either `Valid` (if the username is valid), `Invalid` (if the username is invalid), or `Too short: n` (where n is the length of the too-short username) on a new line for each test case.

Sample Input

3

Upload Code as File Test against custom input

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) Download

```
3
Peter
Me
Arxwwz
```

Your Output (stdout)

```
Valid
Too short: 2
Invalid
```

Expected Output Download

```
Valid
```

Objective

In this challenge, we practice reading input from stdin and printing output to stdout.

In C++, you can read a single whitespace-separated token of input using cin, and print output to stdout using cout. For example, let's say we declare the following variables:

```
string s;
int n;
```

and we want to use cin to read the input "High 5" from stdin. We can do this with the following code:

```
cin >> s >> n;
```

This reads the first word ("High") from stdin and saves it as string s, then reads the second word ("5") from stdin and saves it as integer n. If we want to print these values to stdout, separated by a space, we write the following code:

```
cout << s << " " << n << endl;
```

This code prints the contents of string s, a single space (`" "`), then the integer n. We end our line of output with a newline using endl. This results in the following output:

```
High 5
```

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

Sample Test case 0

Input (stdin) — Download

```
1 2 7
```

Your Output (stdout)

```
10
```

Expected Output — Download

```
10
```

if and else are two of the most frequently used conditionals in C/C++, and they enable you to execute zero or one conditional statement among many such dependent conditional statements. We use them in the following ways:

1. if: This executes the body of bracketed code starting with ***statement1*** if ***condition*** evaluates to true.

```
if (condition) {
   statement1;
   ...
}
```

2. if - else: This executes the body of bracketed code starting with ***statement1*** if ***condition*** evaluates to true, or it executes the body of code starting with ***statement2*** if ***condition*** evaluates to false. Note that only one of the bracketed code sections will ever be executed.

```
if (condition) {
   statement1;
   ...
}
else {
   statement2;
   ...
}
```

3. if - else if - else: In this structure, dependent statements are chained together and the

Congratulations!

You have passed the sample test cases. Click the submit button to run your code against all the test cases.

- Sample Test case 0
- Sample Test case 1
- Sample Test case 2

Input (stdin) Download

```
1 5
```

Your Output (stdout)

```
1 five
```

Expected Output Download

```
1 five
```