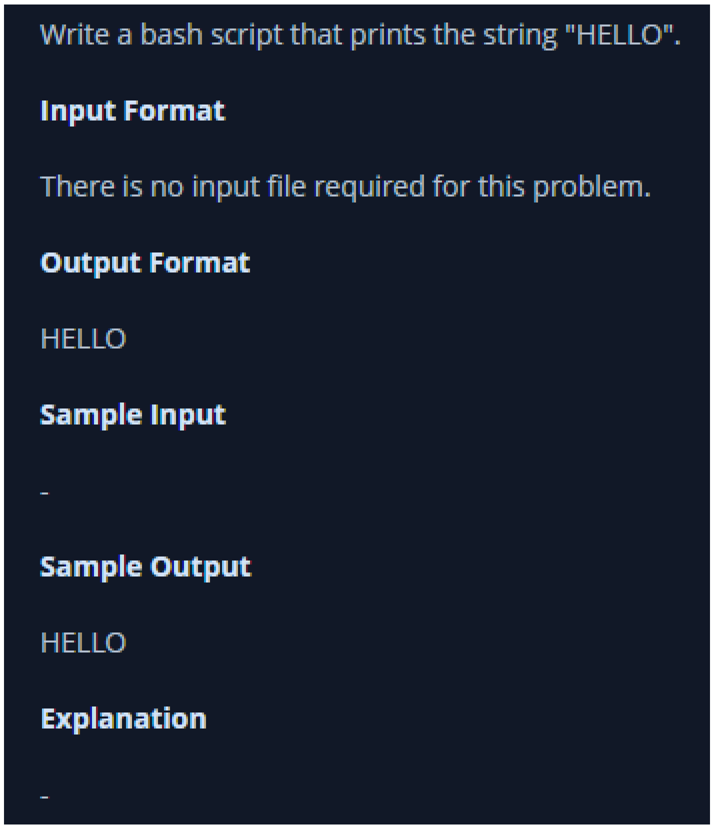

Write a bash script that prints the string "HELLO".

Input Format

There is no input file required for this problem.

Output Format

HELLO

Sample Input

-

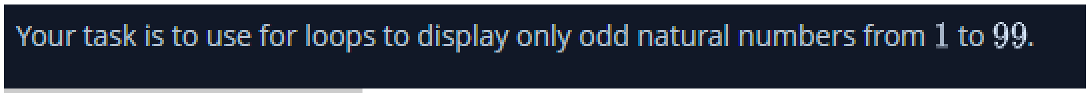
Sample Output

HELLO

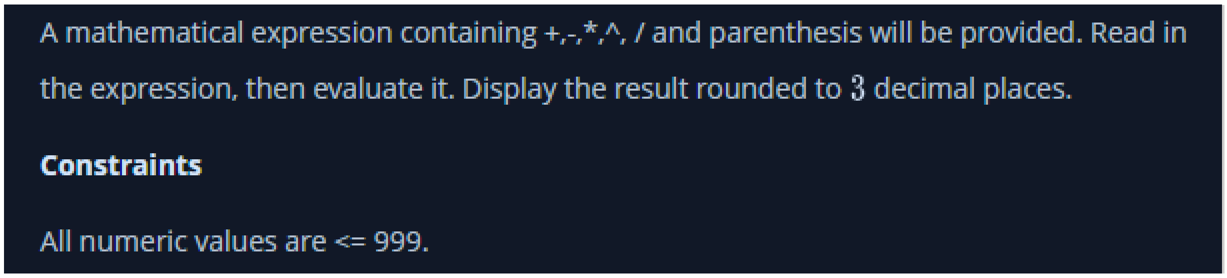
Explanation

-

```
#!/bin/bash
echo "HELLO"
```

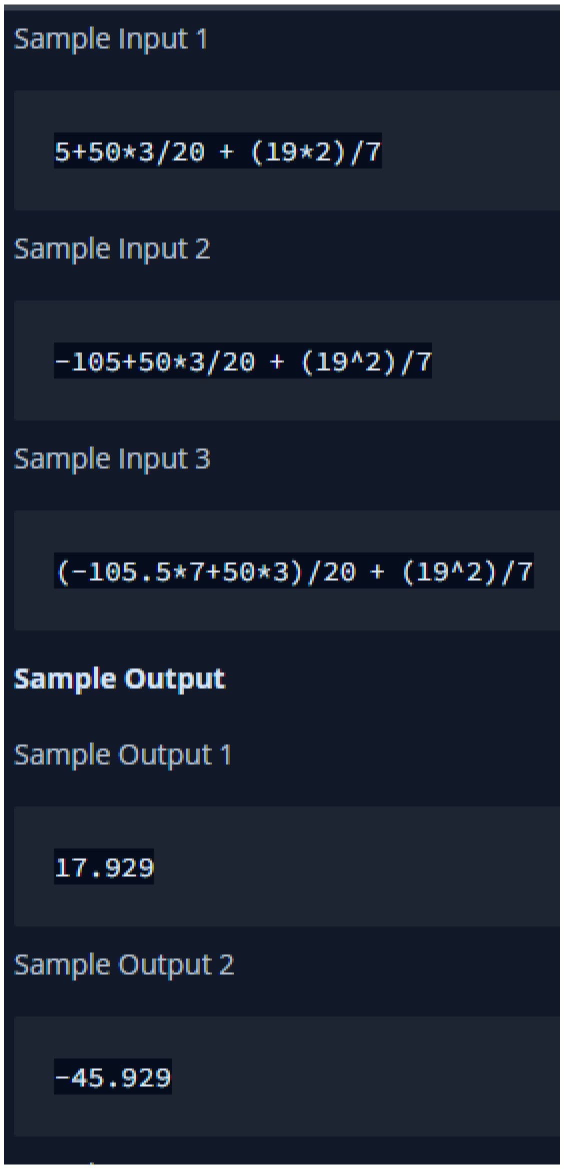

Your task is to use for loops to display only odd natural numbers from 1 to 99.

```
for ((i=1;i<=99;i=i+2))
do
echo "$i";
done
#we use double bracket bcz it wil treat int as string if we don't do so
```


A mathematical expression containing +,-,*,^, / and parenthesis will be provided. Read in the expression, then evaluate it. Display the result rounded to 3 decimal places.

Constraints

All numeric values are <= 999.


Sample Input 1

```
5+50*3/20 + (19*2)/7
```

Sample Input 2

```
-105+50*3/20 + (19^2)/7
```

Sample Input 3

```
(-105.5*7+50*3)/20 + (19^2)/7
```

Sample Output

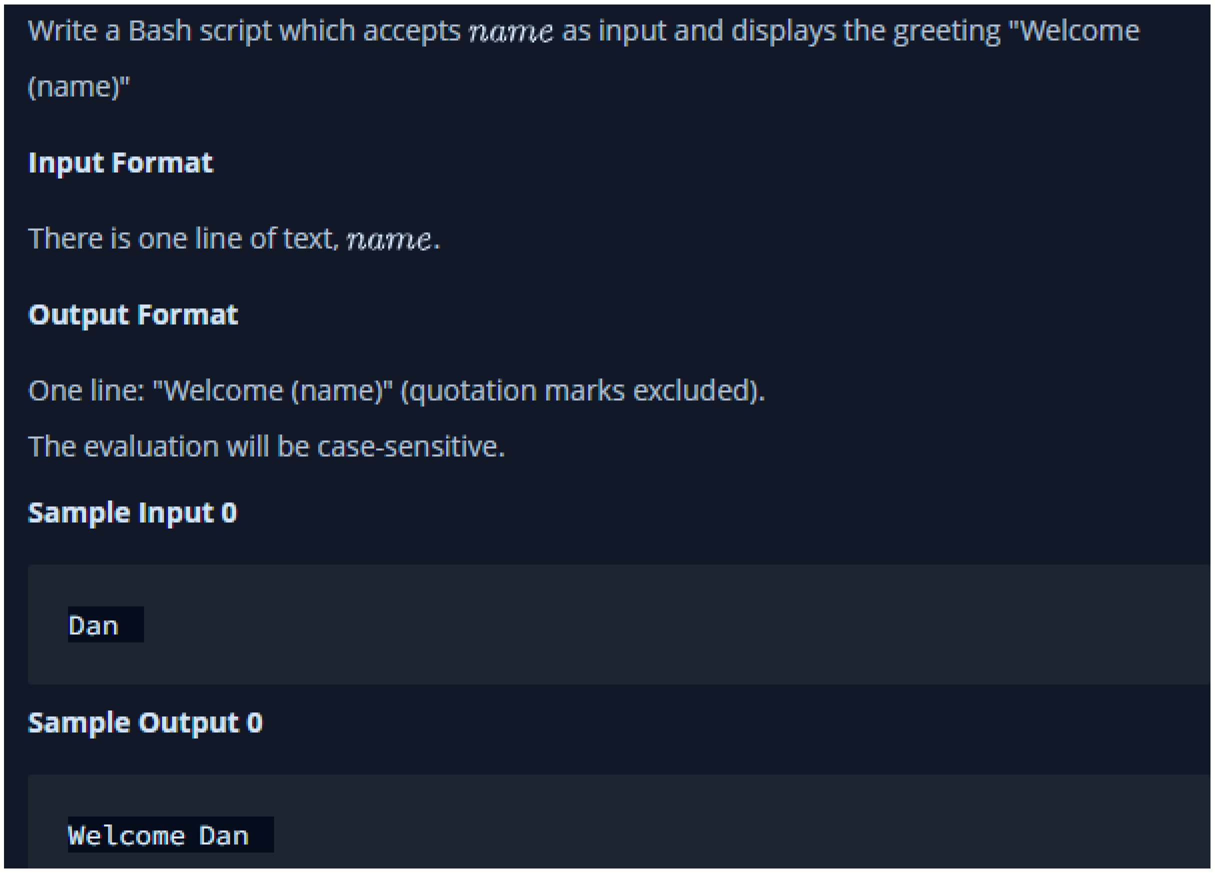
Sample Output 1

```
17.929
```

Sample Output 2

```
-45.929
```

```
read line;
printf "%.3f" $(echo "scale=4; $line" | bc);
printf "%.3f"  - this will rounf decimal values eg 12.26686857 to 12.268
```


Write a Bash script which accepts *name* as input and displays the greeting "Welcome (name)"

Input Format

There is one line of text, *name*.

Output Format

One line: "Welcome (name)" (quotation marks excluded).

The evaluation will be case-sensitive.

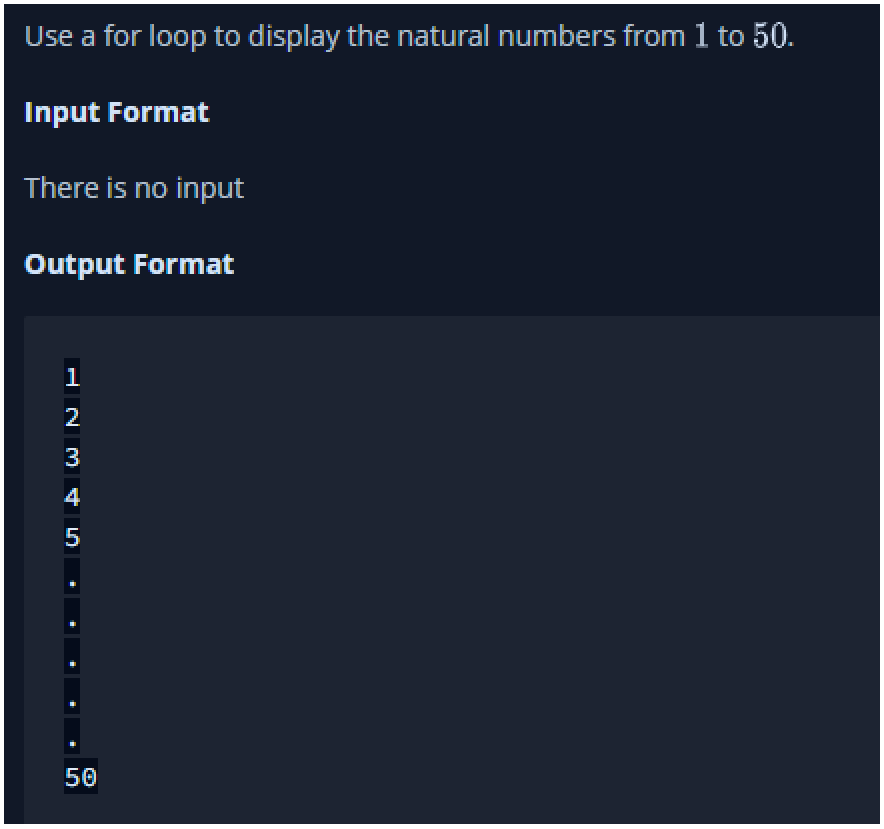
Sample Input 0

```
Dan
```

Sample Output 0

```
Welcome Dan
```

```
#!/bin/bash
read name
echo "Welcome $name"
```



Use a for loop to display the natural numbers from 1 to 50.

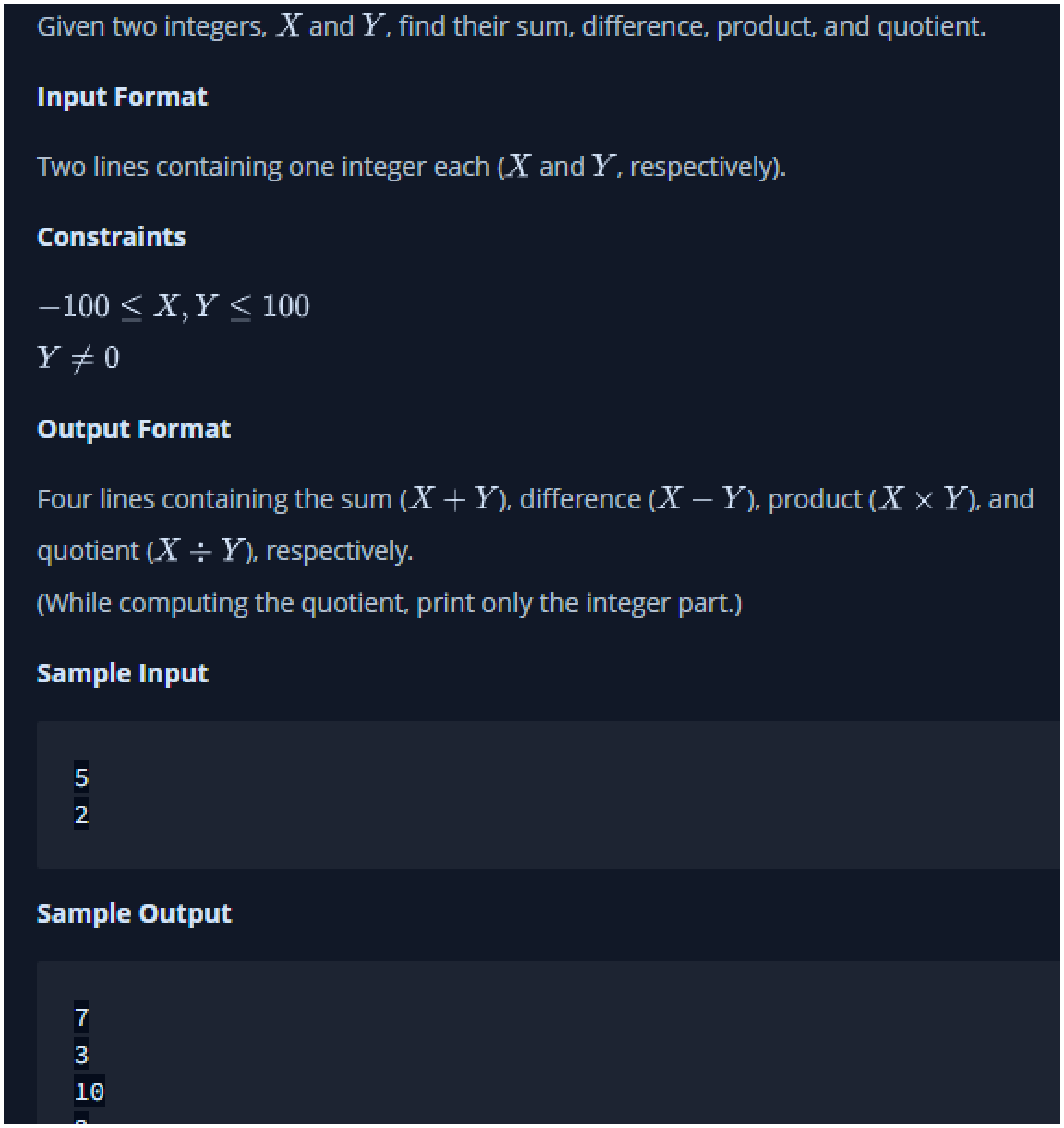
Input Format

There is no input

Output Format

```
1
2
3
4
5
.
.
.
.
.
50
```

```bash
#!/bin/bash
for ((i=1;i<=50;i=i+1))
do
echo $i
done
```


Given two integers, X and Y, find their sum, difference, product, and quotient.

Input Format

Two lines containing one integer each (X and Y, respectively).

Constraints

$-100 \leq X, Y \leq 100$
$Y \neq 0$

Output Format

Four lines containing the sum ($X + Y$), difference ($X - Y$), product ($X \times Y$), and quotient ($X \div Y$), respectively.
(While computing the quotient, print only the integer part.)

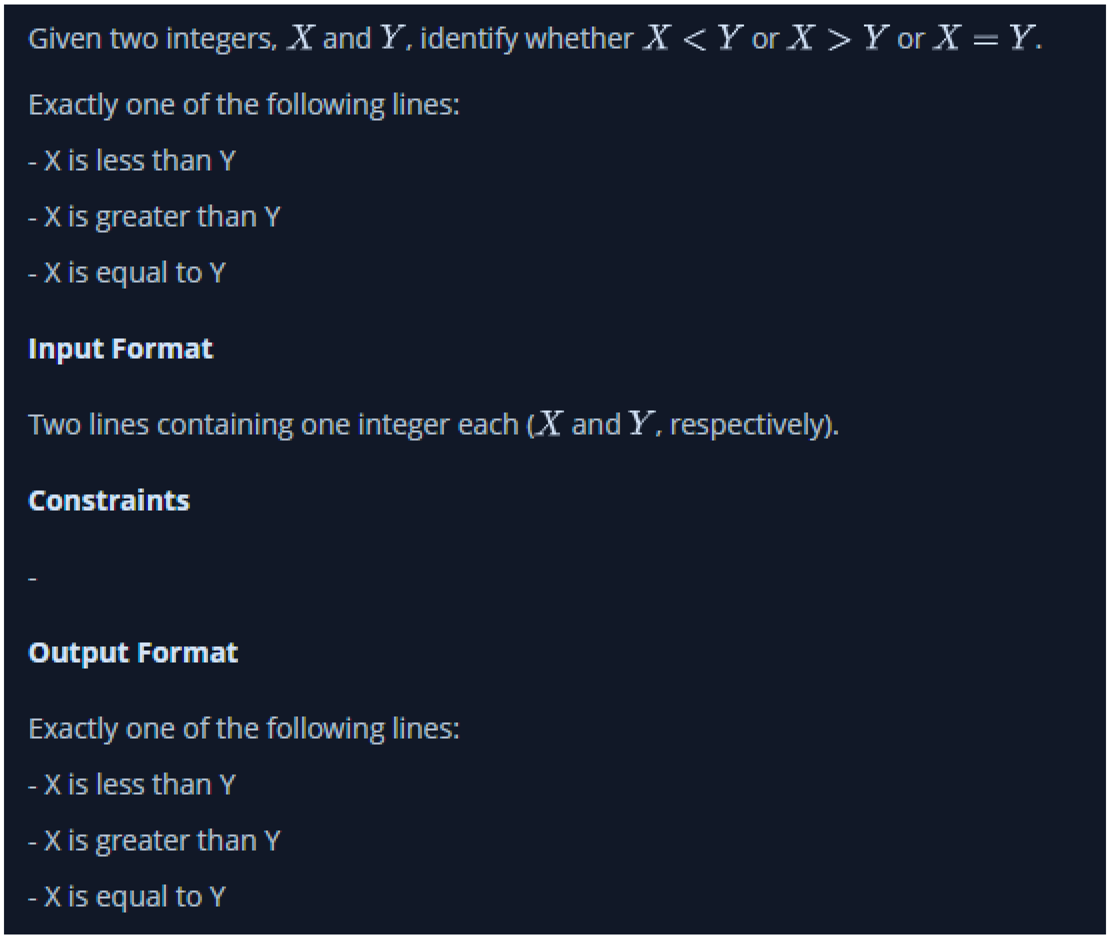
Sample Input

```
5
2
```

Sample Output

```
7
3
10
```

```bash
#!/bin/bash
read x
read y
echo $((x+y));
echo $((x-y));
echo $((x*y));
echo $((x/y));
```


Given two integers, X and Y, identify whether $X < Y$ or $X > Y$ or $X = Y$.

Exactly one of the following lines:

- X is less than Y

- X is greater than Y

- X is equal to Y

Input Format

Two lines containing one integer each (X and Y, respectively).

Constraints

-

Output Format

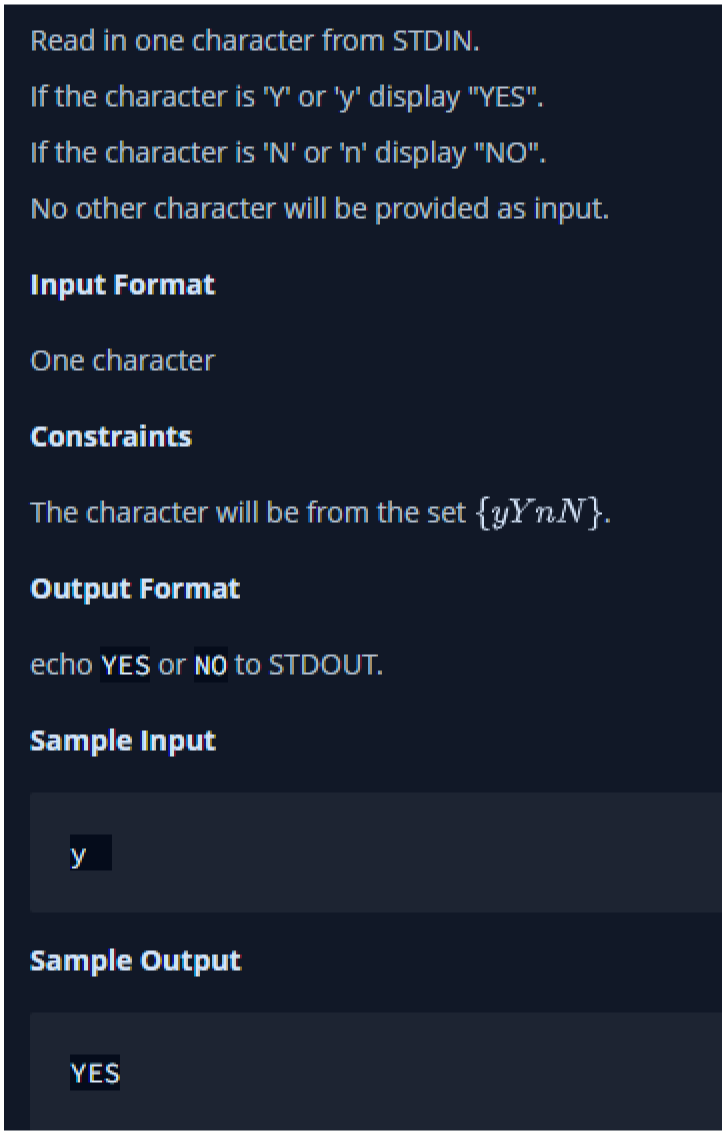
Exactly one of the following lines:

- X is less than Y

- X is greater than Y

- X is equal to Y

```
#!/bin/bash
read x
read y
if (( $x < $y )); then
    echo "X is less than Y"
elif (( $x > $y )); then
    echo "X is greater than Y"
else
    echo "X is equal to Y"
fi
```


Read in one character from STDIN.

If the character is 'Y' or 'y' display "YES".

If the character is 'N' or 'n' display "NO".

No other character will be provided as input.

Input Format

One character

Constraints

The character will be from the set $\{yYnN\}$.

Output Format

echo `YES` or `NO` to STDOUT.

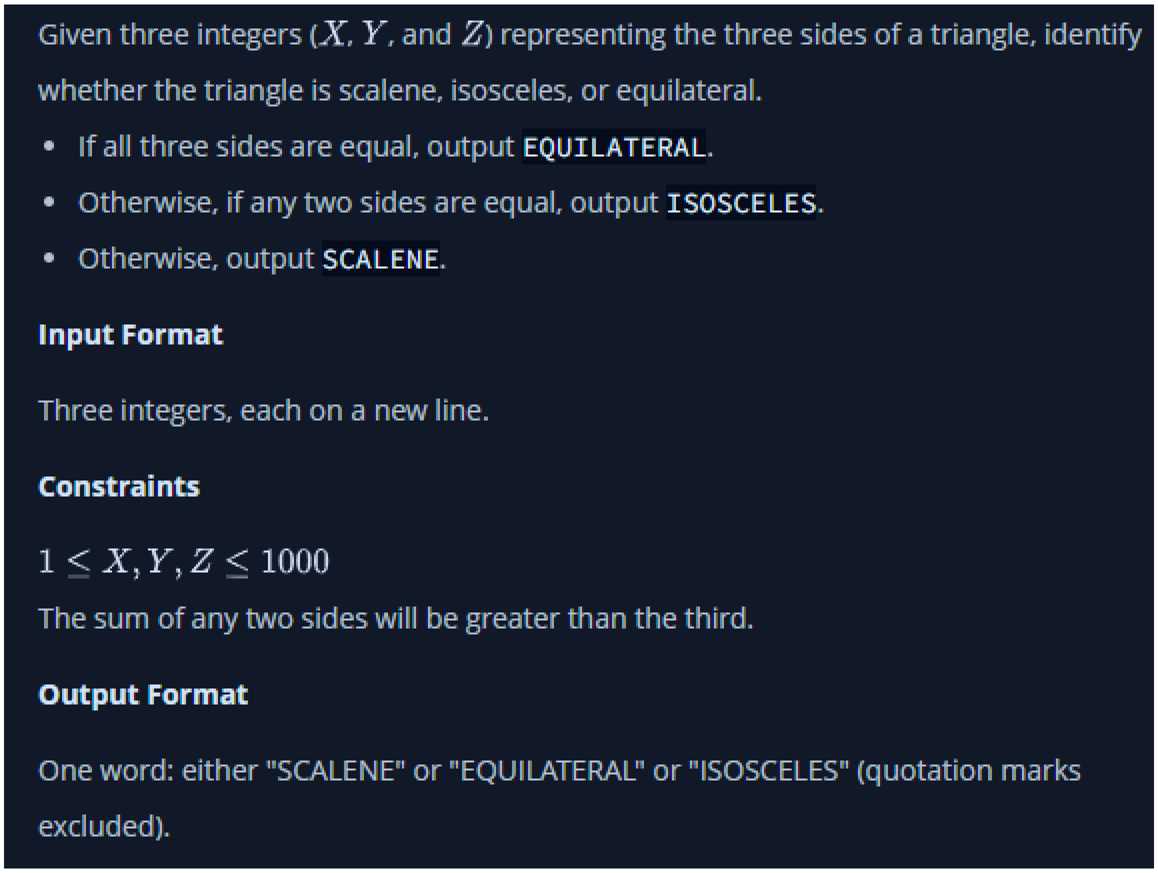
Sample Input

```
y
```

Sample Output

```
YES
```

```
#!/bin/bash
read ch;
if [ $ch == 'y' ] || [ $ch == 'Y' ]
then
echo "YES";
else [ $ch == 'n' ] || [ $ch == 'N' ]
echo "NO";
fi
```


Given three integers (X, Y, and Z) representing the three sides of a triangle, identify whether the triangle is scalene, isosceles, or equilateral.

- If all three sides are equal, output `EQUILATERAL`.
- Otherwise, if any two sides are equal, output `ISOSCELES`.
- Otherwise, output `SCALENE`.

Input Format

Three integers, each on a new line.

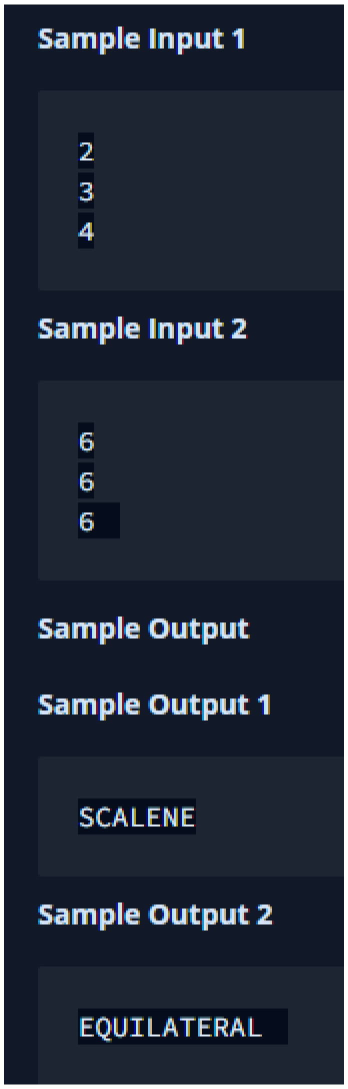
Constraints

$1 \leq X, Y, Z \leq 1000$

The sum of any two sides will be greater than the third.

Output Format

One word: either "SCALENE" or "EQUILATERAL" or "ISOSCELES" (quotation marks excluded).


Sample Input 1

```
2
3
4
```

Sample Input 2

```
6
6
6
```

Sample Output

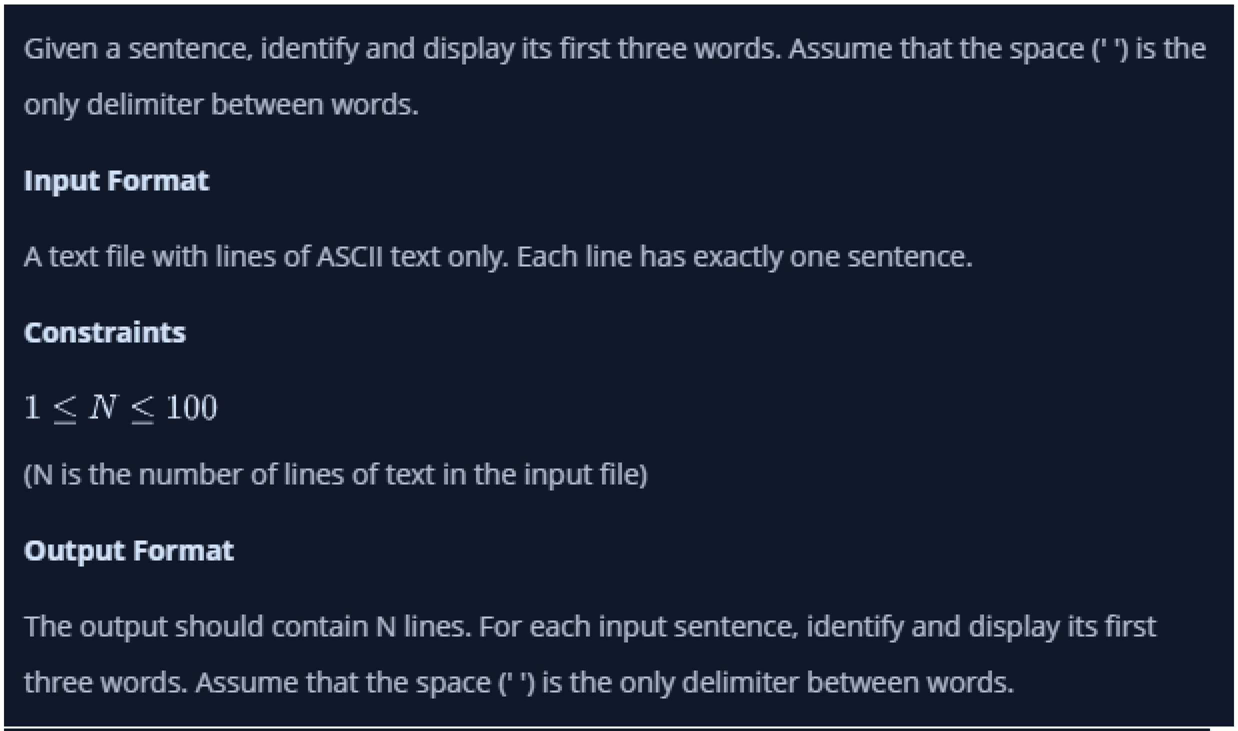
Sample Output 1

```
SCALENE
```

Sample Output 2

```
EQUILATERAL
```

```bash
#!/bin/bash
read x;
read y;
read z;
if [ $x -eq $y ] && [ $y -eq $z ]
then
echo "EQUILATERAL";
elif [ $x -ne $y ] && [ $x -ne $z ] && [ $y -ne $z ]
then
echo "SCALENE";
else
echo "ISOSCELES";
fi
```



Given a sentence, identify and display its first three words. Assume that the space (' ') is the only delimiter between words.

Input Format

A text file with lines of ASCII text only. Each line has exactly one sentence.

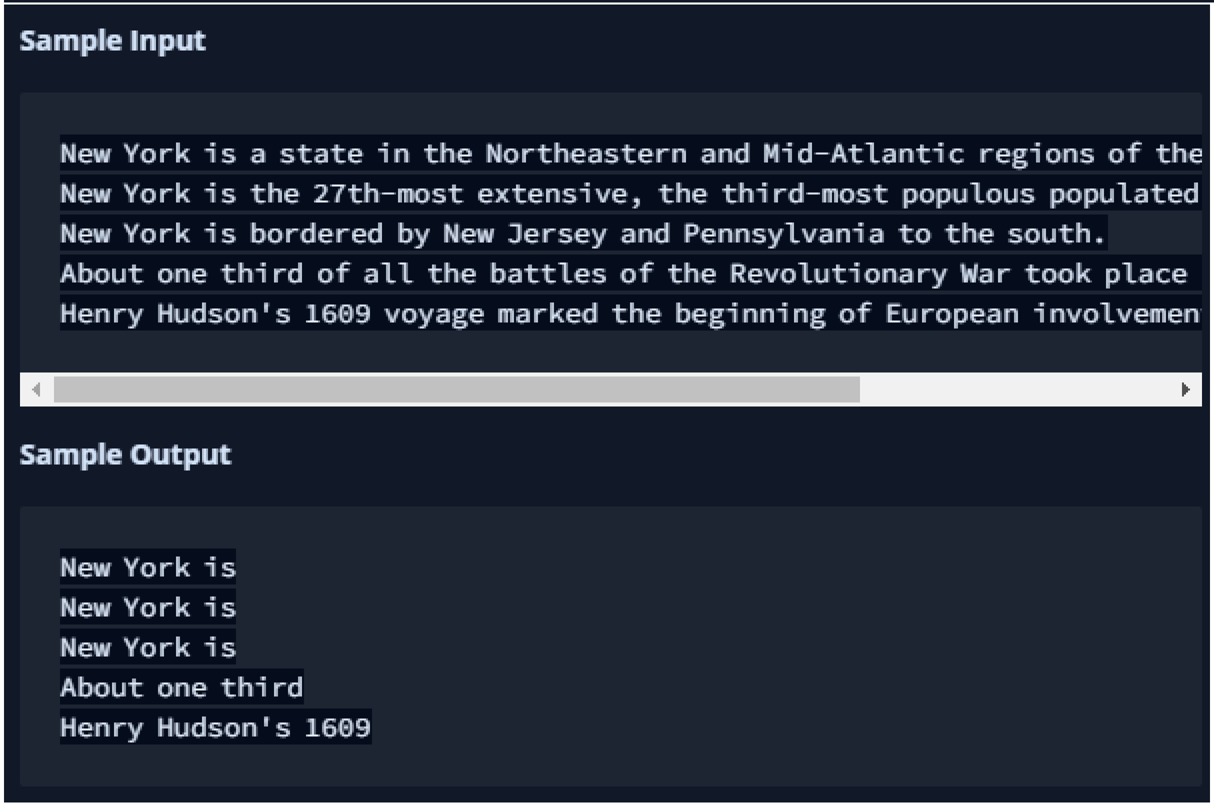
Constraints

$1 \leq N \leq 100$

(N is the number of lines of text in the input file)

Output Format

The output should contain N lines. For each input sentence, identify and display its first three words. Assume that the space (' ') is the only delimiter between words.



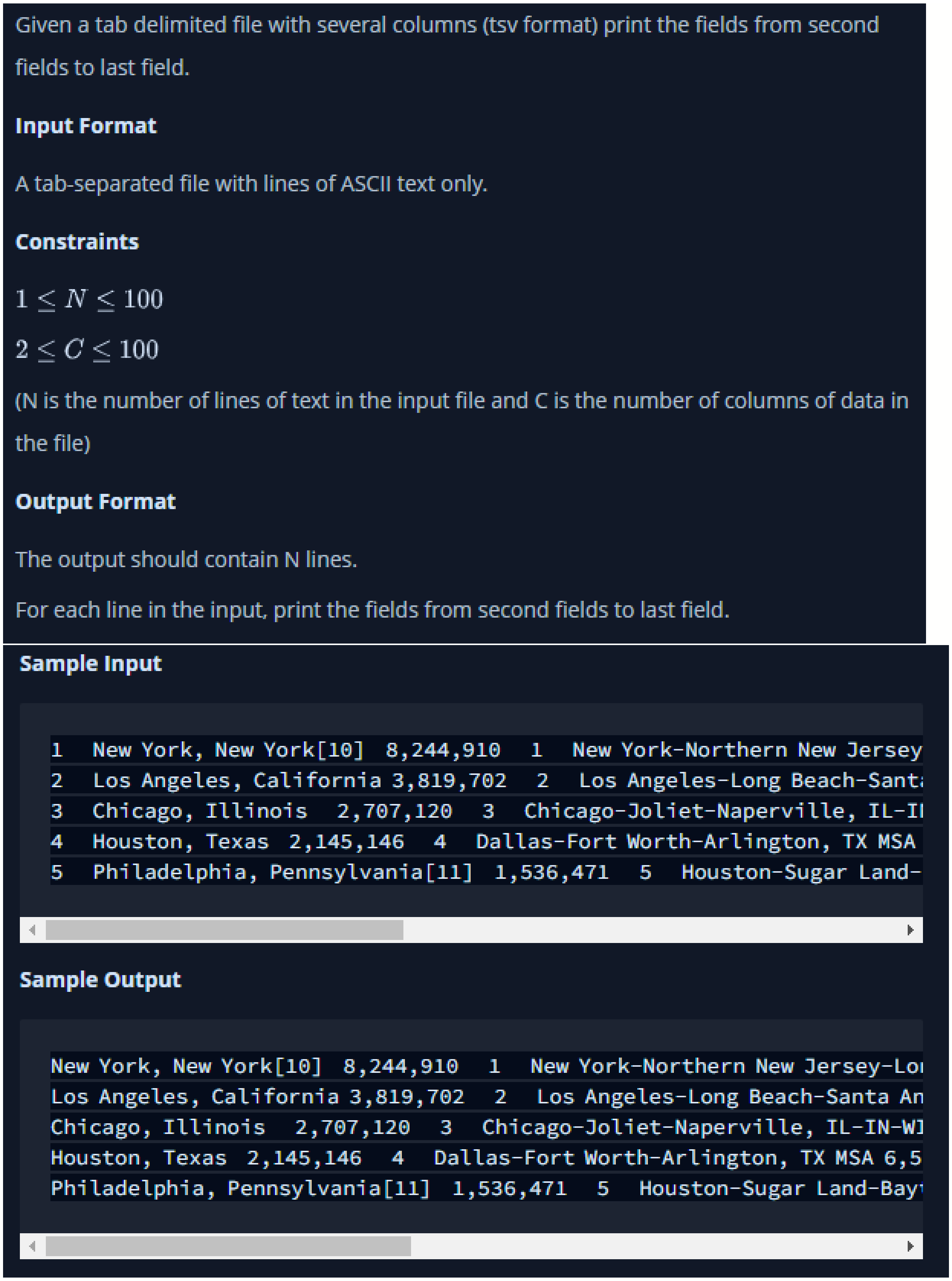
Sample Input

```
New York is a state in the Northeastern and Mid-Atlantic regions of the
New York is the 27th-most extensive, the third-most populous populated
New York is bordered by New Jersey and Pennsylvania to the south.
About one third of all the battles of the Revolutionary War took place
Henry Hudson's 1609 voyage marked the beginning of European involvemen
```

Sample Output

```
New York is
New York is
New York is
About one third
Henry Hudson's 1609
```

```
cut -d " " -f 1-3
```


Given a tab delimited file with several columns (tsv format) print the fields from second fields to last field.

Input Format

A tab-separated file with lines of ASCII text only.

Constraints

$1 \leq N \leq 100$

$2 \leq C \leq 100$

(N is the number of lines of text in the input file and C is the number of columns of data in the file)

Output Format

The output should contain N lines.

For each line in the input, print the fields from second fields to last field.

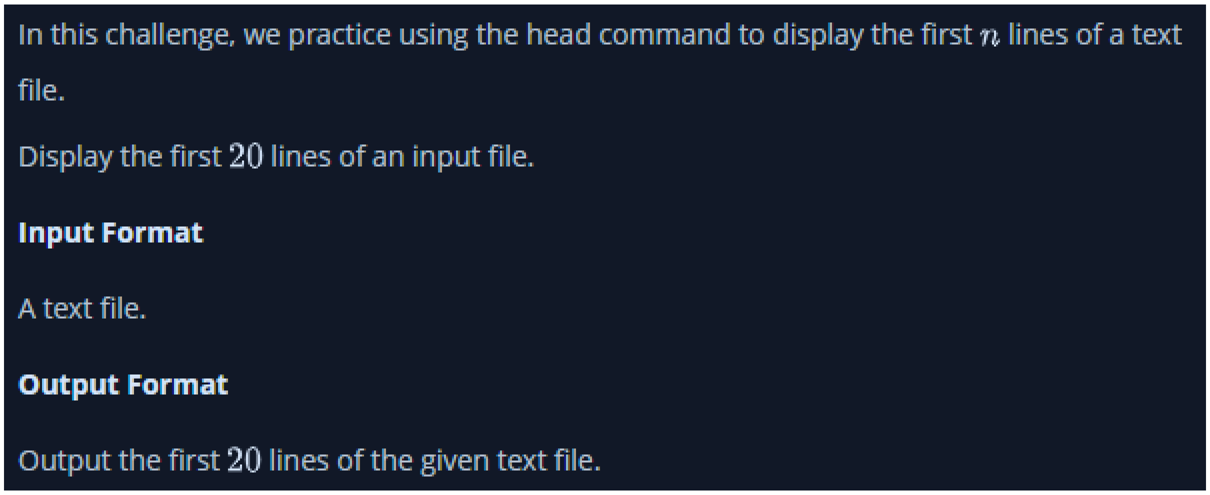
Sample Input

```
1	New York, New York[10]	8,244,910	1	New York-Northern New Jersey
2	Los Angeles, California	3,819,702	2	Los Angeles-Long Beach-Santa
3	Chicago, Illinois	2,707,120	3	Chicago-Joliet-Naperville, IL-II
4	Houston, Texas	2,145,146	4	Dallas-Fort Worth-Arlington, TX MSA
5	Philadelphia, Pennsylvania[11]	1,536,471	5	Houston-Sugar Land-
```

Sample Output

```
New York, New York[10]	8,244,910	1	New York-Northern New Jersey-Lo
Los Angeles, California	3,819,702	2	Los Angeles-Long Beach-Santa An
Chicago, Illinois	2,707,120	3	Chicago-Joliet-Naperville, IL-IN-WI
Houston, Texas	2,145,146	4	Dallas-Fort Worth-Arlington, TX MSA 6,5
Philadelphia, Pennsylvania[11]	1,536,471	5	Houston-Sugar Land-Bay
```

```
cut -d$'\t' -f 2-
```


In this challenge, we practice using the head command to display the first n lines of a text file.

Display the first 20 lines of an input file.

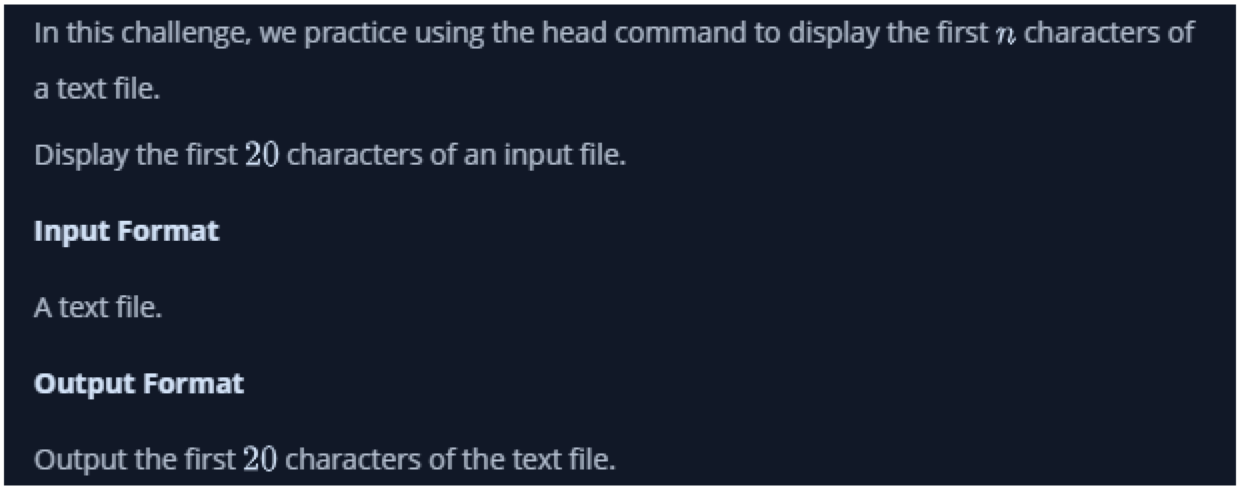
Input Format

A text file.

Output Format

Output the first 20 lines of the given text file.

```
head -20
```


In this challenge, we practice using the head command to display the first n characters of a text file.

Display the first 20 characters of an input file.

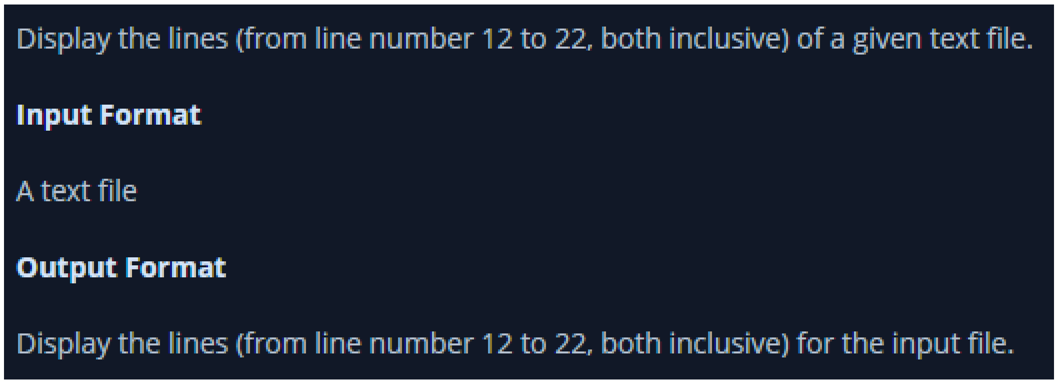
Input Format

A text file.

Output Format

Output the first 20 characters of the text file.

```
head -c 20
```


Display the lines (from line number 12 to 22, both inclusive) of a given text file.

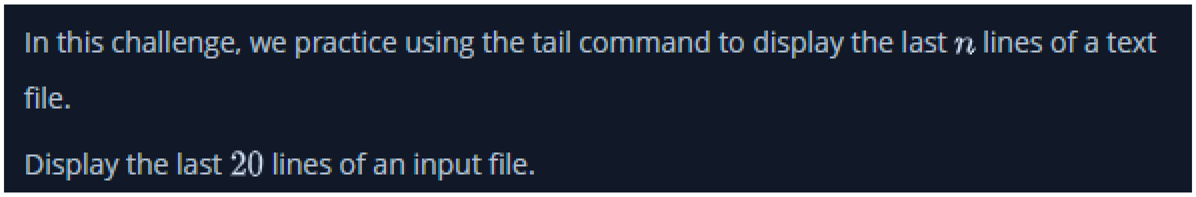
Input Format

A text file

Output Format

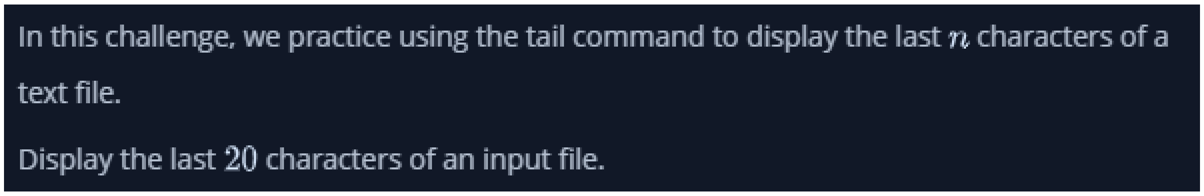
Display the lines (from line number 12 to 22, both inclusive) for the input file.

```
head -n 22 | tail -n 11
```


In this challenge, we practice using the tail command to display the last n lines of a text file.

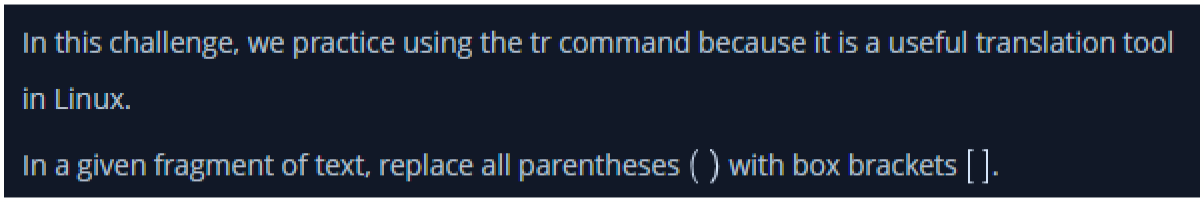
Display the last 20 lines of an input file.

```
tail -20
```


In this challenge, we practice using the tail command to display the last n characters of a text file.

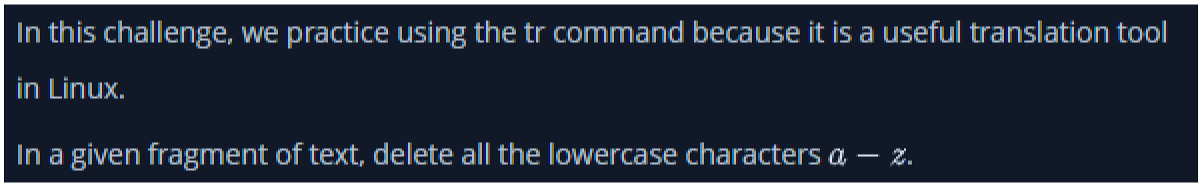
Display the last 20 characters of an input file.

```
tail -c 20
```


In this challenge, we practice using the tr command because it is a useful translation tool in Linux.

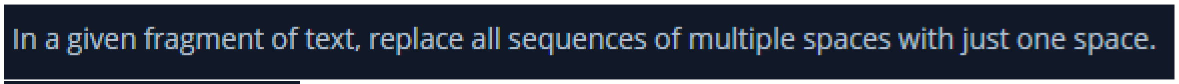
In a given fragment of text, replace all parentheses ( ) with box brackets [ ].

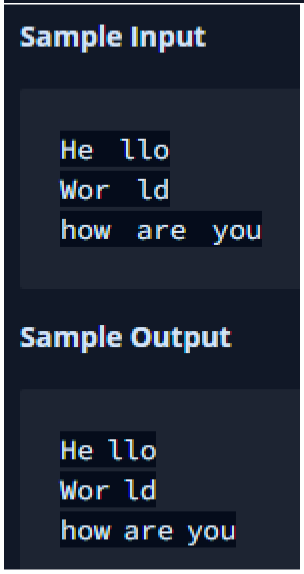
```
tr '()' '[]'
```


In this challenge, we practice using the tr command because it is a useful translation tool in Linux.

In a given fragment of text, delete all the lowercase characters $a - z$.

```
tr -d 'a-z'
```


In a given fragment of text, replace all sequences of multiple spaces with just one space.

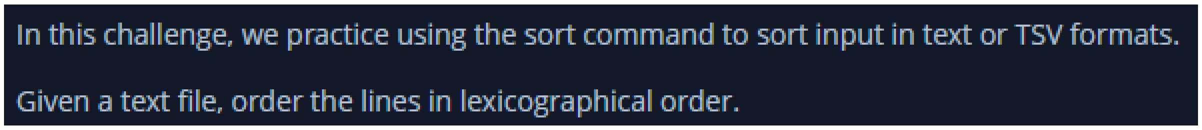

Sample Input

```
He  llo
Wor  ld
how  are  you
```

Sample Output

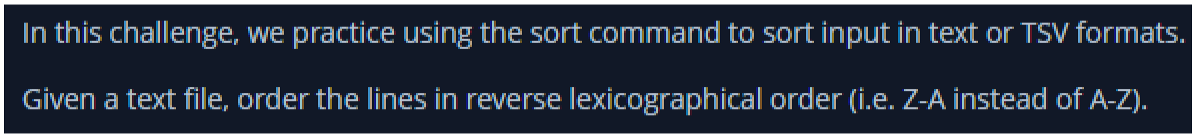
```
He llo
Wor ld
how are you
```

```
tr -s ' '
```


In this challenge, we practice using the sort command to sort input in text or TSV formats.

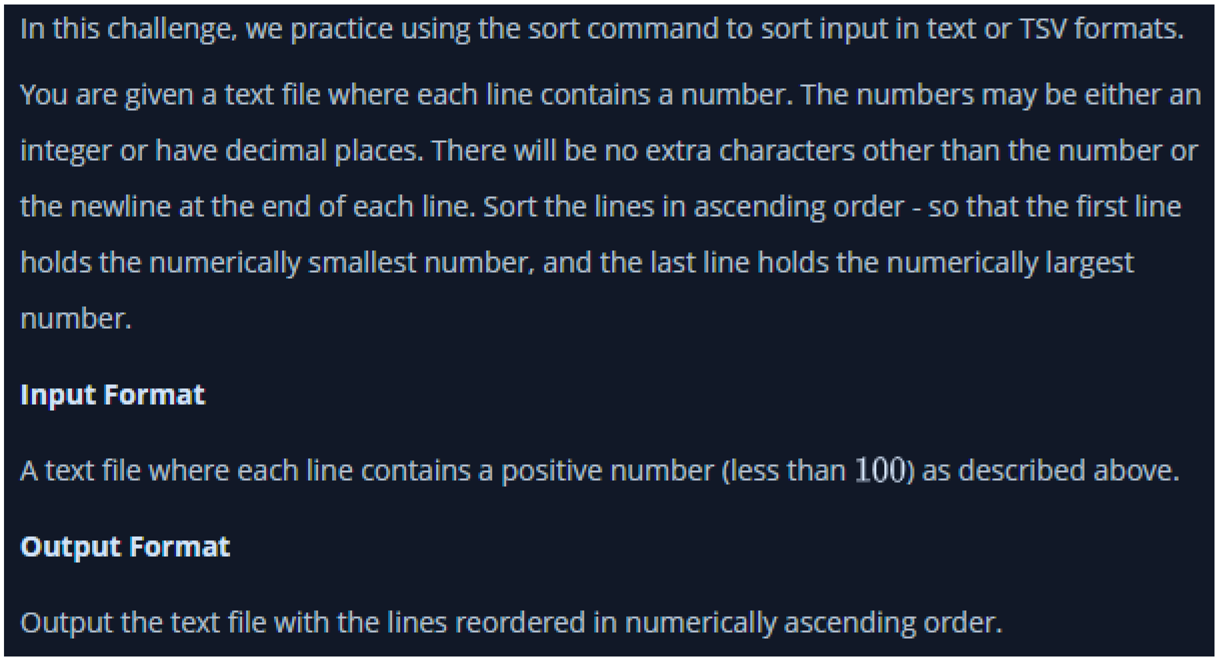
Given a text file, order the lines in lexicographical order.

```
sort
```


In this challenge, we practice using the sort command to sort input in text or TSV formats.

Given a text file, order the lines in reverse lexicographical order (i.e. Z-A instead of A-Z).

```
sort -r
```


In this challenge, we practice using the sort command to sort input in text or TSV formats.

You are given a text file where each line contains a number. The numbers may be either an integer or have decimal places. There will be no extra characters other than the number or the newline at the end of each line. Sort the lines in ascending order - so that the first line holds the numerically smallest number, and the last line holds the numerically largest number.

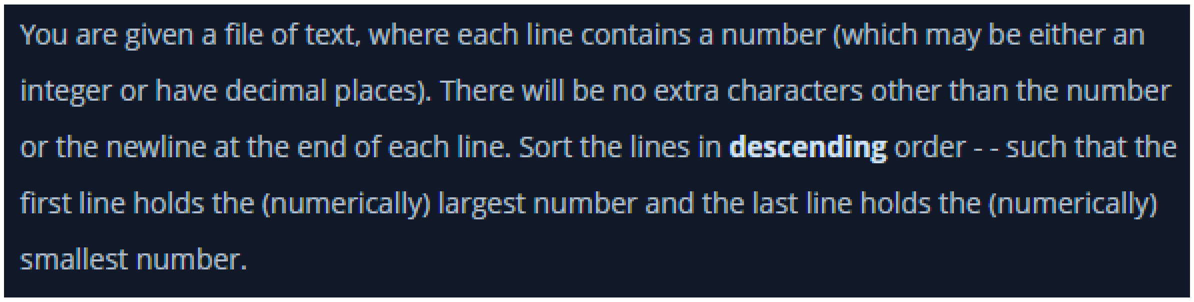
Input Format

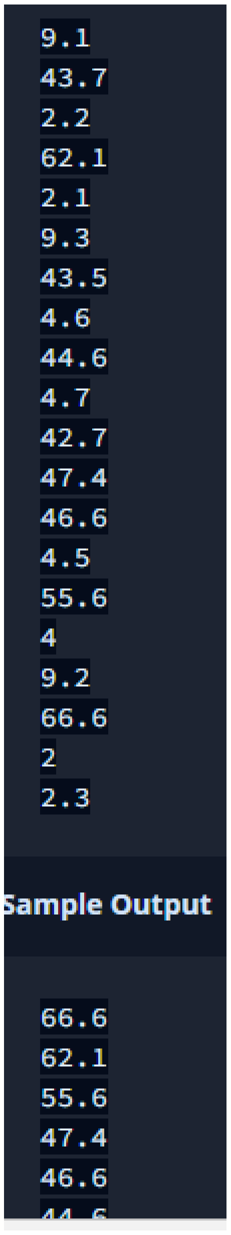
A text file where each line contains a positive number (less than 100) as described above.

Output Format

Output the text file with the lines reordered in numerically ascending order.

```
sort -n
```


You are given a file of text, where each line contains a number (which may be either an integer or have decimal places). There will be no extra characters other than the number or the newline at the end of each line. Sort the lines in **descending** order - - such that the first line holds the (numerically) largest number and the last line holds the (numerically) smallest number.

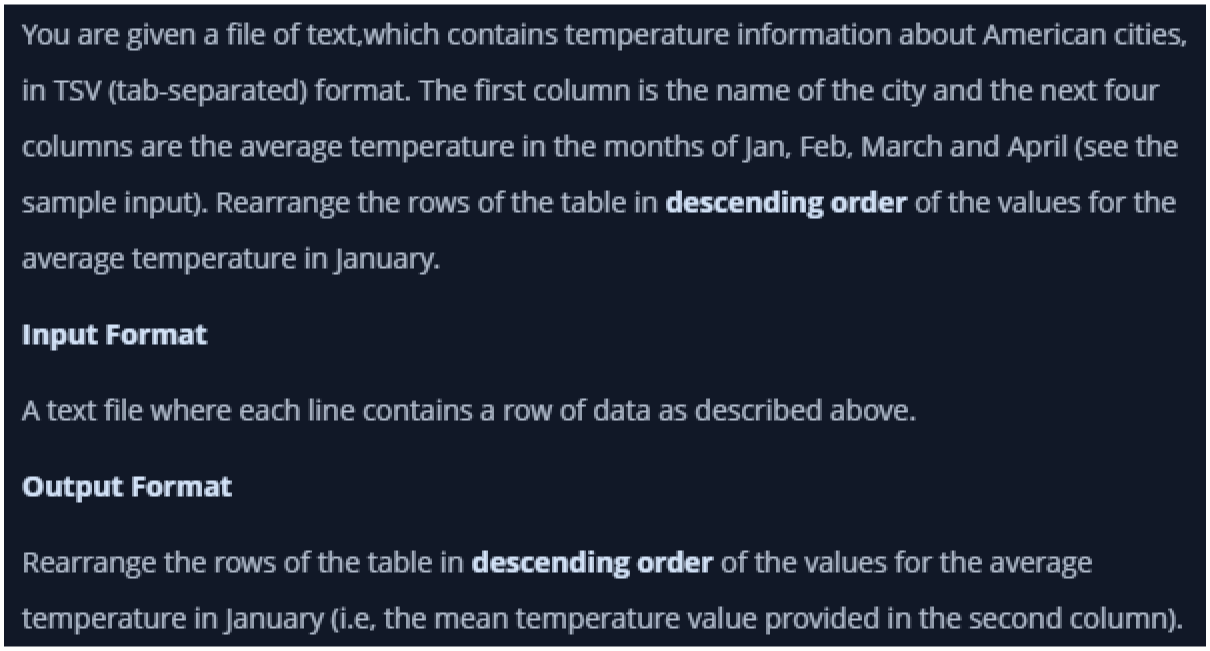


```
9.1
43.7
2.2
62.1
2.1
9.3
43.5
4.6
44.6
4.7
42.7
47.4
46.6
4.5
55.6
4
9.2
66.6
2
2.3
```

Sample Output

```
66.6
62.1
55.6
47.4
46.6
```

```
sort -n -r
```


You are given a file of text,which contains temperature information about American cities, in TSV (tab-separated) format. The first column is the name of the city and the next four columns are the average temperature in the months of Jan, Feb, March and April (see the sample input). Rearrange the rows of the table in **descending order** of the values for the average temperature in January.

Input Format

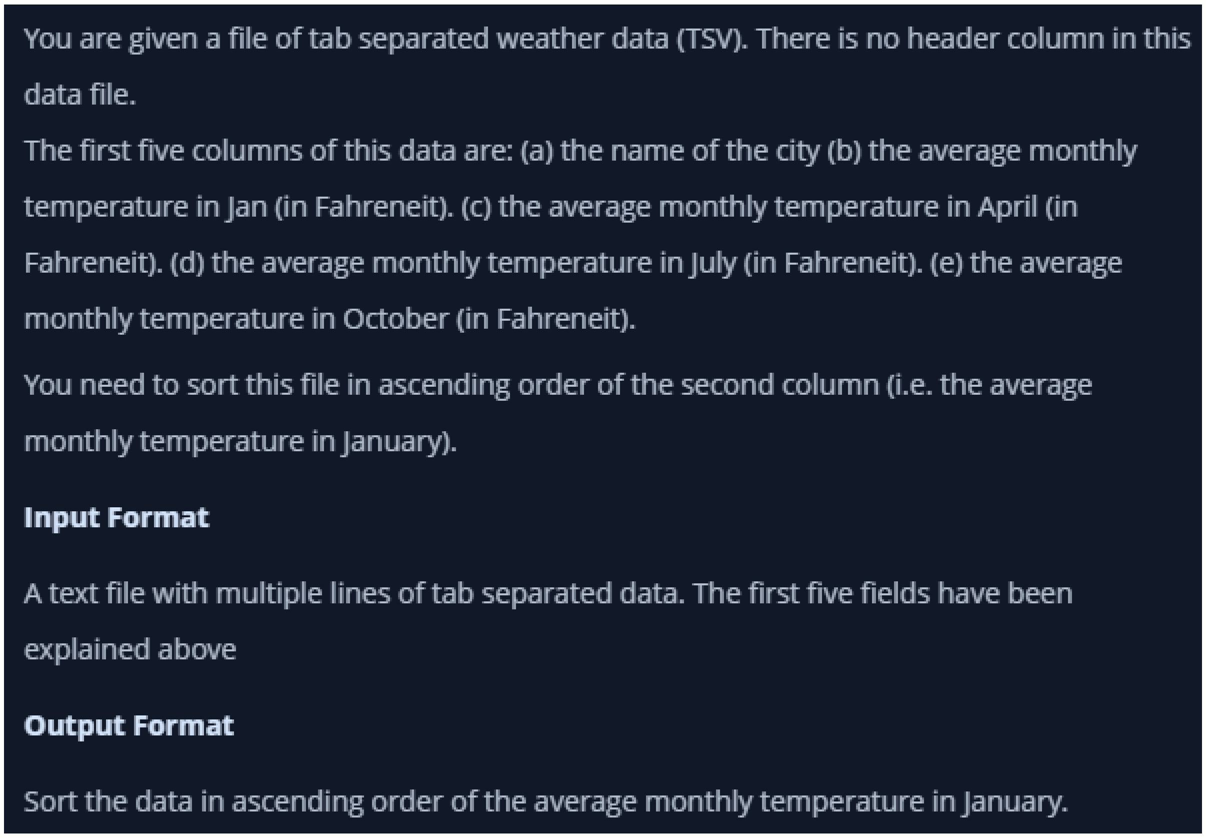
A text file where each line contains a row of data as described above.

Output Format

Rearrange the rows of the table in **descending order** of the values for the average temperature in January (i.e, the mean temperature value provided in the second column).

With the "-k" option, the sort command can be used to sort flat file databases.
The -t option can be used to change the separator.
sort command with 'r' option gives you to sort the contents in reverse order

```
sort -t$'\t' -nr -k2
```


You are given a file of tab separated weather data (TSV). There is no header column in this data file.
The first five columns of this data are: (a) the name of the city (b) the average monthly temperature in Jan (in Fahreneit). (c) the average monthly temperature in April (in Fahreneit). (d) the average monthly temperature in July (in Fahreneit). (e) the average monthly temperature in October (in Fahreneit).

You need to sort this file in ascending order of the second column (i.e. the average monthly temperature in January).

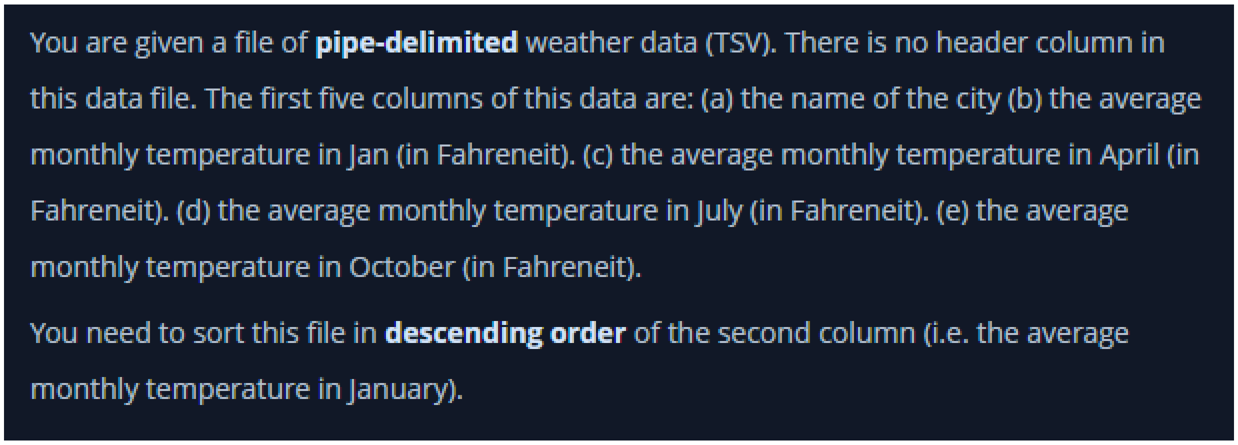
Input Format

A text file with multiple lines of tab separated data. The first five fields have been explained above

Output Format

Sort the data in ascending order of the average monthly temperature in January.

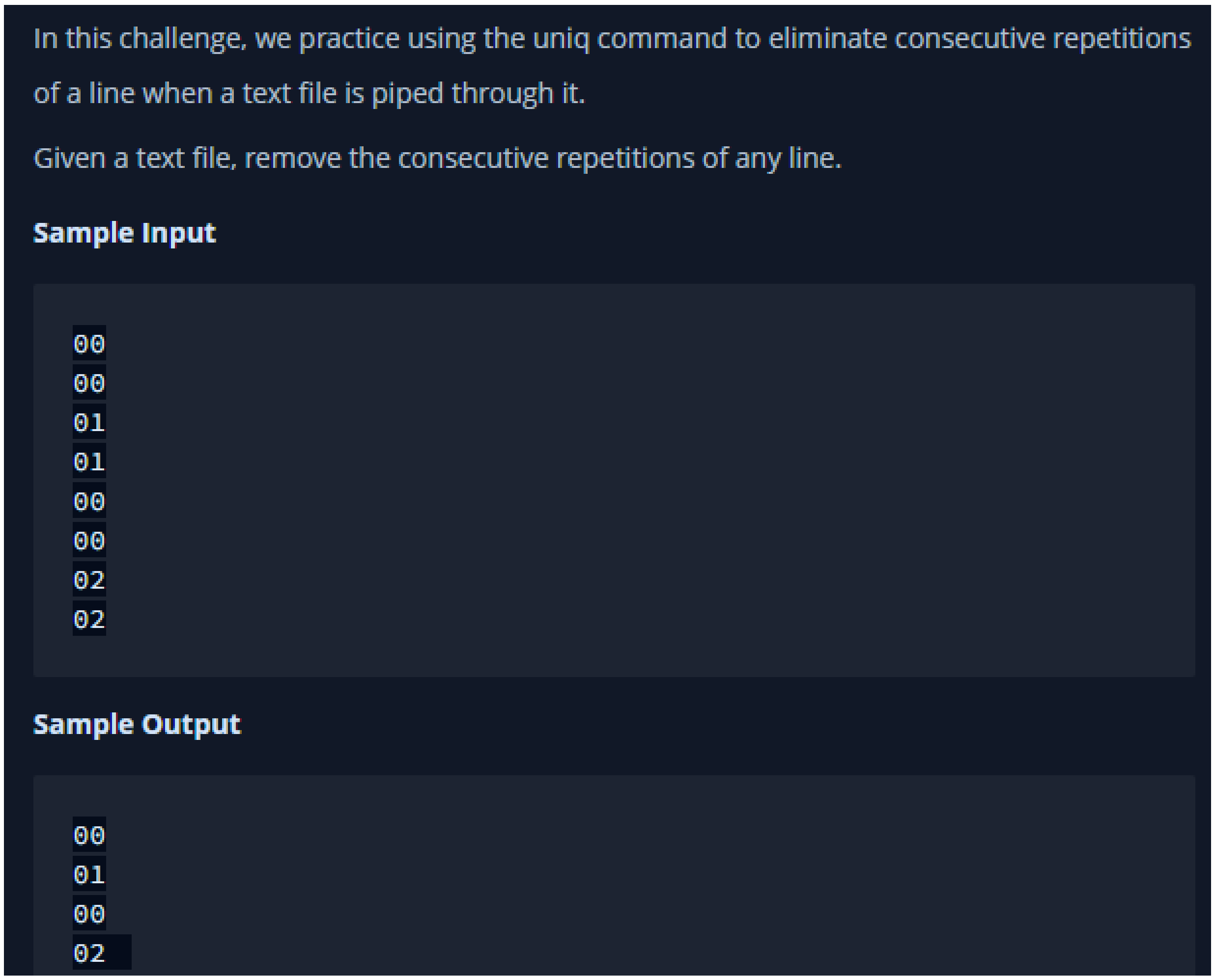
```
sort -t$'\t' -n -k2
```



You are given a file of **pipe-delimited** weather data (TSV). There is no header column in this data file. The first five columns of this data are: (a) the name of the city (b) the average monthly temperature in Jan (in Fahreneit). (c) the average monthly temperature in April (in Fahreneit). (d) the average monthly temperature in July (in Fahreneit). (e) the average monthly temperature in October (in Fahreneit).

You need to sort this file in **descending order** of the second column (i.e. the average monthly temperature in January).

```
sort -t$'|' -nr -k2
```



In this challenge, we practice using the uniq command to eliminate consecutive repetitions of a line when a text file is piped through it.

Given a text file, remove the consecutive repetitions of any line.

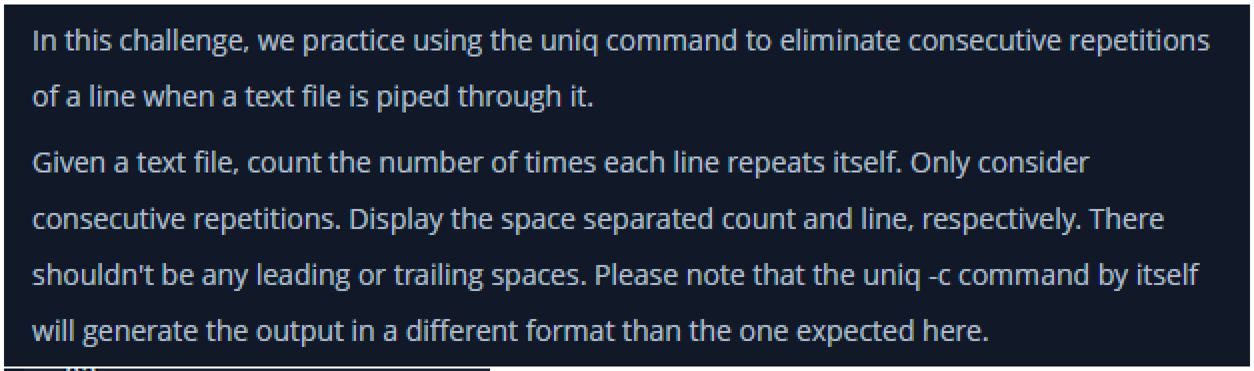
Sample Input

```
00
00
01
01
00
00
02
02
```

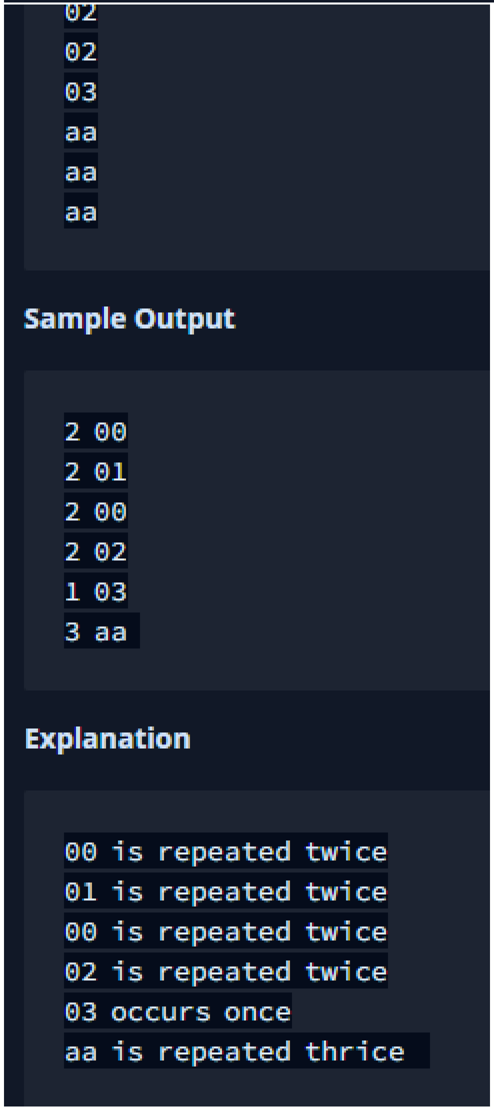
Sample Output

```
00
01
00
02
```

```
uniq
```


In this challenge, we practice using the uniq command to eliminate consecutive repetitions of a line when a text file is piped through it.

Given a text file, count the number of times each line repeats itself. Only consider consecutive repetitions. Display the space separated count and line, respectively. There shouldn't be any leading or trailing spaces. Please note that the uniq -c command by itself will generate the output in a different format than the one expected here.


```
02
02
03
aa
aa
aa
```

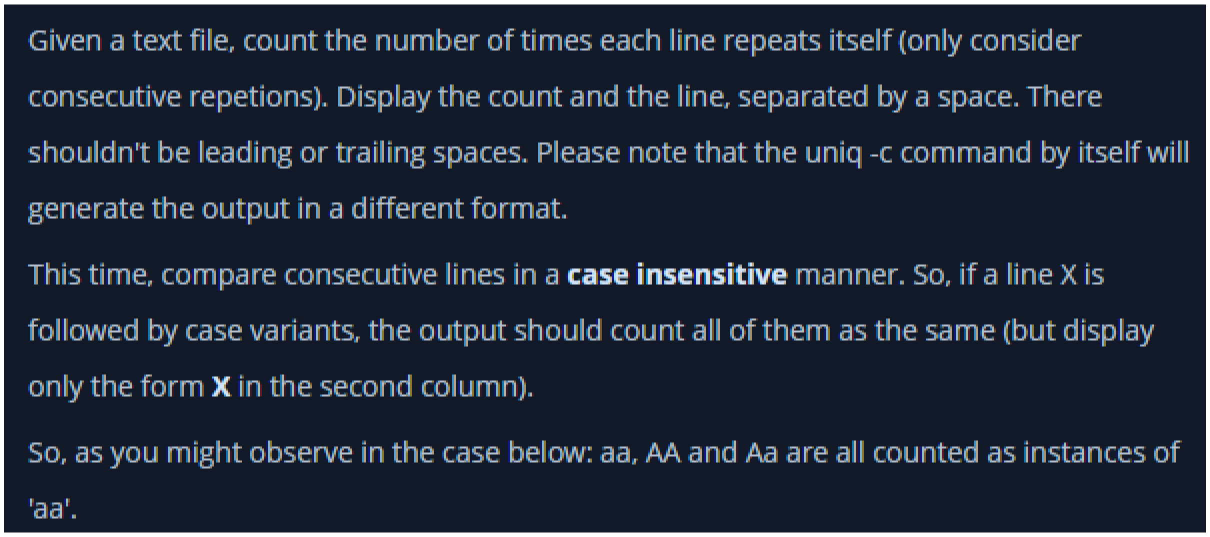
Sample Output

```
2 00
2 01
2 00
2 02
1 03
3 aa
```

Explanation

```
00 is repeated twice
01 is repeated twice
00 is repeated twice
02 is repeated twice
03 occurs once
aa is repeated thrice
```

```
uniq -c | cut -c 7-
```



Given a text file, count the number of times each line repeats itself (only consider consecutive repetions). Display the count and the line, separated by a space. There shouldn't be leading or trailing spaces. Please note that the uniq -c command by itself will generate the output in a different format.

This time, compare consecutive lines in a **case insensitive** manner. So, if a line X is followed by case variants, the output should count all of them as the same (but display only the form **X** in the second column).

So, as you might observe in the case below: aa, AA and Aa are all counted as instances of 'aa'.

```
uniq -c -i | cut -c 7-
```

Given a text file, display only those lines which are not followed or preceded by identical replications.

Sample Input

```
A00
a00
01
01
00
00
02
02
A00
03
aa
aa
aa
```

Sample Output

```
A00
a00
A00
03
```

Explanation

The comparison is case sensitive, so the first instance of "A00" and "a00" are considered different, hence unique.
The next instance of A00 is succeeded and preceded by different lines, so that is also included in the output.
The same holds true for 03 - it is succeeded and preceded by different lines, so that is also included in the output.

SOLUTION :

```
uniq -u
```

Given a CSV file where each row contains the name of a city and its state separated by a comma, your task is to replace the newlines in the file with tabs as demonstrated in the sample.

Input Format

You are given a CSV file where each row contains the name of a city and its state separated by a comma.

Output Format

Replace the newlines in the input with tabs as demonstrated in the sample.

Sample Input

```
Albany, N.Y.
Albuquerque, N.M.
Anchorage, Alaska
Asheville, N.C.
Atlanta, Ga.
Atlantic City, N.J.
Austin, Texas
Baltimore, Md.
Baton Rouge, La.
Billings, Mont.
Birmingham, Ala.
Bismarck, N.D.
Boise, Idaho
Boston, Mass.
Bridgeport, Conn.
```

Sample Output

```
Albany, N.Y.    Albuquerque, N.M.   Anchorage, Alaska   Asheville,
N.C.Atlanta, Ga. Atlantic City, N.J. Austin, Texas   Baltimore, Md.  Baton
Rouge, La.    Billings, Mont. Birmingham, Ala.    Bismarck, N.D.  Boise,
Idaho    Boston, Mass.   Bridgeport, Conn.
```

Explanation

The delimiter between consecutive rows of data has been transformed from the newline to a tab.

first it reads data from file and merge them into single line with each line separated by tab.

SOLUTION :

```
paste -s
```

Given a CSV file where each row contains the name of a city and its state separated by a comma, your task is to restructure the file in such a way, that three consecutive rows are folded into one, and separated by tab.

Input Format

You are given a CSV file where each row contains the name of a city and its state separated by a comma.

Output Format

Restructure the file in such a way, that every group of three consecutive rows are folded into one, and separated by tab.

Sample Input

```
Albany, N.Y.
Albuquerque, N.M.
Anchorage, Alaska
Asheville, N.C.
Atlanta, Ga.
Atlantic City, N.J.
Austin, Texas
```

```
Baltimore, Md.
Baton Rouge, La.
Billings, Mont.
Birmingham, Ala.
Bismarck, N.D.
Boise, Idaho
Boston, Mass.
Bridgeport, Conn.
Sample Output

Albany, N.Y.    Albuquerque, N.M.   Anchorage, Alaska
Asheville, N.C. Atlanta, Ga.    Atlantic City, N.J.
Austin, Texas   Baltimore, Md.  Baton Rouge, La.
Billings, Mont. Birmingham, Ala.    Bismarck, N.D.
Boise, Idaho    Boston, Mass.   Bridgeport, Conn.
```

Normally, paste command prints the line of one file in one column. We can use hyphens(-) to increase the number of columns. For example, for three columns, you can use three hyphens like below.

SOLUTION :

```
paste - - -
```

```
In this challenge, we practice using the paste command to merge lines of a
given file.

You are given a CSV file where each row contains the name of a city and its
state separated by a comma. Your task is to replace the newlines in the file
with semicolons as demonstrated in the sample.

Input Format

You are given a CSV file where each row contains the name of a city and its
state separated by a comma.

Output Format

Replace the newlines in the input file with semicolons as demonstrated in the
sample.
```

Sample Input

```
Albany, N.Y.
Albuquerque, N.M.
Anchorage, Alaska
Asheville, N.C.
Atlanta, Ga.
Atlantic City, N.J.
Austin, Texas
Baltimore, Md.
Baton Rouge, La.
Billings, Mont.
Birmingham, Ala.
Bismarck, N.D.
Boise, Idaho
Boston, Mass.
Bridgeport, Conn.
```

Sample Output

```
Albany, N.Y.;Albuquerque, N.M.;Anchorage, Alaska;Asheville, N.C.;Atlanta,
Ga.;Atlantic City, N.J.;Austin, Texas;Baltimore, Md.;Baton Rouge,
La.;Billings, Mont.;Birmingham, Ala.;Bismarck, N.D.;Boise, Idaho;Boston,
Mass.;Bridgeport, Conn.
```

Explanation

The delimiter between consecutive rows of data has been transformed from the newline to a semicolon.

SOLUTION :

```
paste -d ';' -s
```

we use -d to specify custom delimiter instead of tab .

In this challenge, we practice using the paste command to merge lines of a given file.

You are given a CSV file where each row contains the name of a city and its state separated by a comma. Your task is to restructure the file so that three consecutive rows are folded into one line and are separated by semicolons.

Input Format

You are given a CSV file where each row contains the name of a city and its state separated by a comma.

Output Format

Restructure the file so that three consecutive rows are folded into one line and are separated by semicolons.

Sample Input

```
Albany, N.Y.
Albuquerque, N.M.
Anchorage, Alaska
Asheville, N.C.
Atlanta, Ga.
Atlantic City, N.J.
Austin, Texas
Baltimore, Md.
Baton Rouge, La.
Billings, Mont.
Birmingham, Ala.
Bismarck, N.D.
Boise, Idaho
Boston, Mass.
Bridgeport, Conn.
```

Sample Output

```
Albany, N.Y.;Albuquerque, N.M.;Anchorage, Alaska
Asheville, N.C.;Atlanta, Ga.;Atlantic City, N.J.
Austin, Texas;Baltimore, Md.;Baton Rouge, La.
Billings, Mont.;Birmingham, Ala.;Bismarck, N.D.
Boise, Idaho;Boston, Mass.;Bridgeport, Conn.
```

Explanation

The given input file has been reshaped as required.

SOLUTION :

```
paste -d ';' - - -
```