20-02-2025

GREP:

Tool for searching text using patterns or regular expressions in files or command output.

Flag	Purpose
-i	case senitive search
-v	invert match
-c	count matching lines
-n	show line numbers
-r	recursive search in dirs
-o	show only matched text
-w	match whole words
-A	show n lines before A match
-B	show n lines before A match
-C	Show N lines before and after the match
-E	Use extended regex
-P	Perl regex

.	match any single char
^	match the beginning of a line
&	match the end of a line

Extended Regex:	
cat | dog	OR
()	grouping
+	one or more occurences
*	0 or more occurences
?	0 or 1 occurences
{n,m}	matches n and m occurences of previous char
{n}	matches exactly n number of digits
[^abc]	matched any char except a,b,c
\b	word boundary

Only match the email id from the logfile:

```
root@Goosari:~# grep -Eo '[a-zA-Z0-9._%]+@[a-zAZ0-9.-]+\.[a-zA-Z]{2,}' logfile.md
admin@example.com
john.doe@company.org
sarah.jenkins@company.org
sarah.jenkins@company.org
michael.brown@example.net
lisa.wong@company.org
david.kim@example.com
emma.davis@company.org
carlos.rodriguez@example.org
admin@example.com
olivia.parker@company.org
james.wilson@example.net
sophia.nguyen@company.org
admin@example.com
ethan.miller@example.com
root@Goosari:~#
```

Only match the date from the log file:

```
root@Goosari:~# grep -Eo '[0-9]{4}-[0-9]{2}-[0-9]{2}' logfile.md
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
2024-02-01
```

Match the pattern using “()”:

```
root@Goosari:~# grep -E 'app-server-(1|2)' logfile.md
2024-02-01 07:23:45 INFO [app-server-1] User 'admin@example.com' logged in successfully
2024-02-01 07:24:12 DEBUG [app-server-1] Session a429f1db-ea3c-42f8-a03f-c10b6d8f9f1a created
2024-02-01 07:25:33 INFO [app-server-1] Database connection established - pool size: 25
2024-02-01 07:26:14 WARN [app-server-1] Database query took 1583ms to execute
2024-02-01 07:30:42 INFO [app-server-2] User 'john.doe@company.org' logged in successfully
2024-02-01 07:32:18 ERROR [app-server-1] Failed to connect to payment gateway: Connection timed out
2024-02-01 07:32:19 ERROR [app-server-1] Transaction 392841 failed: PAYMENT_GATEWAY_ERROR
2024-02-01 07:33:01 INFO [app-server-2] API request received: GET /api/v2/products?category=electronics
2024-02-01 07:33:02 DEBUG [app-server-2] Query params: {"category": "electronics", "limit": 50, "sort": "price_a
sc"}
2024-02-01 07:33:04 INFO [app-server-2] API request completed in 2781ms
2024-02-01 07:40:11 WARN [app-server-1] Memory usage at 82%, consider scaling up
```

Match the CPU usage and Memory for particular range using “[]”:

```
root@Goosari:~# grep -E 'CPU usage: [4-9].%, Memory: [8-9].%' logfile.md
2024-02-01 08:51:14 INFO [monitoring] CPU usage: 52%, Memory: 81%, Disk: 52%
2024-02-01 10:19:14 INFO [monitoring] CPU usage: 62%, Memory: 83%, Disk: 53%
2024-02-01 10:51:14 INFO [monitoring] CPU usage: 58%, Memory: 85%, Disk: 54%
root@Goosari:~#
```

Use “?” to match the 0 or 1 occurrences:

```
root@Goosari:~# grep -E " successf?u?l?l?y?" logfile.md
2024-02-01 07:23:45 INFO [app-server-1] User 'admin@example.com' logged in successfully
2024-02-01 07:30:42 INFO [app-server-2] User 'john.doe@company.org' logged in successfully
2024-02-01 07:42:56 INFO [app-server-2] User 'sarah.jenkins@company.org' logged in successfully
2024-02-01 07:45:33 INFO [app-server-1] Order 45928 created successfully
2024-02-01 07:50:22 INFO [app-server-2] User 'michael.brown@example.net' logged in successfully
2024-02-01 08:07:56 INFO [app-server-2] User 'lisa.wong@company.org' logged in successfully
2024-02-01 08:25:36 INFO [backup-service] Backup completed successfully - 1.2GB
2024-02-01 08:30:42 INFO [app-server-2] User 'david.kim@example.com' logged in successfully
2024-02-01 08:42:57 INFO [app-server-1] User settings updated successfully
2024-02-01 08:45:30 INFO [app-server-2] User 'emma.davis@company.org' logged in successfully
2024-02-01 08:56:12 INFO [app-server-2] Service restarted successfully
2024-02-01 09:00:45 INFO [app-server-1] User 'carlos.rodriguez@example.org' logged in successfully
2024-02-01 09:21:12 INFO [notification-service] Test email sent successfully
2024-02-01 09:25:33 INFO [app-server-2] User 'olivia.parker@company.org' logged in successfully
2024-02-01 09:42:12 INFO [app-server-2] User 'james.wilson@example.net' logged in successfully
2024-02-01 09:45:33 INFO [app-server-1] Retry successful after deadlock
2024-02-01 09:55:27 INFO [cron-service] Search index updated successfully - 15203 documents
2024-02-01 10:00:45 INFO [app-server-2] User 'sophia.nguyen@company.org' logged in successfully
2024-02-01 10:10:15 INFO [app-server-2] Feedback #1587 submitted successfully
2024-02-01 10:22:33 INFO [app-server-1] User 'admin@example.com' logged in successfully
2024-02-01 10:42:56 INFO [app-server-2] User 'ethan.miller@example.com' logged in successfully
root@Goosari:~#
```

Count number of lines with “User” keyword:

```
root@Goosari:~# grep -E "User" logfile.md | wc -l
15
```

Example:

```
root@Goosari:~# grep -Eo '[0-9]{4}-[0-9]{2}-[0-9]{2} [0-9]{2}:[0-9]{2}:[0-9]{2}' logfile.md
2024-02-01 07:23:45
2024-02-01 07:24:12
2024-02-01 07:25:33
2024-02-01 07:26:14
2024-02-01 07:30:42
2024-02-01 07:32:18
2024-02-01 07:32:19
2024-02-01 07:33:01
2024-02-01 07:33:02
2024-02-01 07:33:04
2024-02-01 07:35:27
2024-02-01 07:40:11
```

Match lines with ERROR and Failed in the same line:

```
root@Goosari:~# grep -E "(ERROR).*Failed" logfile.md
2024-02-01 07:32:18 ERROR [app-server-1] Failed to connect to payment gateway: Connection timed out
2024-02-01 08:35:27 ERROR [app-server-1] Failed to process payment: INVALID_CARD_NUMBER
2024-02-01 09:20:11 ERROR [notification-service] Failed to send email notification: Authentication failed
2024-02-01 10:15:22 ERROR [app-server-1] Failed to connect to external API: https://partner-api.example.com
root@Goosari:~#
```

Match the IP address:

```
root@Goosari:~# grep -Eo "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" logfile.md
203.0.113.42
```

AWK:

Text-processing tool that works well in Bash scripts for filtering, transforming, and analyzing data.

Syntax:

```
root@Goosari:~# awk [options] 'pattern {action}' filename
```

Filter to a specific time range:

```
root@Goosari:~# awk '/2025-02-19 10:[0-5][0-9]:[0-5][0-9]/' generatedlog.md
2025-02-19 10:04:33 WARNING [database] CPU usage: 44%, Memory: 70%, Disk: 45%
2025-02-19 10:11:33 DEBUG [monitoring] Successfully processed payment
2025-02-19 10:19:33 ERROR [app-server-1] Disk space low on /dev/sda1
2025-02-19 10:21:33 ERROR [app-server-1] Database connection restored
2025-02-19 10:27:33 DEBUG [app-server-1] Disk space low on /dev/sda1
2025-02-19 10:33:33 ERROR [monitoring] Successfully processed payment
2025-02-19 10:40:33 ERROR [auth-service] Database connection restored
2025-02-19 10:50:33 ERROR [auth-service] Successfully processed payment
2025-02-19 10:59:33 DEBUG [app-server-1] Successfully processed payment
root@Goosari:~#
```

Example:

```
root@Goosari:~# awk '{print "User:", $1, "Status:", $3}' generatedlog.md
User: 2025-02-19 Status: ERROR
User: 2025-02-19 Status: INFO
User: 2025-02-19 Status: INFO
User: 2025-02-19 Status: WARNING
User: 2025-02-19 Status: WARNING
User: 2025-02-19 Status: DEBUG
User: 2025-02-19 Status: ERROR
User: 2025-02-19 Status: DEBUG
```

Print the lines with the word ERROR:

```
root@Goosari:~# awk '/ERROR/ {print $0}' generatedlog.md
2025-02-19 09:05:33 ERROR [app-server-1] Failed to connect to database
2025-02-19 09:37:33 ERROR [database] API request completed in 6317ms
2025-02-19 09:46:33 ERROR [app-server-1] API request completed in 4389ms
2025-02-19 10:19:33 ERROR [app-server-1] Disk space low on /dev/sda1
2025-02-19 10:21:33 ERROR [app-server-1] Database connection restored
2025-02-19 10:33:33 ERROR [monitoring] Successfully processed payment
2025-02-19 10:40:33 ERROR [auth-service] Database connection restored
2025-02-19 10:50:33 ERROR [auth-service] Successfully processed payment
2025-02-19 11:47:33 ERROR [app-server-1] CPU usage: 43%, Memory: 89%, Disk: 85%
2025-02-19 13:05:33 ERROR [database] CPU usage: 86%, Memory: 64%, Disk: 56%
2025-02-19 13:16:33 ERROR [auth-service] Successfully processed payment
2025-02-19 13:20:33 ERROR [auth-service] Disk space low on /dev/sda1
2025-02-19 13:51:33 ERROR [app-server-1] Database connection restored
2025-02-19 14:07:33 ERROR [auth-service] Disk space low on /dev/sda1
2025-02-19 14:32:33 ERROR [database] User 'john.doe@company.org' failed login attempt
2025-02-19 14:55:33 ERROR [auth-service] Failed to connect to database
```

Calculate the number of lines:

```
root@Goosari:~# awk 'END {print "Total lines: ",NR}' generatedlog.md
Total lines:  500
```

Calculate the number of lines using grep:

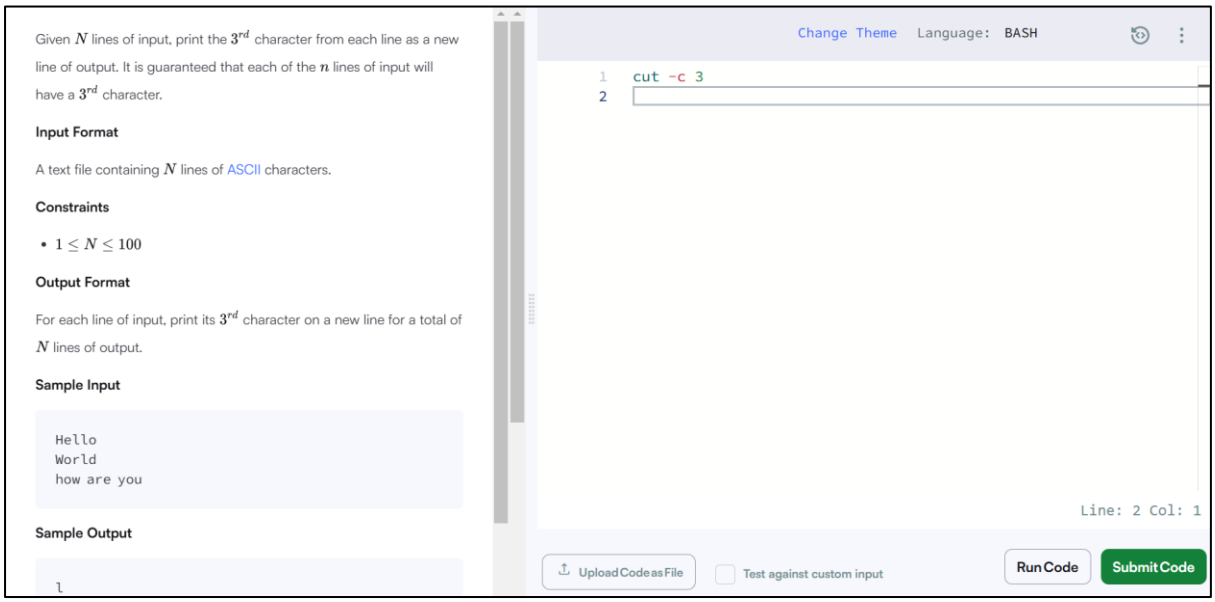
```
root@Goosari:~# grep '[0-9]' generatedlog.md | wc -l
500
```

Convert the logfile into a tsv file using awk:

```
root@Goosari:~# echo '' > output.tsv
root@Goosari:~# awk 'BEGIN {OFS="\t"} {print $1, $2, $3}' generatedlog.md >> output.tsv
root@Goosari:~# cat output.tsv

2025-02-19      09:05:33        ERROR
2025-02-19      09:08:33        INFO
2025-02-19      09:18:33        INFO
2025-02-19      09:23:33        WARNING
2025-02-19      09:33:33        WARNING
2025-02-19      09:36:33        DEBUG
2025-02-19      09:37:33        ERROR
2025-02-19      09:41:33        DEBUG
2025-02-19      09:46:33        ERROR
2025-02-19      09:54:33        DEBUG
2025-02-19      09:56:33        INFO
2025-02-19      09:59:33        DEBUG
2025-02-19      10:04:33        WARNING
```

Practice Problems (Hackerrank):



Display the 2^{nd} and 7^{th} character from each line of text.

Input Format

A text file with N lines of ASCII text only.

Constraints

- $1 \leq N \leq 100$

Output Format

The output should contain N lines. Each line should contain just two characters at the 2^{nd} and the 7^{th} position of the corresponding input line.

Sample Input

Hello

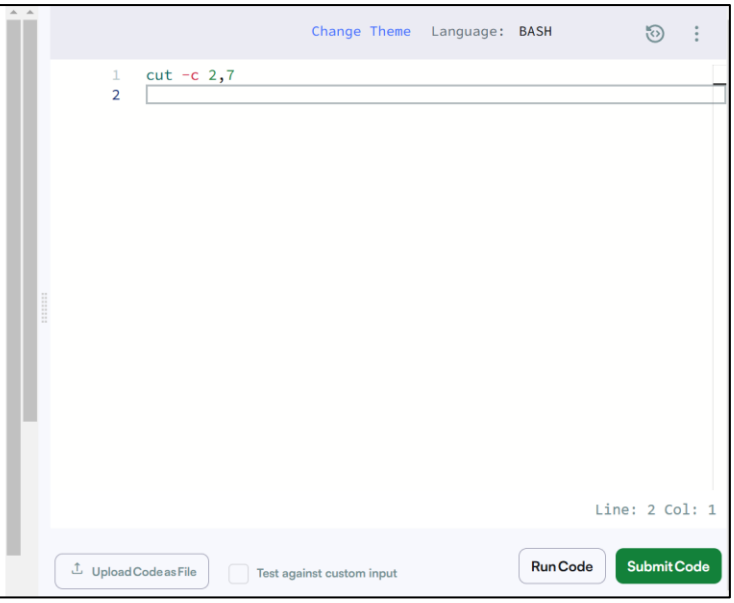
World

how are you

Sample Output

e

o



Display a range of characters starting at the 2^{nd} position of a string and ending at the 7^{th} position (both positions included).

Input Format

A text file containing N lines of ASCII text only.

Constraints

- $1 \leq N \leq 100$

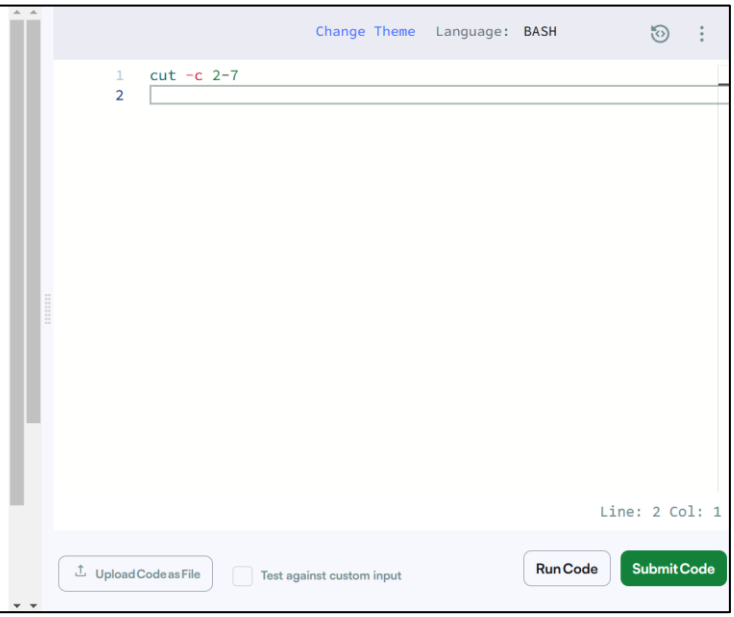
Output Format

The output should contain N lines.
Each line should contain the range of characters starting at the 2^{nd} position of a string and ending at the 7^{th} position (both positions included).

Sample Input

```
Hello
World
how are you
```

Sample Output



Display the first four characters from each line of text.

Input Format

A text file with lines of ASCII text only.

Constraints

$1 \leq N \leq 100$

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

Output Format

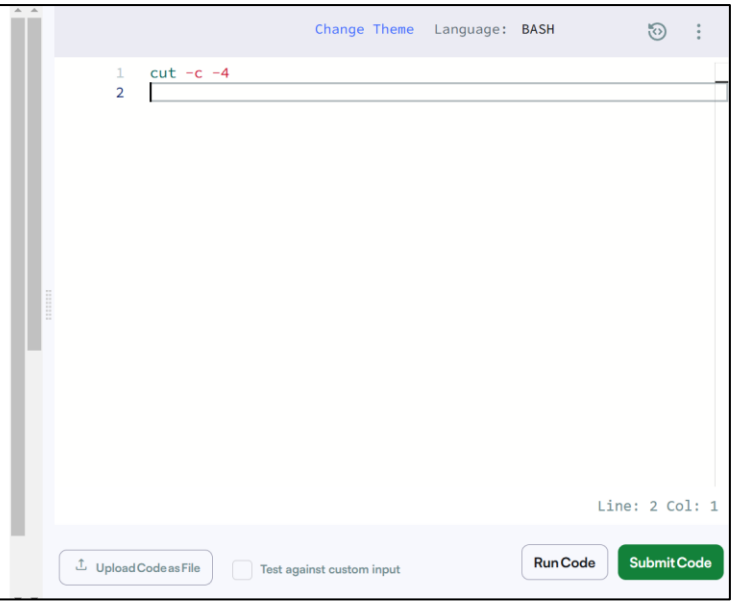
The output should contain **N** lines. Each line should contain just the first four characters of the corresponding input line.

Sample Input

```
Hello
World
how are you
```

Sample Output

```
Hell
Worl
```



Given a tab delimited file with several columns (tsv format) print the first three fields.

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 first three fields.

Sample Input

```
1  New York, New York[10]  8,244,910  1  New York-
2  Los Angeles, California 3,819,702  2  Los Ange
3  Chicago, Illinois  2,707,120  3  Chicago-Joli
4  Houston, Texas  2,145,146  4  Dallas-Fort Worth
5  Philadelphia, Pennsylvania[11]  1,536,471  5
```

Change Theme Language: BASH

```
1 cut -f-3
2
```

Line: 2 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

Print the characters from thirteenth position to the end.

Input Format

A text file with lines of ASCII text only.

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 line, print the characters from thirteenth position to the end.

Sample Input

```
New York is a state in the Northeastern and Mid-Atla
New York is the 27th-most extensive, the third-most
New York is bordered by New Jersey and Pennsylvania
About one third of all the battles of the Revolutior
Henry Hudson's 1609 voyage marked the beginning of
```

Sample Output

Change Theme Language BASH

```
1 cut -c 13-
2
```

Line: 2 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

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

Sample Input

```
Hello
World
how are you
```

Sample Output

Change Theme Language: BASH

```
1 cut -d " " -f 4
2
```

Line: 2 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

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-Atl
New York is the 27th-most extensive, the third-most
New York is bordered by New Jersey and Pennsylvania
About one third of all the battles of the Revolutio
Henry Hudson's 1609 voyage marked the beginning of
```

Change Theme Language: BASH

```
1 cut -d " " -f-3
2
```

Line: 2 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code

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-
2  Los Angeles, California 3,819,702  2  Los Ange
3  Chicago, Illinois  2,707,120  3  Chicago-Joli
4  Houston, Texas  2,145,146  4  Dallas-Fort Worth
```

Change Theme Language: BASH

```
1 cut -f 2-
2
```

Line: 2 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code