

## CS256 ASSIGNMENT-1

### UNIX AND SHELL SCRIPTING LAB

ROLL NO -422151

SEC-A

Write shell script for searching for various patterns using grep, pr, head, tail, cut, paste, sort, uniq, and tr.

#### grep Command

```
(base) student@24:~/Desktop/422151/scriptlab$ grep -i "hyder" City.txt
Hyderabad
```

```
(base) student@24:~/Desktop/422151/scriptlab$ grep -c "Mumbai" City.txt
2
```

```
(base) student@24:~/Desktop/422151/scriptlab$ grep -w "Mumbai" City.txt
Mumbai
Mumbai
(base) student@24:~/Desktop/422151/scriptlab$ grep -w "Mumb" City.txt
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ grep -0 "Mumbai" City.txt
Mumbai
--
Mumbai
(base) student@24:~/Desktop/422151/scriptlab$ grep -0 "Mumb" City.txt
Mumbai
--
Mumbai
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ grep -n "Mumbai" City.txt
9:Mumbai
14:Mumbai
(base) student@24:~/Desktop/422151/scriptlab$ grep -n "Mumb" City.txt
9:Mumbai
14:Mumbai
(base) student@24:~/Desktop/422151/scriptlab$
```

```
student@24: ~/Desktop/422151/scriptlab
14:Mumbai
(base) student@24:~/Desktop/422151/scriptlab$ grep --help
Usage: grep [OPTION]... PATTERNS [FILE]...
Search for PATTERNS in each FILE.
Example: grep -i 'hello world' menu.h main.c
PATTERNS can contain multiple patterns separated by newlines.

Pattern selection and interpretation:
-E, --extended-regexp    PATTERNS are extended regular expressions
-F, --fixed-strings      PATTERNS are strings
-G, --basic-regexp       PATTERNS are basic regular expressions
-P, --perl-regexp        PATTERNS are Perl regular expressions
-e, --regexp=PATTERNS    use PATTERNS for matching
-f, --file=FILE          take PATTERNS from FILE
-i, --ignore-case        ignore case distinctions in patterns and data
                        --no-ignore-case    do not ignore case distinctions (default)
-w, --word-regexp        match only whole words
-x, --line-regexp        match only whole lines
-z, --null-data          a data line ends in 0 byte, not newline

Miscellaneous:
-s, --no-messages        suppress error messages
-v, --invert-match        select non-matching lines
-V, --version            display version information and exit
--help                  display this help text and exit

Output control:
-m, --max-count=NUM      stop after NUM selected lines
-b, --byte-offset        print the byte offset with output lines
-n, --line-number        print line number with output lines
                        --line-buffered    flush output on every line
-H, --with-filename      print file name with output lines
-h, --no-filename        suppress the file name prefix on output
                        --label=LABEL     use LABEL as the standard input file name prefix

ix
-o, --only-matching      show only nonempty parts of lines that match
-q, --quiet, --silent    suppress all normal output
                        --binary-files=TYPE
                        TYPE is 'binary', 'text', or 'without-match'
                        equivalent to --binary-files=text
-a, --text               equivalent to --binary-files=without-match
-I                       equivalent to --binary-files=without-match
-d, --directories=ACTION  how to handle directories;
                        ACTION is 'read', 'recurse', or 'skip'
-D, --devices=ACTION     how to handle devices, FIFOs and sockets;
                        ACTION is 'read' or 'skip'
-r, --recursive          like --directories=recurse
```

### pr command

```
(base) student@24:~/Desktop/422151/scriptlab$ pr City.txt
```

```
2024-02-14 14:20
```

```
City.txt
```

```
Page 1
```

```
New York  
Las vegas  
Tokyo  
New York  
Las vegas  
Fuji  
Atlanta  
Delhi  
Mumbai  
Hyderabad  
Kolkata  
Fuji  
Pune  
Mumbai  
Adelaide  
Oval  
Lords  
Headingley  
Colosseum
```

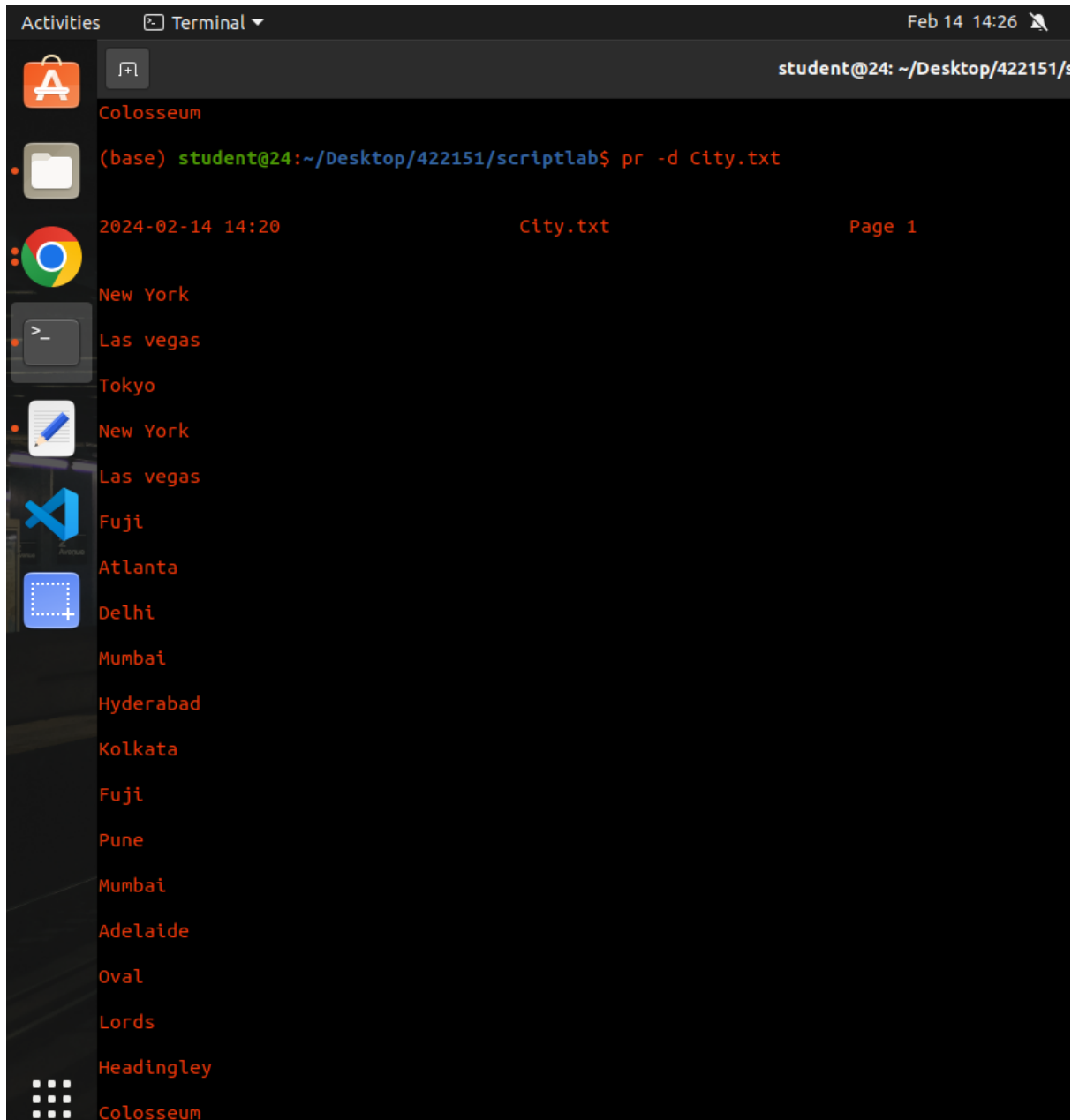
```
(base) student@24:~/Desktop/422151/scriptlab$ pr -t City.txt
```

```
New York  
Las vegas  
Tokyo  
New York  
Las vegas  
Fuji  
Atlanta  
Delhi  
Mumbai  
Hyderabad  
Kolkata  
Fuji  
Pune  
Mumbai  
Adelaide  
Oval  
Lords  
Headingley  
Colosseum
```

```
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ pr --version
pr (GNU coreutils) 8.30
Copyright (C) 2018 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Written by Pete TerMaat and Roland Huebner.
(base) student@24:~/Desktop/422151/scriptlab$
```



The screenshot shows a Linux desktop environment. At the top, there is a taskbar with the 'Activities' button on the left and the system clock 'Feb 14 14:26' on the right. Below the taskbar, there are two windows. The first window is a file manager showing a directory listing of 'Colosseum'. The second window is a terminal window titled 'Terminal' with the prompt 'student@24: ~/Desktop/422151/'. The terminal shows the command 'pr -d City.txt' being executed, which outputs a list of cities: New York, Las Vegas, Tokyo, New York, Las Vegas, Fuji, Atlanta, Delhi, Mumbai, Hyderabad, Kolkata, Fuji, Pune, Mumbai, Adelaide, Oval, Lords, Headingley, and Colosseum. The desktop background is dark, and there are several application icons on the left side of the screen.

```
Activities Terminal Feb 14 14:26 student@24: ~/Desktop/422151/Colosseum
(base) student@24:~/Desktop/422151/scriptlab$ pr -d City.txt
2024-02-14 14:20 City.txt Page 1
New York
Las Vegas
Tokyo
New York
Las Vegas
Fuji
Atlanta
Delhi
Mumbai
Hyderabad
Kolkata
Fuji
Pune
Mumbai
Adelaide
Oval
Lords
Headingley
Colosseum
```

```
(base) student@24:~/Desktop/422151/scriptlab$ pr -4 City.txt
```

```
2024-02-14 14:20
```

```
City.txt
```

```
Page 1
```

New York	Fuji	Kolkata	Oval
Las vegas	Atlanta	Fuji	Lords
Tokyo	Delhi	Pune	Headingley
New York	Mumbai	Mumbai	Colosseum
Las vegas	Hyderabad	Adelaide	

```
student@24: ~/Desktop/422151/scriptlab
^C
(base) student@24:~/Desktop/422151/scriptlab$ pr --help
Usage: pr [OPTION]... [FILE]...
Paginate or columnate FILE(s) for printing.

With no FILE, or when FILE is -, read standard input.

Mandatory arguments to long options are mandatory for short options too.
  +FIRST_PAGE[:LAST_PAGE], --pages=FIRST_PAGE[:LAST_PAGE]
                                begin [stop] printing with page FIRST_[LAST_]PAGE
  -COLUMN, --columns=COLUMN
                                output COLUMN columns and print columns down,
                                unless -a is used. Balance number of lines in the
                                columns on each page
  -a, --across
                                print columns across rather than down, used together
                                with -COLUMN
  -c, --show-control-chars
                                use hat notation (^G) and octal backslash notation
  -d, --double-space
                                double space the output
  -D, --date-format=FORMAT
                                use FORMAT for the header date
  -e[CHAR[WIDTH]], --expand-tabs[=CHAR[WIDTH]]
                                expand input CHARs (TABS) to tab WIDTH (8)
  -F, -f, --form-feed
                                use form feeds instead of newlines to separate pages
                                (by a 3-line page header with -F or a 5-line header
                                and trailer without -F)
  -h, --header=HEADER
                                use a centered HEADER instead of filename in page head
er,
                                -h "" prints a blank line, don't use -h""
  -i[CHAR[WIDTH]], --output-tabs[=CHAR[WIDTH]]
                                replace spaces with CHARs (TABS) to tab WIDTH (8)
  -J, --join-lines
                                merge full lines, turns off -W line truncation, no col
umn
                                alignment, --sep-string[=STRING] sets separators
  -l, --length=PAGE_LENGTH
                                set the page length to PAGE_LENGTH (66) lines
                                (default number of lines of text 56, and with -F 63).
                                implies -t if PAGE_LENGTH <= 10
  -m, --merge
                                print all files in parallel, one in each column,
                                truncate lines, but join lines of full length with -J
  -n[SEP[DIGITS]], --number-lines[=SEP[DIGITS]]
                                number lines, use DIGITS (5) digits, then SEP (TAB),
                                default counting starts with 1st line of input file
```

### head command

```
(base) student@24:~/Desktop/422151/scriptlab$ head City.txt
New York
Las vegas
Tokyo
New York
Las vegas
Fuji
Atlanta
Delhi
Mumbai
Hyderabad
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ head -n 2 City.txt
New York
Las vegas
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ head -c 20 City.txt
New York
Las vegas
T(base) student@24:~/Desktop/422151/scriptlab$
```

```
T(base) student@24:~/Desktop/422151/scriptlab$ head -v City.txt
==> City.txt <==
New York
Las vegas
Tokyo
New York
Las vegas
Fuji
Atlanta
Delhi
Mumbai
Hyderabad
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ head numbers.txt City.txt
==> numbers.txt <==
1
2
3
4
5
6
7
8
9

==> City.txt <==
New York
Las vegas
Tokyo
New York
Las vegas
Fuji
Atlanta
Delhi
Mumbai
Hyderabad
(base) student@24:~/Desktop/422151/scriptlab$
```

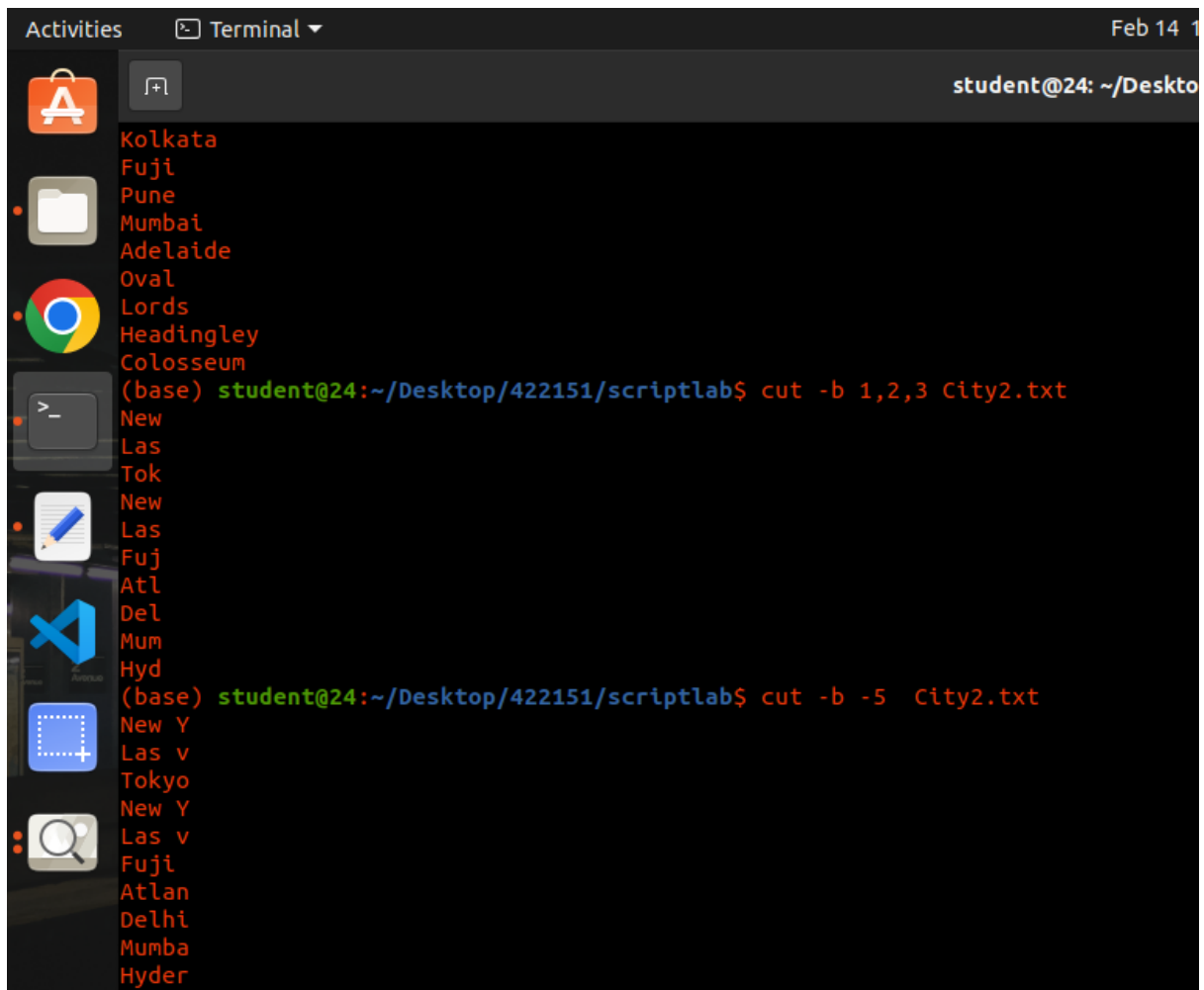
```
(base) student@24:~/Desktop/422151/scriptlab$ head numbers.txt > out_numbers.txt
(base) student@24:~/Desktop/422151/scriptlab$ cat out_numbers.txt
1
2
3
4
5
6
7
8
9
(base) student@24:~/Desktop/422151/scriptlab$
```

#### tail command

```
(base) student@24:~/Desktop/422151/scriptlab$ tail +6 out_numbers.txt
6
7
8
9
(base) student@24:~/Desktop/422151/scriptlab$
```



## cut command



A terminal window titled "Terminal" with a date of "Feb 14 1" in the top right corner. The window shows a list of city names: Kolkata, Fuji, Pune, Mumbai, Adelaide, Oval, Lords, Headingley, Colosseum, New, Las, Tok, New, Las, Fuj, Atl, Del, Mum, Hyd. The prompt is "(base) student@24:~/Desktop/422151/scriptlab\$". The command "cut -b 1,2,3 City2.txt" is entered, and the output shows the first three characters of each line: New, Las, Tok, New, Las, Fuj, Atl, Del, Mum, Hyd.

```
(base) student@24:~/Desktop/422151/scriptlab$ cut -b 1,2,3 City2.txt
New
Las
Tok
New
Las
Fuj
Atl
Del
Mum
Hyd
```



A terminal window showing the command "cut -c 3-7 City2.txt" being executed. The output shows the characters from the 3rd to the 7th position of each line: w Yor, s veg, kyo, w Yor, s veg, ji, lanta, lhi, mbai, derab.

```
(base) student@24:~/Desktop/422151/scriptlab$ cut -c 3-7 City2.txt
w Yor
s veg
kyo
w Yor
s veg
ji
lanta
lhi
mbai
derab
(base) student@24:~/Desktop/422151/scriptlab$
```

# paste command

```
Activities Terminal Feb 14 15:04
student@24: ~/Desktop/422151/scriptlab

Atlanta
Delhi
Mumbai
Hyderabad
Kolkata
Fuji
Pune
Mumbai
Adelaide
Oval
Lords
Headingley
Colosseum
(base) student@24:~/Desktop/422151/scriptlab$ paste numbers.txt City2.txt
1 New York
2 Las Vegas
3 Tokyo
4 New York
5 Las Vegas
6 Fuji
7 Atlanta
8 Delhi
9 Mumbai
Hyderabad
(base) student@24:~/Desktop/422151/scriptlab$ paste -d "|" City.txt numbers.txt out_numbers.txt
New York|1,1
Las Vegas|2,2
Tokyo|3,3
New York|4,4
Las Vegas|5,5
Fuji|6,6
Atlanta|7,7
Delhi|8,8
Mumbai|9,9
Hyderabad|,
Kolkata|,
Fuji|,
Pune|,
Mumbai|,
Adelaide|,
Oval|,
Lords|,
Headingley|,
Colosseum|,
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ paste -s -d ":" City.txt numbers.txt out_numbers.txt
New York:Las Vegas:Tokyo:New York:Las Vegas:Fuji:Atlanta:Delhi:Mumbai:Hyderabad:Kolkata:Fuji:Pune:Mumbai:Adelaide:Oval:Lords:Headingley:Colosseum
1:2:3:4:5:6:7:8:9
1:2:3:4:5:6:7:8:9
(base) student@24:~/Desktop/422151/scriptlab$
```

## sort command

```
(base) student@24:~/Desktop/422151/scriptlab$ sort City2.txt
Atlanta
Delhi
Fuji
Hyderabad
Las vegas
Las vegas
Mumbai
New York
New York
Tokyo
(base) student@24:~/Desktop/422151/scriptlab$ cat City2.txt
New York
Las vegas
Tokyo
New York
Las vegas
Fuji
Atlanta
Delhi
Mumbai
Hyderabad
(base) student@24:~/Desktop/422151/scriptlab$ sort -r City2.txt
Tokyo
New York
New York
Mumbai
Las vegas
Las vegas
Hyderabad
Fuji
Delhi
Atlanta
```

## uniq command

```
(base) student@24:~/Desktop/422151/scriptlab$ cat City3.txt
Adelaide
Atlanta
Colosseum
Delhi
Fuji
Fuji
Headingley
Hyderabad
Kolkata
Las vegas
Las vegas
Lords
Mumbai
Mumbai
New York
New York
Oval
Pune
Tokyo
```

```
(base) student@24:~/Desktop/422151/scriptlab$ uniq City3.txt
Adelaide
Atlanta
Colosseum
Delhi
Fuji
Headingley
Hyderabad
Kolkata
Las vegas
Lords
Mumbai
New York
Oval
Pune
Tokyo
```

```
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ uniq -c City3.txt
1 Adelaide
1 Atlanta
1 Colosseum
1 Delhi
2 Fuji
1 Headingley
1 Hyderabad
1 Kolkata
2 Las vegas
1 Lords
2 Mumbai
2 New York
1 Oval
1 Pune
1 Tokyo
1
```

```
(base) student@24:~/Desktop/422151/scriptlab$ uniq -d City3.txt
Fuji
Las vegas
Mumbai
New York
```

```
(base) student@24:~/Desktop/422151/scriptlab$ uniq -u City3.txt
Adelaide
Atlanta
Colosseum
Delhi
Headingley
Hyderabad
Kolkata
Lords
Oval
Pune
Tokyo
```

## tr command

```
(base) student@24:~/Desktop/422151/scriptlab$ cat City2.txt | tr [a-z] [A-Z]
NEW YORK
LAS VEGAS
TOKYO
NEW YORK
LAS VEGAS
FUJI
ATLANTA
DELHI
MUMBAI
HYDERABAD
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ echo "Welcome everyone"|tr -d W  
elcome everyone  
(base) student@24:~/Desktop/422151/scriptlab$
```

```
(base) student@24:~/Desktop/422151/scriptlab$ echo "today is 13 and id is 422151" | tr -d [:digit:]  
today is and id is  
(base) student@24:~/Desktop/422151/scriptlab$
```