Birla Institute of Technology & Science, Pilani , K. K. BIRLA Goa Campus
Computer Programming (CS F111)
Second Semester 2013-2014
Lab – 01 (VI Editor)

Objectives:

1. Introduction
2. Creating your first file using VI Editor
3. Basic Operating Modes
4. Commands for deleting file contents
5. Miscellaneous VI commands
6. Copying and Pasting file contents
7. Pattern Searching
8. Substitution of Text
9. Exercises
10. Additional Reading Material

----

1. Introduction

Lab session-1 introduced with the directory structure used in UNIX and ways to navigate across directories. Also we encountered various types of files present in the directories. This lab session will guide you systematically to create such files in UNIX and some essential features like copy/paste.

A file is created / modified and saved by a program called text editor. There are several editor programs available in UNIX. VI is one such editor and most popular among people who use UNIX.

2. Creating your first file using VI:

Let us try to create a simple text file using VI. A file name is needed to uniquely identify a file on the file system. Let say file name is ***first***

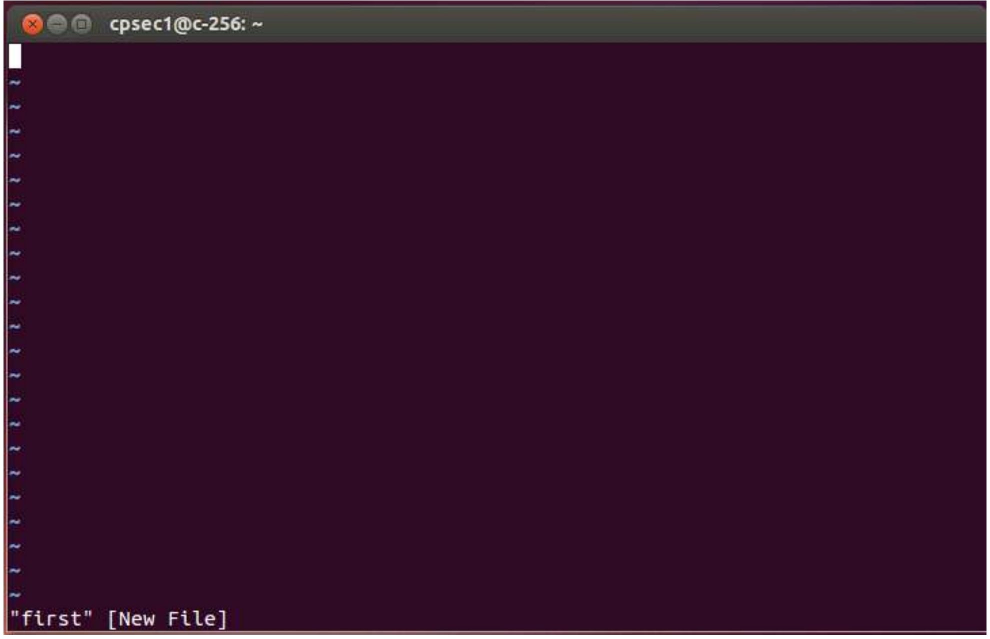
Let the current working directory be

```
[CPSEC1@localhost ~]$
```

The current working directory is initially empty. To start the vi program to create/edit a file named '***first***', type

```
[CPSEC1@localhost ~]$ vi first
```

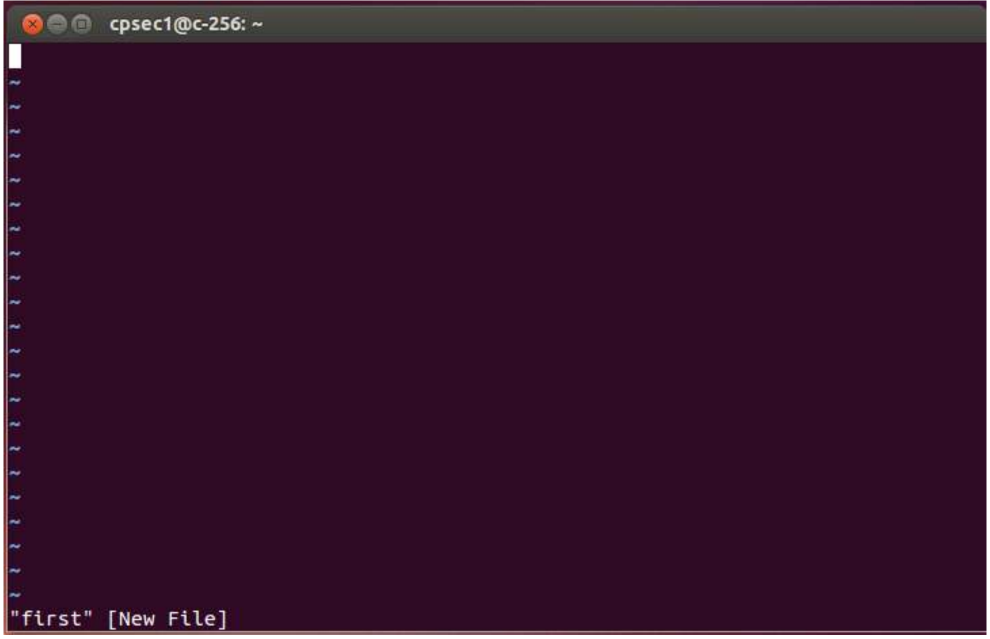
You will see the VI program opens (Fig-1) which is ready to create/edit the file named ***first***.



```
cpsec1@c-256: ~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"first" [New File]
```

Fig-1: VI editor, ready to edit the new file named '*first*'

Observe the cursor in the top of the screen. The editor is in ***command mode*** now. It is looking for the command from the user which directs it to do one of the many jobs that it knows. We want to add some contents to the file '***first***'. This can be done when the editor is in the ***insert mode***. To bring the editor to the insert mode from the current command mode, just press the key 'i'. You will be able to enter text into the editor only after pressing the key 'i' once.



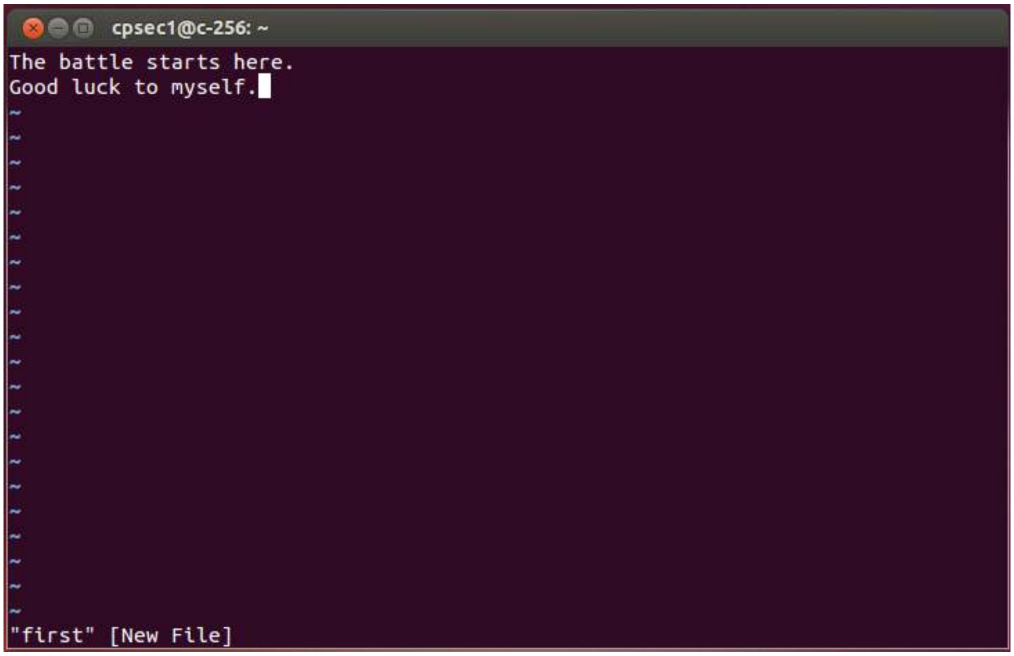
```

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"first" [New File]
```

Fig-2: Ready to input text

Type the following in the editor.

```
The battle starts here.
Good luck to myself.
```



```
The battle starts here.
Good luck to myself.
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"first" [New File]
```

Fig-3: Saving the typed contents

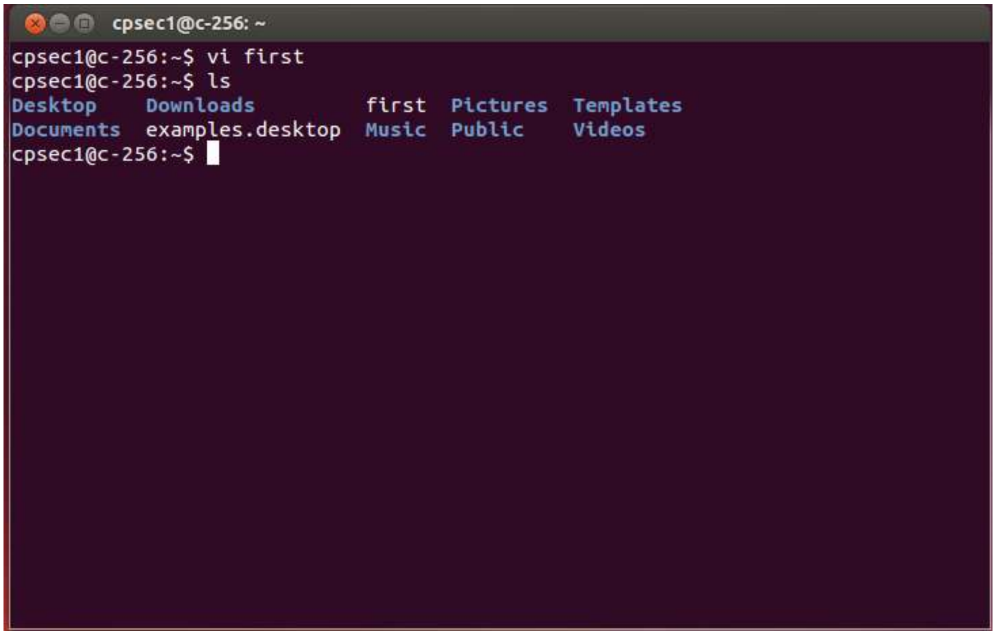
We have inserted the contents to the file '***first***'. But the content we inserted will now be in the temporary memory until we explicitly direct our editor to save it. This direction can be given only if the editor is in command mode. But the editor is in insert mode now. To bring the editor back to command mode, press the ESC key.

Now issue the command to save the contents seen in the screen shown in Fig. 3 to save in the file '***first***'. Press **:w** and then press the enter key.

Observe the (bottom of the) editor now. It reads 2 Lines, 44 Characters are written into the file. The VI editor is still in the command mode. If you want to continue editing the file, press 'i' and continue edit the file.

Now let us try to quit from the VI program by typing **:q**. Once you quit VI editor program, use ls command to see the contents of the current directory. The file name '***first***' is available in the current directory as in Fig. 4.

Save and Quit both can be done simultaneously with a single command by typing **:x** OR by typing **:wq** and press Enter key. It takes you to $ prompt.



```
cpsec1@c-256:~$ vi first
cpsec1@c-256:~$ ls
Desktop     Downloads         first  Pictures  Templates
Documents   examples.desktop  Music  Public    Videos
cpsec1@c-256:~$ 
```

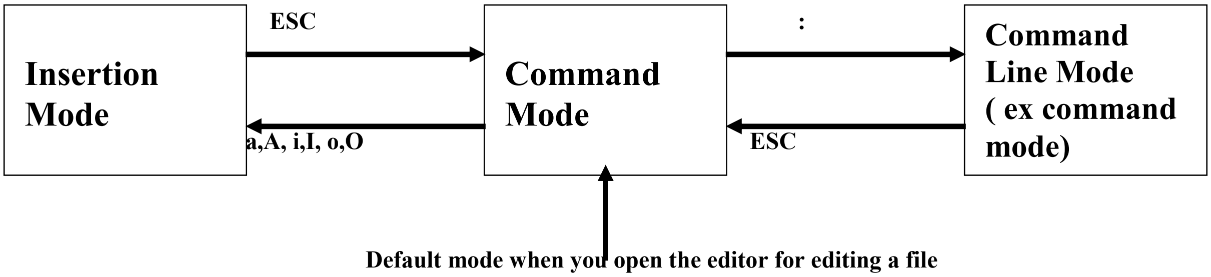
Fig-4: List of files

3. Basic Operating Modes:

We mentioned while in the process of creating our first file, that we are switching between two operating modes. The VI Editor at any instant can be found to be in one of the following 3 modes.

- Command Mode
- Command Line Mode
- Insert mode

We didn't differentiate between the command mode and command line mode when we created our first file named '***first***'. The navigation among the modes is best illustrated by the following figure.



Insertion Mode
ESC
a,A, i,I, o,O
Command Mode
:
ESC
Command Line Mode (ex command mode)
Default mode when you open the editor for editing a file

Fig-5: Modes in VI

The technical details of each of the modes are as follows:

Insert Mode:
Insert mode editing the contents of the current file, making a new file and other manipulations with the file content. All such operations are done in insert mode. VI opens in command mode. To perform the said activities on the file, we should change the current working mode to insert mode.
This can be done by pressing the keys i, a, o, O at command mode. (Table 1) Pressing ‘i’ on the keyboard at command mode allows to insert character in the page on the left side of the cursor, press ‘a’ to insert characters on the right side of the cursor.
To open a new line under the cursor, type ‘o’ at command mode and pressing ‘O’ (Uppercase letter) allows opening a new line above the cursor. Once we are changed to insert mode, any character of the keyboard can be typed and written on the screen.

Table 1: Commands for editing text

i	Inserts text to the left of the cursor.
I	Inserts text at the beginning of the line, no matter where the cursor is positioned on the current line.
a	Begins inserting after the character (append) on which the cursor is positioned.
A	Begins inserting at the end of the current line, no matter where the cursor is positioned on that line.
o	Begins inserting text on a new line below the current line
O	Begins inserting text on a new line above the current line.

Practise to use i, a, o, O with the file you have created ‘first’. While working with it, try to keep track of the current mode you are in with the help be the above figure.

Command Mode:
Once you finished entering or changing the text, you can move to command mode by pressing ESC key. Unlike in Insertion mode, when a key is pressed in the command mode, it doesn’t show up on the screen but it performs some function based on the key pressed. Refer to the Table-2 for the basic cursor movement commands and Table-1 for commands for editing a text.

Table: 2 Basic Cursor Movement commands

h	Move cursor to the left one character.
l	Move cursor to the right one character.
j	Move cursor down one line.
k	Move cursor up one line.
^	Move cursor to the beginning of the line.
$	Move cursor to the end of the current line.
G	Move cursor to the last line of your file.

Command Line Mode:
The bottom line of the vi is called command line. This mode allows giving command at the command line. (This mode is also called as ex command mode, as the commands issued at the command line are compatible with ex editor.) All commands issued in command line mode are displayed in the command line.

Let us introduce a minimal set of commands issued at command line mode here. Of course the one's introduced already

Command	**Purpose**
:w	To save your file but not quit *vi*
:w *filename*	To save the contents in the screen (buffer) into the file named *filename*.
:q	To quit if you haven't made any edits
:q!	To quit without saving the changes made in the current edit
:wq	To both quit and save your edits.

4. Commands for deleting the file contents

To delete a character/word/line, it is necessary first to position the cursor at the appropriate place. For example, when you are in command mode, pressing **x** will cause deleting the character over the cursor. Similarly, from the current cursor position, you want to delete 3 characters. This can be done by 2 ways

1. Bring the cursor over the first of n characters you want to delete. And in command mode, press the key **x**, n times. That is for deleting 3 characters from the current position; you will press the key x, 3 times.
2. Alternatively you may bring the cursor to the first of n characters you want to delete and issue the command **nx**. That is, to delete you will press **3x**. This command will delete 3 characters from the current cursor position.

The following table (Table-3) shows the commands issued at command mode to perform deletion.

Table: 3 Commands for deleting characters

x	Deletes the character at the current cursor position
X	Deletes the character left to the cursor
dw	Deletes a word from the cursor to the next space, or to the next punctuation
dd	Deletes the current line
nx, ndw, ndd	Deletes n characters, n words, n lines respectively
d0	Deletes the current line from the cursor to the beginning of the line
d$	Deletes the current line from the cursor to the end of the line

5. Miscellaneous VI Commands

Here are some more VI commands which will be useful at this level. Refer Table-4.

Table-4: Commands for deleting characters

R	Replaces the content of the file with whatever you type from the position where the cursor is initially placed. This command brings you to insert mode, where the only editing operation possible is to replacing the contents over the cursor
.	Repeats the action performed by the last command
u	Undoes the effect of the last command
U	Restores the changes to the current line since you moved the cursor to this line
w	Move to the next word
b	Move to the previous word
J	Joins the line immediately below the current line with the current line
~	Changes the character at current position from uppercase to lower case or vice versa

6. Copying and Pasting Text:

For experimenting with various copy and paste options, create a file named lab2 (vi lab2) with the following contents

woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep

The commands for copy/paste are issued in command mode. Make sure that you are in command line mode. If not, first switch to command mode by pressing Esc key. Refer Table 5 for various commands for copy and paste.

Table 5: Commands for Copy and Paste

Command	Explanation
yy	Copy (yank) the current line
Nyy	Copy N lines, beginning with the current line (Example 6yy)
yw	Copy (yank) the current word
p	Put the text copied already in next line
P	Put the text copied already before in previous line

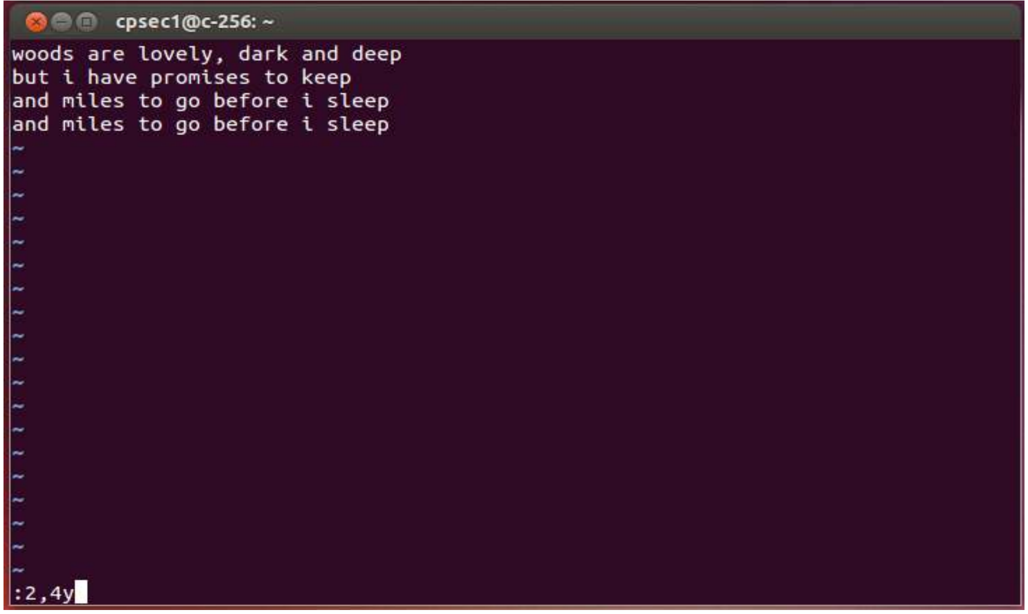
> Once you issue the copy command (yy, 2yy, 4yy etc), the copied text goes into a temporary memory area (buffer) that is replaced each time you copy (or delete) more text. Only the current contents of the temporary buffer can be put back into your file. As a result, when you use copy (y), use the put (p) command immediately.

yank (Copy) and put procedure using colon commands:

yank (Copy) procedure using colon command:

:n1,n2y where n1 specifies the beginning line number and n2 specifies the ending line number (both inclusive). Example **:2,4y copies lines 2, 3 and 4**.

Screenshot in Figure 6 shows how to issue the yank command using colon operator in terminal window. After copying line(s) you can use **p**, **P** or **:put** to paste the copied lines.



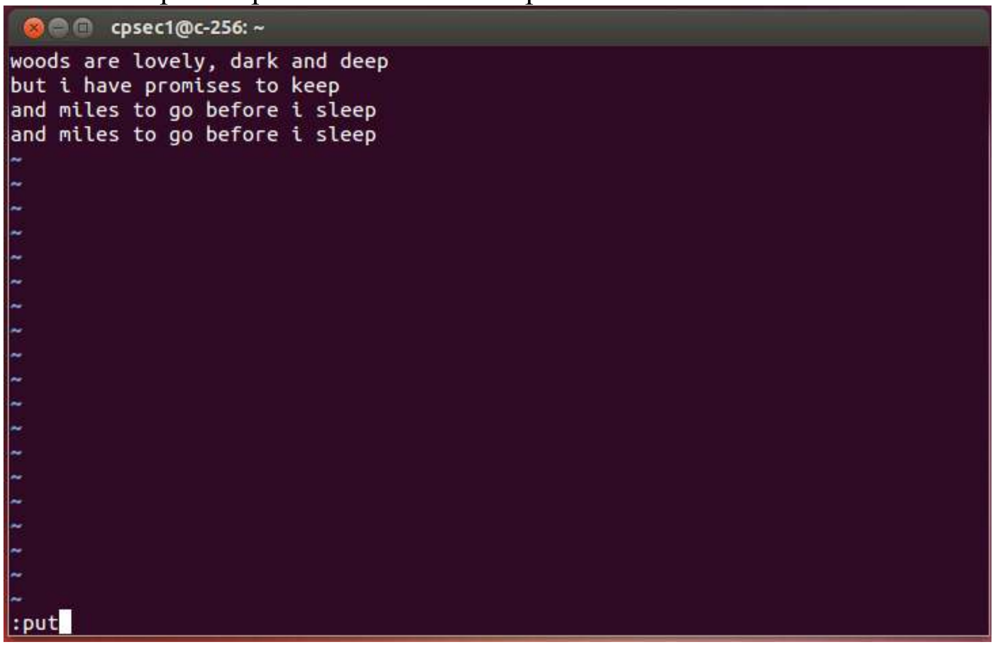
```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:2,4y
```

Figure 6: Yank (Copy) lines 2 to 4 using :2,4y

put procedure using colon command

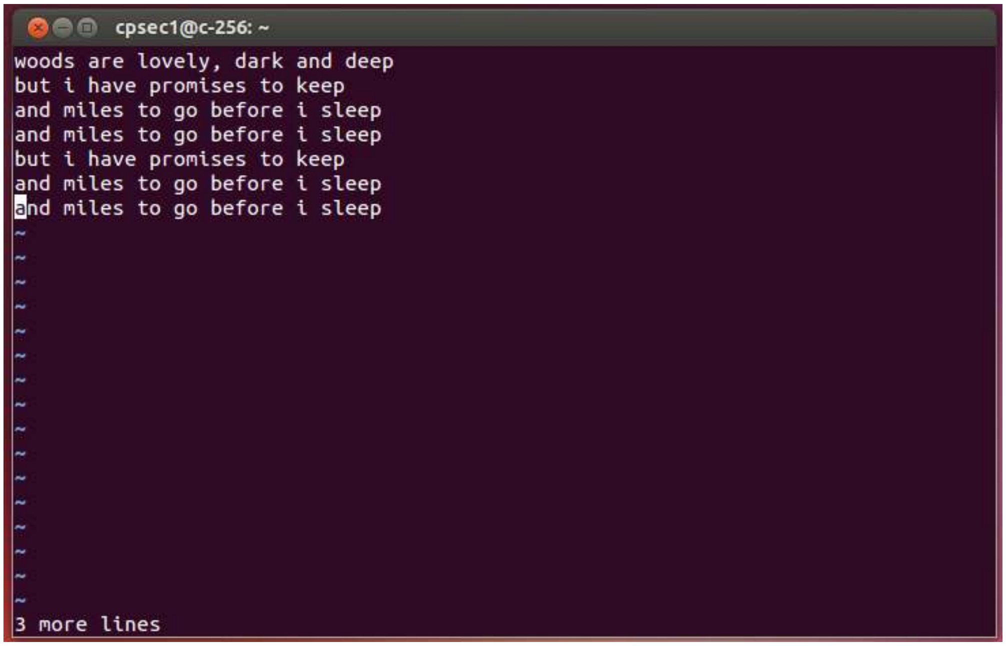
After copying the line(s) using **yy**, **Nyy**, or **:n1,n2y**, move the cursor to the desired location for pasting. Use the put procedure as shown in Figure 7.

:put It puts copied text after cursor position.



```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:put
```

Figure 7: Paste the copied text at the desired cursor position using :put



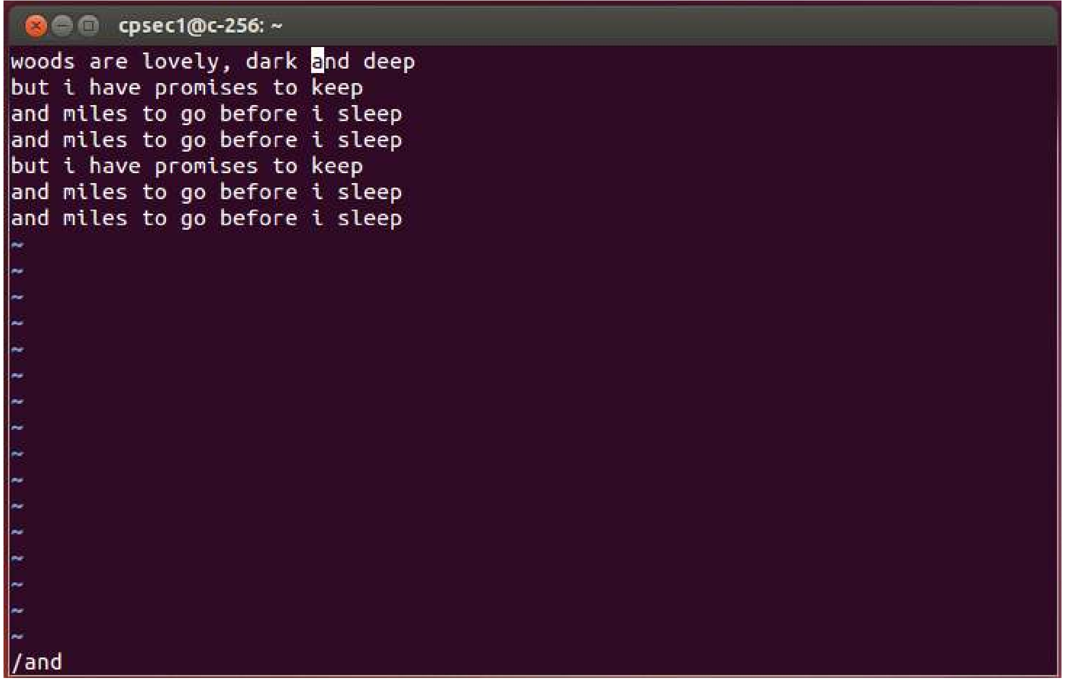
```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
3 more lines
```

Figure 8: The resultant text after executing :2,4y and :put commands

7. Pattern Searching:

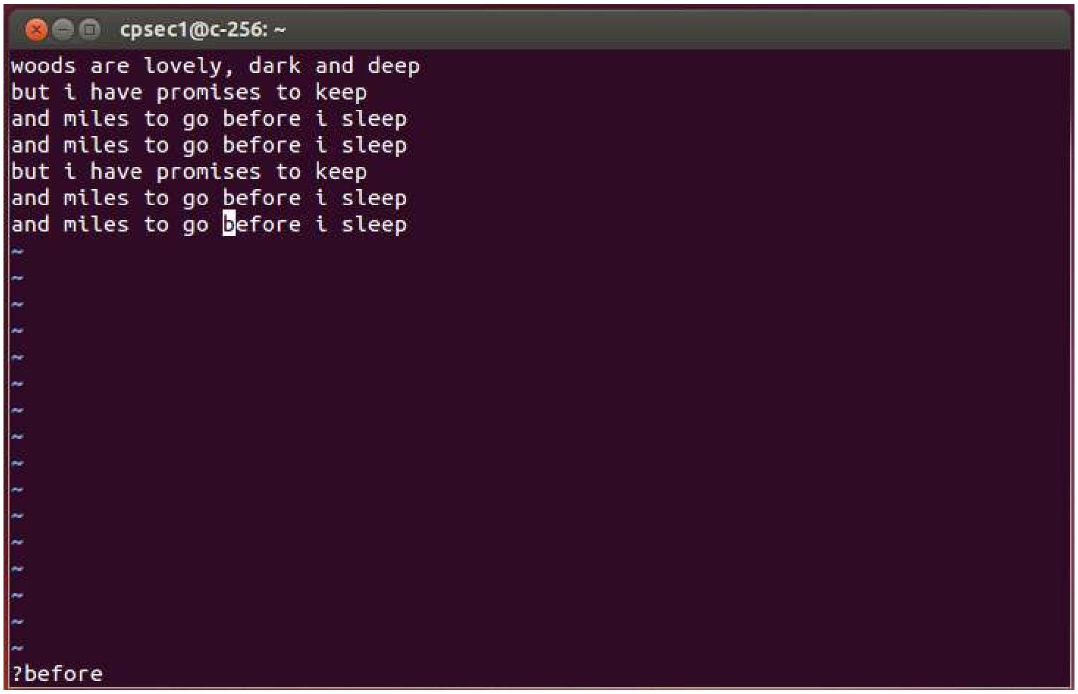
Searching a string or a character is possible with VI in command mode. For a string search, the / and ? commands are used. When you start these commands, the command just typed will be shown on the bottom line as shown in Figure 9 where you type the particular string to look for. These two commands differ only in the direction where the search takes place. The / command searches forward (downward) direction in the file, while the ? command searches backward (upward) direction in the file. List of commands are as follows:

Command	**Explanation**
/text	Search forward (down) for text (Example shown in Figure 9)
?text	Search backward (up) for text (Example shown in Figure 10)
n	Repeat last search in the same direction
N	Repeat last search in the opposite direction
f<char>	Search forward for a character on current line (Example fe)
F<char>	Search backward for a character on current line (Example Fe)
;	Repeat last character search in the same direction
%	Find matching ( ), [ ], or { }



```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
/and
```

Figure 9: Searching text ‘and’ in forward direction



```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
?before
```

Figure 10: searching text ‘before ’ in backward direction

8. Substitution of Text

If you want to do substitutions over a range of lines, or throughout the file, the **s** command is used as follows:

:n1,n2s/oldtext/newtext/gc

The oldtext is replaced by newtext from lines n1 to n2

n1 is the beginning line number

n2 is the ending line number

s means to substitute text matching the pattern (**oldtext**) with text specified by (**newtext**)

g stands for global (optional). Indicates you want to substitute all occurrences on the indicated lines. If you don't use g, the editor substitutes only the first occurrence on the indicated lines.

c stands for confirm (optional). It indicates you want to confirm each substitution before vi completes it.

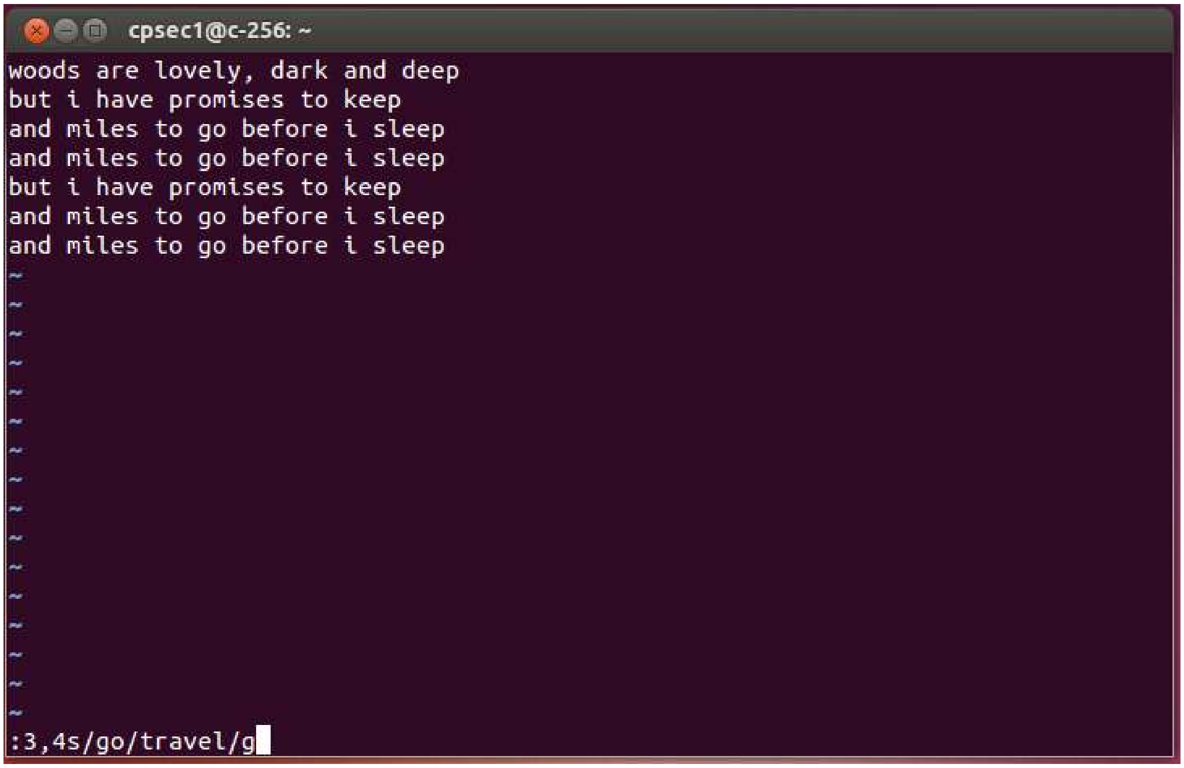
Try the following examples:

Example 1 **:3,4s/go/travel/g**

Figure 11 shows the text before substitution and Figure 12 shows the text after substitution.

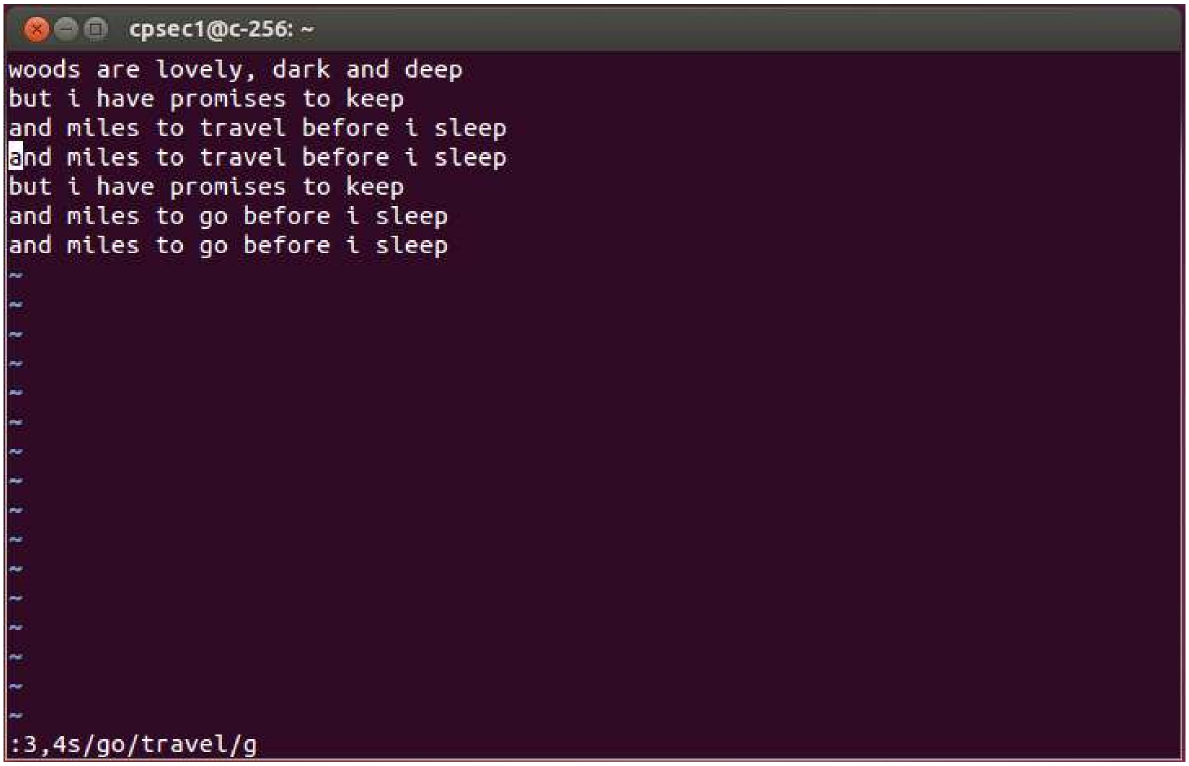
Example2 **:2,4s/i/we/gc**

(Observe the use of option c)



```
woods are lovely, dark and deep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:3,4s/go/travel/g
```

Figure 11: The text before substitution (We will substitute go with travel in lines 3 and 4)



```
woods are lovely, dark and deep
but i have promises to keep
and miles to travel before i sleep
and miles to travel before i sleep
but i have promises to keep
and miles to go before i sleep
and miles to go before i sleep
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
:3,4s/go/travel/g
```

Figure 12: After Substitution. Substitutes go with travel in lines 3 and 4.

Some more commands:

:%s/oldtext/newtext/g

Substitutes oldtext with newtext throughout the file

Example :%s/to/for/g

:.,$s/oldtext/newtext/g

Substitutes oldtext with newtext from current cursor position to the end of the file

Example :.,$s/for/to/g

:1,.s/oldtext/newtext/g

Substitutes oldtext with newtext from beginning of the file to the current cursor position

Example :^,.s/for/to/g

:&

Repeats the last substitute (:s) command

9. Exercises

Q 1) Prepare a file containing the given 10 lines below and perform the operations stated below (each line in the following text should be a line ending with an enter key in your file).

I had no where to turn, had no where to go.
this is just something I think you need to know
I don't know what made me trust you
I still remember the day when I told you what I've been through
I thought I should run away, go hide in a hole
but then you really brought out my true soul

(A) Copy (Yank) the first 2 lines in your file and paste it after line 4 in your file using Nyy and p commands.

(B) Copy (Yank) the lines 8 to 10 (both inclusive) in your file and paste it before line 1 in your file using :n1,n2y and P commands.

(C) Copy (Yank) the lines 4 and 5 in your file and paste it after line 5 in your file using Nyy and :put commands.

(D) Copy (Yank) the first 3 lines in your file and paste it after the last line your file using :n1,n2yand :put commands.

Q 2) Create a file named exer1 with following lines (each line in the following text should be a line ending with an enter key in your file) and perform the operations stated below.

Raise the door or just take off the cover,
there she sits just waiting for us to hover.

Our first glance we know she waits,
just as our mind anticipates.

As we sit on the seat, a turn of the key,
we know for sure this is the way it should be.

(A) Copy (Yank) the last 2 lines in your file and paste it after line 2 in your file using Nyy and p commands

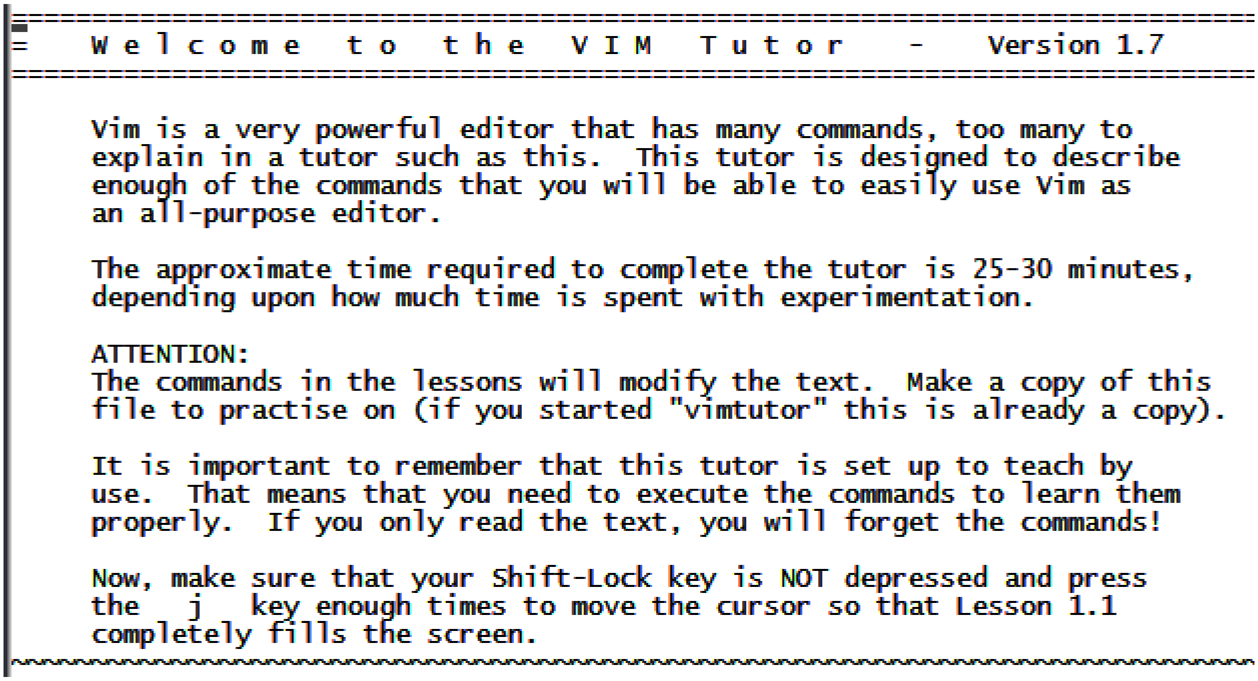
(B) Copy (Yank) the lines 3 to 5 (both inclusive) in your file and paste it before line 2 in your file using :n1,n2y and P commands

(C) Copy (Yank) the lines 4 and 5 in your file and paste it after line 4 in your file using Nyy and :put commands

(D) Copy (Yank) the first 3 lines in your file and paste it after the last line your file using :n1,n2yand :put commands.

10. Additional Reading

Most of Linux systems have command “vimtutor”, which is basically small program allowing user to practice basic vi commands. To use it, type *vimtutor* and press Enter. Following dialogue appears →



```
===============================================================================
=    W e l c o m e   t o   t h e   V I M   T u t o r    -    Version 1.7      =
===============================================================================

     Vim is a very powerful editor that has many commands, too many to
     explain in a tutor such as this.  This tutor is designed to describe
     enough of the commands that you will be able to easily use Vim as
     an all-purpose editor.

     The approximate time required to complete the tutor is 25-30 minutes,
     depending upon how much time is spent with experimentation.

     ATTENTION:
     The commands in the lessons will modify the text.  Make a copy of this
     file to practise on (if you started "vimtutor" this is already a copy).

     It is important to remember that this tutor is set up to teach by
     use.  That means that you need to execute the commands to learn them
     properly.  If you only read the text, you will forget the commands!

     Now, make sure that your Shift-Lock key is NOT depressed and press
     the   j   key enough times to move the cursor so that Lesson 1.1
     completely fills the screen.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

The tutorial is self-explanatory and consists of practice examples which can be completed while reading the document simultaneously. Go through the document to complete the exercises as per given instructions.

How to recover your work if something goes wrong!!!

The VI editor edits a temporary copy of your file, and after the editing is complete, or when you tell it to save, it puts the contents of the temporary copy into the original file. If something goes wrong while you are editing your file, the VI editor will attempt to save whatever work you had in progress, and store it for later recovery. If you were editing the file ***first***, and you accidentally got logged out, then the **-r** option of the vi editor helps.

Use the following command to open the file:

```
vi -r first
```

It will show you the temporary file options for recovery.

The **-r** option stands for recovery.

After using the **-r** option once, though, you MUST save what you have recovered to the actual file.