Overview: Print out an ASCII triangle

Description: Given a number *n*, print out an ASCII triangle with *n* levels using asterisks as the output. The triangle should be *n* rows tall, with each row having an increasing number of asterisks, and the last row should have *n* asterisks. For example, if *n*=5, then the desired output will look like

```
*
**
***
****
*****
```

Filename: bug1.{java, cpp, c, cc, py}

Input: The input will contain exactly one line containing *n*, the number of levels.

Output: A triangle with the specifications given in the problem statement.

Assumptions: $1 \le n \le 1{,}000$

Sample Input #1:

```
5
```

Sample Output #1:

```
*
**
***
****
*****
```

Sample Input #2:

```
10
```

Sample Output #2:

```
*
**
***
****
*****
******
*******
********
*********
**********
```

Overview: Print the n^{th} Fibonacci number

Description: The Fibonacci numbers are a sequence in which each number is the sum of the previous two. More formally, $f(n) = f(n-1)+f(n-2)$ for $n>2$, with $f(1) = f(2) = 1$. Given an integer n, output $f(n)$.

Filename: bug2.{java, cpp, c, cc, py}

Input: The input will contain exactly one line containing a single integer n

Output: The n^{th} Fibonacci number

Assumptions: $1 \le n \le 30$

Sample Input #1:
```
5
```

Sample Output #1:
```
5
```

Sample Input #2:
```
20
```

Sample Output #2:
```
6765
```

Overview: Given a list of strings, sort them based on the numbers at the end of the strings.

Description: Given a list of strings whose suffixes are integers, your talks is to sort the strings in ascending order based on the integers at the end of the strings. For example, “cloud9” would be placed after “you4”, since 4 < 9. If the numbers at the end of two strings are equal, output them in the same order as they are given.

Filename: bug3.{java, cpp, c, cc, py}

Input: The input will contain exactly n+1 lines.
The first line contains n, the number of strings.
Lines 2 to n+1 contain the strings to be sorted.

Output: n lines containing the sorted strings as specified in the description.

Assumptions: $1 \le n \le 1{,}000$
Each string will be less than 100 characters long. All characters will be alphanumeric.
Every string will end with a number, which will be at most four characters long (i.e. less than 10,000).

Sample Input #1:
```
4
bug3
bug1
bug5
bug2
```

Sample Output #1:
```
bug1
bug2
bug3
bug5
```

Sample Input #2:
```
4
13goingon30
catch22
aroundtheworldin80
0dark30
```

Sample Output #2:
```
catch22
13goingon30
0dark30
aroundtheworldin80
```

Overview: Find the smallest number of points that cover all of the given intervals

Description: Given a set of *n* intervals *[a, b]*, a set of points *S* covers the set of intervals if every interval contains at least one of the points in *S*. An interval *[a,b]* contains the point *p* if $a \le p \le b$. You must find the smallest possible size of *S*.

Filename: bug4.{java, cpp, c, cc, py}

Input: The first line of the input will contain a single integer *n*: the number of intervals
The following *n* lines will each specify an interval. They will contain two space separated integers *a b*: the endpoints of the interval.

Output: Output a single integer: the minimum number of points needed to cover all the intervals.

Assumptions: $1 \le n \le 100$
$0 \le a \le b \le 100{,}000$ for every interval

Sample Input #1:
```
1
1 10
```

Sample Output #1:
```
1
```

Sample Input #2:
```
3
1 3
2 10
8 10
```

Sample Output #2:
```
2
```

Overview: Find a query within an sorted array

Description: You are given a query integer *q* and a sorted array of *n* distinct integers, *a[0], a[1], … a[n - 1]*. The elements in the array will be strictly increasing order. Find the index at which the query appears in the array.

Filename: bug5.{java, cpp, c, cc, py}

Input: The first line of input will consist of two integers: *n* and *q*
The next line of input will consist of *n* integers: *a[0], a[1], … , a[n - 1]*

Output: Output a single integer: the index in the array of the element that matches the query.
If the query is not present in the array, output -1 instead.

Assumptions: $1 < n \leq 1{,}000{,}000$
$0 \leq q \leq 1{,}000{,}000$
$0 \leq a[0] < a[1] < a[2] < \ldots < a[n - 1] \leq 1{,}000{,}000$

Sample Input #1:
```
3 2
2 4 6
```

Sample Output #1:
```
0
```

Sample Input #2:
```
3 7
2 4 6
```

Sample Output #2:
```
-1
```

Overview: Find the largest angle between two cuts of a pizza

Description: You have a new pizza cutting machine with an interesting algorithm for slicing the pizza. It starts by making an initial cut from the center of the pizza to the edge. The machine then turns the pizza *d* degrees before making another cut. This process repeats until the machine has made *n* cuts. After making the *n* cuts there are some number of slices in the pizza. Can you figure out the angle of the largest slice in the pizza?

Filename: bug6.{java, cpp, c, cc, py}

Input: The input will have one line containing two space separated integers: *n d*

Output: Output a single integer: the largest angular size of a slice (i.e. the largest angle between two cuts) in degrees

Assumptions: $1 \le n \le 100$
$0 \le d < 360$

Sample Input #1:
```
3 90
```

Sample Output #1:
```
180
```

Sample Input #2:
```
12 70
```

Sample Output #2:
```
60
```

Overview: Find the minimum weight vertex cover in a tree

Description: You will be given a tree with *n* nodes, each of which has an associated weight. You want to pick a subset of the nodes with minimum weight such that each of the *n-1* edges has an endpoint in the subset. Such a subset is called the minimum weight vertex cover. Output the weight of a minimum weight vertex cover.

Filename: bug7.{java, cpp, c, cc, py}

Input: The first line will contain a single integer *n.*
The second line will contain *n* integers representing the weights of the nodes. The integer at index *i* will represent the weight w_i of node *i* ($0 \le i < n$).
The next *n-1* lines will contain two space separated integers *i j* indicating that an edge exists between node *i* and node *j* ($0 \le i, j \le n$).

Output: Output one integer representing the weight of the minimum weight vertex cover.

Assumptions: $1 \le n \le 1{,}000$
$0 \le w_i \le 1{,}000$
The given graph is a tree.

Sample Input #1:
```
1
1
```

Sample Output #1:
```
0
```

Sample Input #2:
```
4
1 2 3 4
0 1
0 2
2 3
```

Sample Output #2:
```
4
```

Overview: Manage a collection of numbers

Description: You are in charge of managing a collection of numbers. The collection supports two operations: *insert* and *delete*. On an *insert*, the collection stores the provided number. On a *delete*, the collection removes the largest stored number less than or equal to the provided number. To make your life easier, your client has told you that the numbers in an operation will be positive integers no larger than *n*.

Filename: bug8.{java, cpp, c, cc, py}

Input: The first line contains a single integer *n*
The second line contains a single integer *o* representing the number of operations
The next *o* lines consist of the operations. An *insert* operation will be specified by the character 'i' followed by an integer *x*. A *delete* operation will be specified by the character 'd' followed by an integer *x*. The character and integer are space-separated.

Output: Output one line for each *delete* operation containing a single integer representing the number removed from the collection.

Assumptions: $0 \le n \le 10{,}000{,}000$
$0 \le o \le 1{,}000$
$1 \le x \le n$
Every *delete* operation will be satisfiable

Sample Input #1:
```
1000
3
i 1
i 3
r 4
```

Sample Output #1:
```
3
```

Sample Input #2:
```
6
6
i 6
r 6
i 1
i 3
r 2
r 6
```

Sample Output #2:
```
6
1
3
```