Week-4:

a) Use the sort command to sort the file mytable according to the first field. Call the sorted file mytable (same name)
b) Print the file mytable
c) Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it my table (same name)
d) Print the new file, mytable
e) Logout of the system.

a. Use the sort command to sort the file mytable according to the first field. Call the sorted file mytable

$ cat mytable

```
1425 Ravi 15.65
4320 Ramu 26.27
6830 Sita 36.15
1450 Raju 21.86
```

```
$ sort -k1 -u mytable > mytable        (-u is used to delete duplicate records)
```

```
1425 Ravi 15.65
1450 Raju 21.86
4320 Ramu 26.27
6830 Sita 36.15
```

Description:-

sort command

Sort command is used to rearrange the lines in a text file in sorted order numerically or alphabetically.

Syntax: sort [OPTION]... [FILE]...

Options

- -k, --field wise sort.
- -b, --ignore-leading-blanks.
- -u is used to delete duplicate entries
- -d, --dictionary-order Consider only blanks and alphanumeric characters.
- -f, --ignore-case, lower case to upper case characters.
- -g, --general-numeric-sort Compare according to general numerical value.
- -i, --ignore-nonprinting. Consider only printable characters.
- -M, --month-sort Compare (unknown) < `JAN' < ... < `DEC'.
- -h, --human-numeric-sort Compare human readable numbers (e.g., "2K", "1G").
- -n, --numeric-sort. Compare according to string numerical value.
- -r, --reverse Reverse the result of comparisons.
- --sort=WORD Sort according to WORD: general-numeric -g, human-numeric -h, month -M, numeric -n, random -R, version -V.
- -V, --version-sort Natural sort of (version) numbers within text.

Note that this command does not actually change the input file, data.txt. If you want to write the output to a new file, output.txt, redirect the output like this:

b. Print the file mytable

```
$ cat mytable

1425 Ravi 15.65
1450 Raju 21.86
4320 Ramu 26.27
6830 Sita 36.15
```

c. Use the cut and paste commands to swap fields 2 and 3 of mytable. Call it my table (same name)

```
$ cut -d ' ' -f1 mytable > mytable1
$ cat mytable1
1425
1450
4320
6830


$ cut -d ' ' -f2 mytable > mytable2
$ cat mytable2

Ravi
Raju
Ramu
Sita

$ cut -d ' ' -f3 mytable > mytable3
$ cat mytable3
15.65
21.86
26.27
36.15

$ paste mytable3 mytable2
15.65   Ravi
21.86   Raju
26.27   Ramu
36.15   Sita

$ paste mytable1 mytable3 mytable2 > mytable

$ cat mytable

1425    15.65   Ravi
1450    21.86   Raju
4320    26.27   Ramu
6830    36.15   Sita
```

Description:

cut command

Remove or "cut out" sections of each line of a file or files.

Syntax

```
$ cut OPTION... [FILE]...
```

Options

- -b, --bytes=LIST Select only the bytes from each line as specified in LIST. LIST specifies a byte, a set of bytes, or a range of bytes; see Specifying LIST below.
- -c, --characters=LIST Select only the characters from each line as specified in LIST. LIST specifies a character, a set of characters, or a range of characters; see Specifying LISTbelow.
- -d, --delimiter=DELIM use character DELIM instead of a tab for the field delimiter.
- -f, --fields=LIST select only these fields on each line; also print any line that contains no delimiter character, unless the -s option is specified. LIST specifies a field, a set of fields, or a range of fields; see Specifying LIST below.
- -n This option is ignored, but is included for compatibility reasons.
- --complement complement the set of selected bytes, characters or fields.
- -s, --only-delimited do not print lines not containing delimiters.
- --output-delimiter=STRING use STRING as the output delimiter string. The default is to use the input delimiter.

To "cut" only the third field of each line, use the command:

```
$ cut -f 3 data.txt
```

If instead you want to "cut" only the second-through-fourth field of each line, use the command:

```
$ cut -f 2-4 data.txt
```

If you want to "cut" only the first-through-second and fourth-through-fifth field of each line (omitting the third field), use the command:

```
$ cut -f 1-2, 4-5 data.txt
```

If you want the third field and every field after it, omitting the first two fields. In this case, you could use the command:

```
$ cut -f 3- data.txt
```

Specifying a range with LIST also applies to cutting characters (-c) or bytes (-b) from a line. For example, to output only the third-through-twelfth character of every line ofdata.txt, use the command:

```
$ cut -c 3-12 data.txt
```

paste command

The paste command displays the corresponding lines of multiple files side-by-side.
Syntax: paste [OPTION]... [FILE]...
Examples:

```
$ paste file1.txt file2.txt
```

This command would display the contents of file1.txt and file2.txt, side-by-side, with the corresponding lines of each file separated by a tab.

```
$ cat file1
Linux
Unix
Solaris
HPUX
AIX
```

paste command with a single file:

1. paste command without any options is as good as the cat command when operated on a single file.

```
$ paste file1
Linux
Unix
Solaris
HPUX
AIX
```

2. Join all lines in a file:

```
$ paste -s file1
Linux Unix Solaris HPUX AIX
```

-s option of paste joins all the lines in a file. Since no delimiter is specified, default delimiter tab is used to separate the columns.

3. Join all lines using the comma delimiter:

```
$ paste -d, -s file1
Linux,Unix,Solaris,HPUX,AIX
```

-d option is used to specify the delimiter. Using this -d and -s combination, all the lines in the file get merged into a single line.

4. Merge a file by pasting the data into 2 columns:

```
$ paste - - < file1
Linux Unix
Solaris HPUX
AIX
```

The '-' reads a line from the standard input. Two '-' reads 2 lines and pastes them side by side.

5. Merge a file by pasting the data into 2 columns using a colon separator:

```
$ paste -d':' - - < file1
Linux:Unix
Solaris:HPUX
AIX:
```

This is same as joining every 2 lines in a file.

6. Merge a file by pasting the file contents into 3 columns:

```
$ paste - - - < file1
Linux Unix Solaris
HPUX AIX
```

7. Merge a file into 3 columns using 2 different delimiters:

```
$ paste -d ':,' - - - < file1
Linux:Unix,Solaris
HPUX:AIX,
```

The -d option can take multiple de-limiters. The 1st and 2nd columns is separated by ':', whereas the 2nd and 3rd are separated by a ','.

paste command examples for multiple files handling

Let us consider a file, file2, with the following contents:

```
$ cat file2
Suse
Fedora
CentOS
OEL
Ubuntu
```

8. paste contents of 2 files side by side.

```
$ paste file1 file2
Linux Suse
Unix Fedora
Solaris CentOS
HPUX OEL
AIX Ubuntu
```

paste command is used in scenarios to merge multiple files side by side. As shown above, the file contents are pasted side by side.

9. paste contents of 2 files side by side with a comma separator:

```
$ paste -d, file1 file2
Linux,Suse
Unix,Fedora
Solaris,CentOS
HPUX,OEL
AIX,Ubuntu
```

10. Read lines in both the files alternatively:

```
$ paste -d'\n' file1 file2
Linux
Suse
Unix
Fedora
Solaris
CentOS
HPUX
OEL
```

```
AIX
Ubuntu
```

d. Print the new file, mytable

```
$ cat mytable
1425    15.65   Ravi
1450    21.86   Raju
4320    26.27   Ramu
6830    36.15   Sita
```

e. Logout the system.

```
$ exit
```

Exit to log out from the operating system.