UNIVERSITY INSTITUTE OF ENGINEERING
&
TECHNOLOGY,

MAHARSHI DAYANAND UNIVERSITY, ROHTAK

PRACTICAL FILE

OPERATING SYSTEM

SUBMITTED TO: **SUBMITTED BY:**
Dr. KAMNA SOLANKI
(Assistant Professor CSE Deptt.)

INDEX

Sr. No.	*Program Name*	*Teacher's Sign*
1	*History of UNIX*	
2	*Description of UNIX Commands*	
3	*Introduction to VI editor*	
4	*Shell program to print a sentence.*	
5	*Shell program to add two numbers entered by user.*	
6	*Shell program to perform the arithmetic operations.*	
7(a)	*Shell program to demonstrate the use of loops.*	
7(b)	*Shell program to create Fibonacci series*	
7(c)	*Shell program to find factorial of the number entered by user.*	
7(d)	*Shell program to print counting up to the number entered by user.*	
7(e)	*Shell program to print table of a number entered by user.*	
8(a)	*Shell program to find the greatest number among three numbers.*	
8(b)	*Shell program to check whether the number entered by user is odd or even.*	
8(c)	*Shell program to check whether the number entered by user is prime or not.*	

8(d)	***Shell program to check whether the two numbers entered by user are equal or not.***	
9	***Shell program to print the consecutive numbers.***	
10	***Shell program to print the decade numbers.***	
11	***Shell program to find the simple interest of the values entered by the user.***	
12	***Shell program to check whether the entered alphabet is a vowel or not.***	
13	***Shell program to find the square of the number entered by the user.***	
14(a)	***Shell program to reverse consecutive numbers entered***	
14(b)	***Shell program to reverse a number itself***	
15	***Shell program to check whether the string entered by user is palindrome or not.***	

PROGRAM 1

HISTORY OF UNIX

Since it began to escape from AT&T's Bell Laboratories in the early 1970's, the success of the UNIX operating system has led to many different versions: recipients of the (at that time free) UNIX system code all began developing their own different versions in their own, different, ways for use and sale. Universities, research institutes, government bodies and computer companies all began using the powerful UNIX system to develop many of the technologies which today are part of a UNIX system.

Computer aided design, manufacturing control systems, laboratory simulations, even the Internet itself, all began life with and because of UNIX systems. Today, without UNIX systems, the Internet would come to a screeching halt. Most telephone calls could not be made, electronic commerce would grind to a halt and there would have never been "Jurassic Park"!

By the late 1970's, a ripple effect had come into play. By now the under- and post-graduate students whose lab work had pioneered these new applications of technology were attaining management and decision-making positions inside the computer system suppliers and among its customers. And they wanted to continue using UNIX systems.

Soon all the large vendors, and many smaller ones, were marketing their own, diverging, versions of the UNIX system optimized for their own computer architectures and boasting many different strengths and features. Customers found that, although UNIX systems were available everywhere, they seldom were able to interwork or co-exist without significant investment of time and effort to make them work effectively. The trade mark UNIX was ubiquitous, but it was applied to a multitude of different, incompatible products.

In the early 1980's, the market for UNIX systems had grown enough to be noticed by industry analysts and researchers. Now the question was no longer "What is a UNIX system?" but "Is a UNIX system suitable for business and commerce?"

Throughout the early and mid-1980's, the debate about the strengths and weaknesses of UNIX systems raged, often fuelled by the utterances of the vendors themselves who sought to protect their profitable proprietary system sales by talking UNIX systems down. And, in an effort to further differentiate their competing UNIX system products, they kept developing and adding features of their own.

In 1984, another factor brought added attention to UNIX systems. A group of vendors concerned about the continuing encroachment into their markets and control of system interfaces by the larger companies, developed the concept of "open systems."

Open systems were those that would meet agreed specifications or standards. This resulted in the formation of X/Open Company Ltd whose remit was, and today in the guise of The Open Group remains, to define a comprehensive open systems environment. Open systems, they declared, would save on costs, attract a wider portfolio of applications and competition on equal terms. X/Open chose the UNIX system as the platform for the basis of open systems.

Although UNIX was still owned by AT&T, the company did little commercially with it until the mid-1980's. Then the spotlight of X/Open showed clearly that a single, standard version of the UNIX system would be in the wider interests of the industry and its customers. The question now was, "which version?".

In a move intended to unify the market in 1987, AT&T announced a pact with Sun Microsystems, the leading proponent of the Berkeley derived strain of UNIX. However, the rest of the industry viewed the development with considerable concern. Believing that their own markets were under threat they clubbed together to develop their own "new" open systems operating system. Their new organization was called the Open Software Foundation (OSF). In response to this, the AT&T/Sun faction formed UNIX International.

The ensuing "UNIX wars" divided the system vendors between these two camps clustered around the two dominant UNIX system technologies: AT&T's System V and the OSF system called OSF/1. In the meantime, X/Open Company held the center ground. It continued the process of standardizing the APIs necessary for an open operating system specification.

In addition, it looked at areas of the system beyond the operating system level where a standard approach would add value for supplier and customer alike, developing or adopting specifications for languages, database connectivity, networking and mainframe interworking. The results of this work were published in successive X/Open Portability Guides.

XPG 4 was released in October 1992. During this time, X/Open had put in place a brand program based on vendor guarantees and supported by testing. Since the publication of XPG4, X/Open has continued to broaden the scope of open systems specifications in line with market requirements. As the benefits of the X/Open brand became known and understood, many large organizations began using X/Open as the basis for system design and procurement. By 1993, over $7 billion had been spent on X/Open branded systems. By the start of 1997 that figure has risen to over $23 billion. To date, procurements referencing the Single UNIX Specification amount to over $5.2 billion.

In early 1993, AT&T sold it UNIX System Laboratories to Novell which was looking for a heavyweight operating system to link to its NetWare product range. At the same time, the company recognized that vesting control of the definition (specification) and trademark with a vendor-neutral organization would further facilitate the value of UNIX as a foundation of open systems. So the constituent parts of the UNIX System (source code/technology and specification/trademark), previously owned by a single entity are now quite separate

In 1995 X/Open introduced the UNIX 95 brand for computer systems guaranteed to meet the Single UNIX Specification. The Single UNIX Specification brand program has now achieved critical mass: vendors whose products have met the demanding criteria now account for the majority of UNIX systems by value.

For over twenty years, since the inception of X/Open, UNIX had been closely linked with open systems. X/Open, now The Open Group, continues to develop and evolve the Single UNIX Specification and associated brand program on behalf of the IT community. The freeing of the specification of the interfaces from the technology is allowing many systems to support the UNIX philosophy of small, often simple tools , that can be combined in many ways to perform often complex tasks. The stability of the core interfaces preserves existing investment, and is allowing development of a rich set of software tools. The Open Source movement is building on this stable foundation and is creating a resurgence of enthusiasm for the UNIX philosophy. In many ways Open Source can be seen as the true delivery of Open Systems that will ensure it continues to go from strength to strength.

CHART

1969	**The Beginning**	The history of UNIX starts back in 1969, when Ken Thompson, Dennis Ritchie and others started working on the "little-used PDP-7 in a corner" at Bell Labs and what was to become UNIX.
1971	**First Edition**	It had a assembler for a PDP-11/20, file system, fork(), roff and ed. It was used for text processing of patent

		documents.
1973	**Fourth Edition**	It was rewritten in C. This made it portable and changed the history of OS's.
1975	**Sixth Edition**	UNIX leaves home. Also widely known as Version 6, this is the first to be widely available out side of Bell Labs. The first BSD version (1.x) was derived from V6.
1979	**Seventh Edition**	It was a "improvement over all preceding and following Unices" [Bourne]. It had C, UUCP and the Bourne shell. It was ported to the VAX and the kernel was more than 40 Kilobytes (K).
1980	**Xenix**	Microsoft introduces Xenix. 32V and 4BSD introduced.
1982	**System III**	AT&T's UNIX System Group (USG) release System III, the first public release outside Bell Laboratories. SunOS 1.0 ships. HP-UX introduced. Ultrix-11 Introduced.
1983	**System V**	Computer Research Group (CRG), UNIX System Group (USG) and a third group merge to become UNIX System Development Lab. AT&T announces UNIX System V, the first supported release. Installed base 45,000.
1984	**4.2BSD**	University of California at Berkeley releases 4.2BSD, includes TCP/IP, new signals and much more. X/Open formed.
1984	**SVR2**	System V Release 2 introduced. At this time there are 100,000 UNIX installations around the world.
1986	**4.3BSD**	4.3BSD released, including internet name server. SVID introduced. NFS shipped. AIX announced. Installed base 250,000.
1987	**SVR3**	System V Release 3 including STREAMS, TLI, RFS. At this time there are 750,000 UNIX installations around the world. IRIX introduced.
1988		POSIX.1 published. Open Software Foundation (OSF) and UNIX International (UI) formed. Ultrix 4.2 ships.
1989		AT&T UNIX Software Operation formed in preparation for spinoff of USL. Motif 1.0 ships.
1989	**SVR4**	UNIX System V Release 4 ships, unifying System V, BSD and Xenix. Installed base 1.2 million.
1990	**XPG3**	X/Open launches XPG3 Brand. OSF/1 debuts. Plan 9 from Bell Labs ships.
1991		UNIX System Laboratories (USL) becomes a company -

		majority-owned by AT&T. Linus Torvalds commences Linux development. Solaris 1.0 debuts.
1992	**SVR4.2**	USL releases UNIX System V Release 4.2 (Destiny). October - XPG4 Brand launched by X/Open. December 22nd Novell announces intent to acquire USL. Solaris 2.0 ships.
1993	**4.4BSD**	4.4BSD the final release from Berkeley. June 16 Novell acquires USL
Late 1993	**SVR4.2MP**	Novell transfers rights to the "UNIX" trademark and the Single UNIX Specification to X/Open. COSE initiative delivers "Spec 1170" to X/Open for fasttrack. In December Novell ships SVR4.2MP , the final USL OEM release of System V
1994	**Single UNIX Specification**	BSD 4.4-Lite eliminated all code claimed to infringe on USL/Novell. As the new owner of the UNIX trademark, X/Open introduces the Single UNIX Specification (formerly Spec 1170), separating the UNIX trademark from any actual code stream.
1995	**UNIX 95**	X/Open introduces the UNIX 95 branding programme for implementations of the Single UNIX Specification. Novell sells UnixWare business line to SCO. Digital UNIX introduced. UnixWare 2.0 ships. OpenServer 5.0 debuts.
1996		The Open Group forms as a merger of OSF and X/Open.
1997	**Single UNIX Specification, Version 2**	The Open Group introduces Version 2 of the Single UNIX Specification, including support for realtime, threads and 64-bit and larger processors. The specification is made freely available on the web. IRIX 6.4, AIX 4.3 and HP-UX 11 ship.
1998	**UNIX 98**	The Open Group introduces the UNIX 98 family of brands, including Base, Workstation and Server. First UNIX 98 registered products shipped by Sun, IBM and NCR. The Open Source movement starts to take off with announcements from Netscape and IBM. UnixWare 7 and IRIX 6.5 ship.
1999	**UNIX at 30**	The UNIX system reaches its 30th anniversary. Linux 2.2 kernel released. The Open Group and the IEEE commence joint development of a revision to POSIX and

		the Single UNIX Specification. First LinuxWorld conferences. Dot com fever on the stock markets. Tru64 UNIX ships.
2001	**Single UNIX Specification, Version 3**	Version 3 of the Single UNIX Specification unites IEEE POSIX, The Open Group and the industry efforts. Linux 2.4 kernel released. IT stocks face a hard time at the markets. The value of procurements for the UNIX brand exceeds $25 billion. AIX 5L ships.
2003	**ISO/IEC 9945:2003**	The core volumes of Version 3 of the Single UNIX Specification are approved as an international standard. The "Westwood" test suite ship for the UNIX 03 brand. Solaris 9.0 E ships. Linux 2.6 kernel released.
2007		Apple Mac OS X certified to UNIX 03.
2008	**ISO/IEC 9945:2008**	Latest revision of the UNIX API set formally standardized at ISO/IEC, IEEE and The Open Group. Adds further APIs
2009	**UNIX at 40**	IDC on UNIX market -- says UNIX $69 billion in 2008, predicts UNIX $74 billion in 2013
2010	**UNIX on the Desktop**	Apple reports 50 million desktops and growing -- these are Certified UNIX systems.

Comparing Linux and Unix

This book uses the term “Unix-like” to describe systems intentionally like Unix. In particular, the term “Unix-like” includes all major Unix variants and Linux distributions. Note that many people simply use the term “Unix” to describe these systems instead. Originally, the term “Unix” meant a particular product developed by AT&T. Today, the Open Group owns the Unix trademark, and it defines Unix as “the worldwide Single UNIX Specification”.

Linux is not derived from Unix source code, but its interfaces are intentionally like Unix. Therefore, Unix lessons learned generally apply to both, including information on security. Most of the information in this book applies to any Unix-like system. Linux-specific information has been intentionally added to enable those using Linux to take advantage of Linux’s capabilities.

Unix-like systems share a number of security mechanisms, though there are subtle differences and not all systems have all mechanisms available. All include user and group ids (uids and gids) for each process and a filesystem with read, write, and execute permissions (for user, group, and other). See Thompson [1974] and Bach [1986] for general information on Unix systems, including their basic security mechanisms.

PROGRAM 2:

The UNIX operating system has for many years formed the backbone of the Internet, especially for large servers and most major university campuses. However, a free version of UNIX called **Linux** has been making significant gains against Macintosh and the Microsoft Windows 95/98/NT environments, so often associated with personal computers. Developed by a number of volunteers on the Internet such as the Linux group and the GNU project, much of the open-source software is copyrighted, but available for free. This is especially valuable for those in educational environments where budgets are often limited.

UNIX commands can often be grouped together to make even more powerful commands with capabilities known as **I/O redirection** (**<** for getting input from a file input and **>** for outputing to a file) and **piping** using **|** to feed the output of one command as input to the next. Please investigate manuals in the lab for more examples than the few offered here.

The following charts offer a summary of some simple UNIX commands. These are certainly not all of the commands available in this robust operating system, but these will help you get started.

Ten ESSENTIAL UNIX Commands
These are ten commands that you really need to know in order to get started with UNIX. They are probably similar to commands you already know for another operating system.

Command		Example	Description
1.	**ls**	ls ls -alF	Lists files in current directory List in long format
2.	**cd**	cd tempdir cd .. cd ~dhyatt/web-docs	Change directory to tempdir Move back one directory Move into dhyatt's web-docs directory
3.	**mkdir**	mkdir graphics	Make a directory called graphics
4.	**rmdir**	rmdir emptydir	Remove directory (must be empty)
5.	**cp**	cp file1 web-docs cp file1 file1.bak	Copy file into directory Make backup of file1
6.	**rm**	rm file1.bak rm *.tmp	Remove or delete file Remove all file
7.	**mv**	mv old.html new.html	Move or rename files
8.	**more**	more index.html	Look at file, one page at a time
9.	**lpr**	lpr index.html	Send file to printer
10.	**man**	man ls	Online manual (help) about command

Ten VALUABLE UNIX Commands

Once you have mastered the basic UNIX commands, these will be quite valuable in managing your own account.

Command	Example	Description
1. **grep <str><files>**	grep "bad word" *	Find which files contain a certain word
2. **chmod <opt> <file>**	chmod 644 *.html chmod 755 file.exe	Change file permissions read only Change file permissions to executable
3. **passwd**	passwd	Change passwd
4. **ps <opt>**	ps aux ps aux | grep dhyatt	List all running processes by #ID List process #ID's running by dhyatt
5. **kill <opt> <ID>**	kill -9 8453	Kill process with ID #8453
6. **gcc (g++) <source>**	gcc file.c -o file g++ fil2.cpp -o fil2	Compile a program written in C Compile a program written in C++
7. **gzip <file>**	gzip bigfile gunzip bigfile.gz	Compress file Uncompress file
8. **mail (pine)**	mail me@tjhsst.edu < file1 pine	Send file1 by email to someone Read mail using pine
9. **telnet <host> ssh <host>**	telnet vortex.tjhsst.edu ssh -l dhyatt jazz.tjhsst.edu	Open a connection to vortex Open a secure connection to jazz as user dhyatt
10. **ftp <host>**	ftp station1.tjhsst.edu	Upload or Download

ncftp <host/directory>	ncftp metalab.unc.edu	files to station1 Connect to archives at UNC

Ten FUN UNIX Commands
These are ten commands that you might find interesting or amusing. They are actually quite helpful at times, and should not be considered idle entertainment.

Command	**Example**	**Description**
1. **who**	who	Lists who is logged on your machine
2. **finger**	finger	Lists who is on computers in the lab
3. **ytalk <user@place>**	ytalk dhyatt@threat	Talk online with dhyatt who is on threat
4. **history**	history	Lists commands you've done recently
5. **fortune**	fortune	Print random humerous message
6. **date**	date	Print out current date
7. **cal <mo> <yr>**	cal 9 2000	Print calendar for September 2000
8. **xeyes**	xeyes &	Keep track of cursor (in "background")
9. **xcalc**	xcalc &	Calculator ("background" process)
10. **mpage <opt>**	mpage -8 file1	Print 8 pages on a single sheet

<file>	**|** lpr	and send to printer (the font will be small!)

Ten HELPFUL UNIX Commands
These ten commands are very helpful, especially with graphics and word processing type applications.

Command	**Example**	**Description**
1. **netscape**	netscape &	Run Netscape browser
2. **xv**	xv &	Run graphics file converter
3. **xfig / xpaint**	xfig & (xpaint &)	Run drawing program
4. **gimp**	gimp &	Run photoshop type program
5. **ispell <fname>**	ispell file1	Spell check file1
6. **latex <fname>**	latex file.tex	Run LaTeX, a scientific document tool
7. **xemacs / pico**	xemacs (or pico)	Different editors
8. **soffice**	soffice &	Run StarOffice, a full word processor
9. **m-tools (mdir, mcopy, mdel, mformat, etc.)**	mdir a: mcopy file1 a:	DOS commands from UNIX (dir A:) Copy file1 to A:
10. **gnuplot**	gnuplot	Plot data graphically

Ten USEFUL UNIX Commands:
These ten commands are useful for monitoring system access, or simplifying your own environment.

Command	**Example**	**Description**
1. **df**	df	See how much free disk space
2. **du**	du -b subdir	Estimate disk usage of directory in Bytes
3. **alias**	alias lls="ls -alF"	Create new command "lls" for long format of ls
4. **xhost**	xhost + threat.tjhsst.edu xhost -	Permit window to display from x-window program from threat Allow no x-window access from other systems
5. **fold**	fold -s file1 **|** lpr	Fold or break long lines at 60 characters and send to printer
6. **tar**	tar -cf subdir.tar subdir tar -xvf subdir.tar	Create an archive called subdir.tar of a directory Extract files from an archive file
7. **ghostview (gv)**	gv filename.ps	View a Postscript file
8. **ping (traceroute)**	ping threat.tjhsst.edu traceroute www.yahoo.com	See if machine is alive Print data path to a machine
9. **top**	top	Print system usage and top resource hogs

10.	**logout (exit)**	logout or exit	How to quit a UNIX shell.

PROGRAM 3

General Introduction
The vi editor (short for visual editor) is a screen editor which is available on almost all Unix systems. Once you have learned vi, you will find that it is a fast and powerful editor. vi has no menus but instead uses combinations of keystrokes in order to accomplish commands. If you are just beginning to learn Unix, you might find the Pico editor easier to use (most command options are displayed at the bottom of the screen). If you use the Pine email application and have composed or replied to a message you have probably already used Pico as it is used for text entry.

Starting vi
To start using vi, at the Unix prompt type **vi** followed by a file name. If you wish to edit an existing file, type in its name; if you are creating a new file, type in the name you wish to give to the new file.
%vi *filename*
Then hit **Return**. You will see a screen similar to the one below which shows blank lines with tildes and the name and status of the file.

```
~

~

"myfile" [New file]
```

vi's Modes and Moods
vi has two modes: the command mode and the insert mode. It is essential that you know which mode you are in at any given point in time. When you are in command mode, letters of the keyboard will be interpreted as commands. When you are in insert mode the same letters of the keyboard will type or edit text. vi always starts out in command mode. When you wish to move between the two modes, keep these things in mind. You can type **i** to enter the insert mode. If you wish to leave insert mode and return to the command mode, hit the **ESC** key. If

you're not sure where you are, hit **ESC** a couple of times and that should put you back in command mode.

General Command Information

As mentioned previously, vi uses letters as commands. It is important to note that in general vi commands:

- are case sensitive - lowercase and uppercase command letters do different things
- are not displayed on the screen when you type them
- generally do not require a **Return** after you type the command.

You will see some commands which start with a colon (**:**). These commands are *ex* commands which are used by the *ex* editor. *ex* is the true editor which lies underneath vi -- in other words, vi is the interface for the ex editor.

Entering Text

To begin entering text in an empty file, you must first change from the command mode to the insert mode. To do this, type the letter **i**. When you start typing, anything you type will be entered into the file. Type a few short lines and hit **Return** at the end of each of line. Unlike word processors, vi does not use word wrap. It will break a line at the edge of the screen. If you make a mistake, you can use the Backspace key to remove your errors. If the Backspace key doesn't work properly on your system, try using the Ctrl h key combination.

Cursor Movement

You must be in command mode if you wish to move the cursor to another position in your file. If you've just finished typing text, you're still in insert mode and will need to press **ESC** to return to the command mode.

Moving One Character at a Time

Try using your direction keys to move up, down, left and right in your file. Sometimes, you may find that the direction keys don't work. If that is the case, to move the cursor one character at the time, you may use the **h**, **j**, **k**, and **l** keys. These keys move you in the following directions:

h	left one space	**l**	right one space
j	down one space	**k**	up one space

If you move the cursor as far as you can in any direction, you may see a screen flash or hear a beep.

Moving among Words and Lines

While these four keys (or your direction keys) can move you just about anywhere you want to go in your file, there are some shortcut keys that you can use to move a little more quickly through a document. To move more quickly among words, you might use the following:

w	moves the cursor forward one word
b	moves the cursor backward one word (if in the middle of a word, b will move you to the beginning of the current word).
e	moves to the end of a word.

To build on this further, you can precede these commands with a number for greater movement. For example, 5w would move you forward five words; 12b would move you backwards twelve words. [You can also use numbers with the commands mentioned earlier. For example, 5j would move you down 5 characters.]

Command Keys and Case

You will find when using vi that lower case and upper case command keys are interpreted differently. For example, when using the lower case **w**, **b**, and **e** commands, words will be defined by a space or a punctuation mark. On the other hand, **W**, **B**, and **E** commands may be used to move between words also, but these commands ignore punctuation.

Shortcuts

Two short cuts for moving quickly on a line include the **$** and the **0** (zero) keys. The **$** key will move you to the end of a line, while the **0** will move you quickly to the beginning of a line.

Screen Movement

To move the cursor to a line within your current screen use the following keys:

H	moves the cursor to the top line of the screen.
M	moves the cursor to the middle line of the screen.
L	moves the cursor to the last line of the screen.

To scroll through the file and see other screens use:

ctrl-f	scrolls down one screen
ctrl-b	scrolls up one screen
ctrl-u	scrolls up a half a screen
ctrl-d	scrolls down a half a screen

Two other useful commands for moving quickly from one end to the other of a document are **G** to move to the end of the file and **1G** to move to the beginning of the file. If you precede **G** with a number, you can move to a specific line in the document (e.g. 15G would move you to line 15).

Moving by Searching

One method for moving quickly to a particular spot in your file is to search for specific text. When you are in command mode, type a **/** followed the text you wish to search for. When you press **Return**, the cursor will move to the first incidence of that string of text. You can repeat the search by typing **n** or search in a backwards direction by using **N**.

Basic Editing

To issue editing commands, you must be in command mode. As mentioned before, commands will be interpreted differently depending upon whether they are issued in lower or upper case. Also, many of the editing commands can be preceded by a number to indicate a repetition of the command.

Deleting (or Cutting) Characters, Words, and Lines

To delete a character, first place your cursor on that character. Then, you may use any of the following commands:

x	deletes the character under the cursor.
X	deletes the character to the left of your cursor.
dw	deletes from the character selected to the end of the word.
dd	deletes all the current line.
D	deletes from the current character to the end of the line.

Preceding the command with a number will delete multiple characters. For example, **10x** will delete the character selected and the next 9 characters; **10X** will delete the 10 characters to the left of the currently selected character. The command **5dw** will delete 5 words, while **4dd** deletes four lines.

Pasting Text using Put

Often, when you delete or cut text, you may wish to reinsert it in another location of the document. The Put command will paste in the last portion of text that was deleted since deleted text is stored in a buffer. To use this command, place the cursor where you wish the deleted text to appear. Then use **p** to reinsert the text. If you are inserting a line or paragraph use the lower case **p** to insert on the line below the cursor or upper case **P** to place in on the line above the cursor.

Copying Text with Yank

If you wish to make a duplicate copy of existing text, you may use the yank and put commands to accomplish this function. Yank copies the selected text into a buffer and holds it until another yank or deletion occurs. Yank is usually used in combination with a word or line object such as the ones shown below:

yw	copies a word into a buffer (7yw copies 7 words)
yy	copies a line into a buffer (3yy will copy 3 lines)

Once the desired text is yanked, place the cursor in the spot in which you wish to insert the text and then use the put command (**p** for line below or **P** for line above) to insert the contents of the buffer.

Replacing or Changing Characters, Words, and Lines
When you are using the following commands to replace text, you will be put temporarily into insert mode so that you can change a character, word, line, or paragraph of text.

r	replaces the current character with the next character you enter/type. Once you enter the character you are returned to command mode.
R	puts you in overtype mode until you hit **ESC** which will then return you to command mode.
cw	changes and replaces the current word with text that you type. A dollar sign marks the end of the text you're changing. Pressing **ESC** when you finish will return you to command mode.

Inserting Text
If you wish to insert new text in a line, first position the cursor to the right of where you wish the inserted text to appear. Type **i** to get into insert mode and then type in the desired text (note that the text is inserted before the cursor). Press **ESC** to return to command mode.

Inserting a Blank Line
To insert a blank line below the line your cursor is currently located on, use the **o** key and then hit **ESC** to return to the command mode . Use **O** to insert a line above the line the cursor is located on.

Appending Text
You can use the append command to add text at any place in your file. Append (**a**) works very much like Insert (**i**) except that it insert text *after* the cursor rather than before it. Append is probably used most often for adding text to the end of a line. Simply place your cursor where you wish to append text and press **a.** Once you've finished appending, press **ESC** to go back to command mode.

Joining Lines

Since vi does not use automatic word wrap, it is not unusual in editing lines to end up with lines that are too short and that might be improved if joined together. To do this, place your cursor on the first line to be joined and type **J**. As with other commands, you can precede **J** with a number to join multiple lines (**4J** joins 4 lines).

Undoing

Be sure to remember this command. When you make a mistake you can undo it. **DO NOT** move the cursor from the line where you made the change. Then try using one of the following two commands:

u	undoes the last change you made anywhere in the file. Using **u** again will "undo the undo".
U	undoes all recent changes to the current line. You can not have moved from the line to recover the original line.

Closing and Saving Files

When you edit a file in vi, you are actually editing a copy of the file rather than the original. The following sections describe methods you might use when closing a file, quitting vi, or both.

Quitting and Saving a File

The command **ZZ** (notice that it is in uppercase) will allow you to quit vi and save the edits made to a file. You will then return to a Unix prompt. Note that you can also use the following commands:

:w	to save your file but not quit vi (this is good to do periodically in case of machine crash!).
:q	to quit if you haven't made any edits.
:wq	to quit and save edits (basically the same as ZZ).

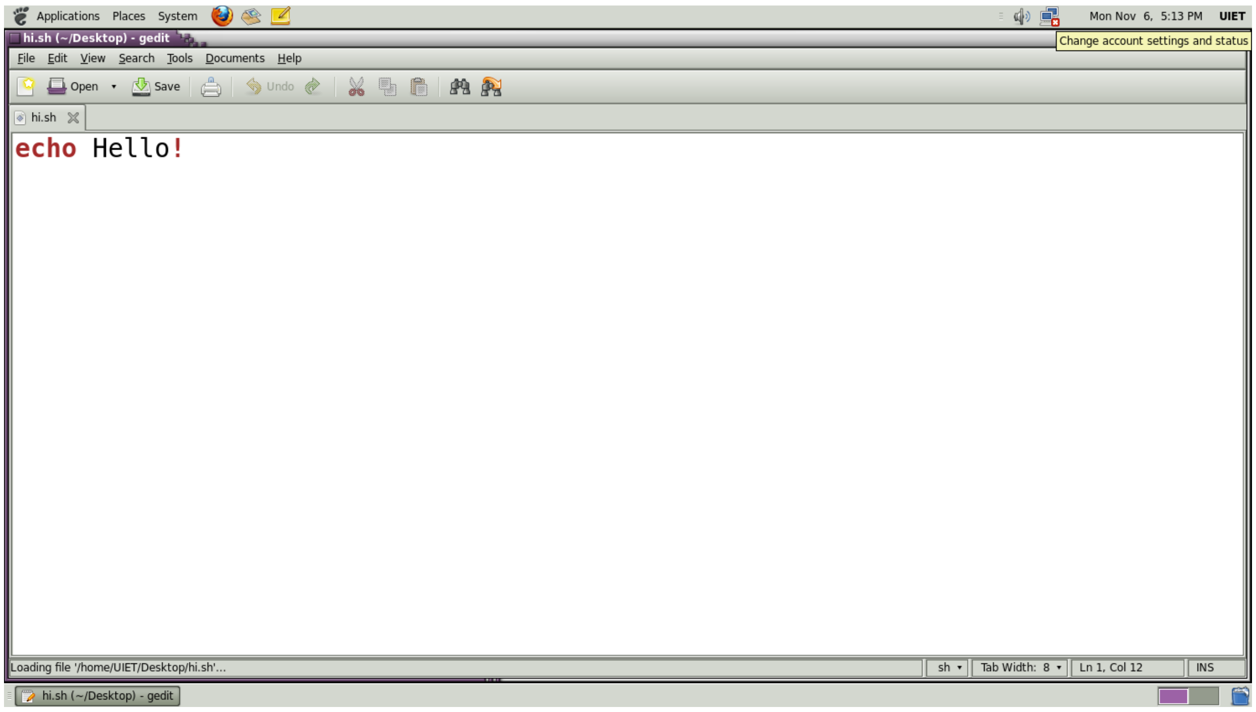
Quitting without Saving Edits

Sometimes, when you create a mess (when you first start using vi this is easy to do!) you may wish to erase all edits made to the file and either start over or quit. To do this, you can choose from the following two commands:

:e!	reads the original file back in so that you can start over.
:q!	wipes out all edits and allows you to exit from vi.

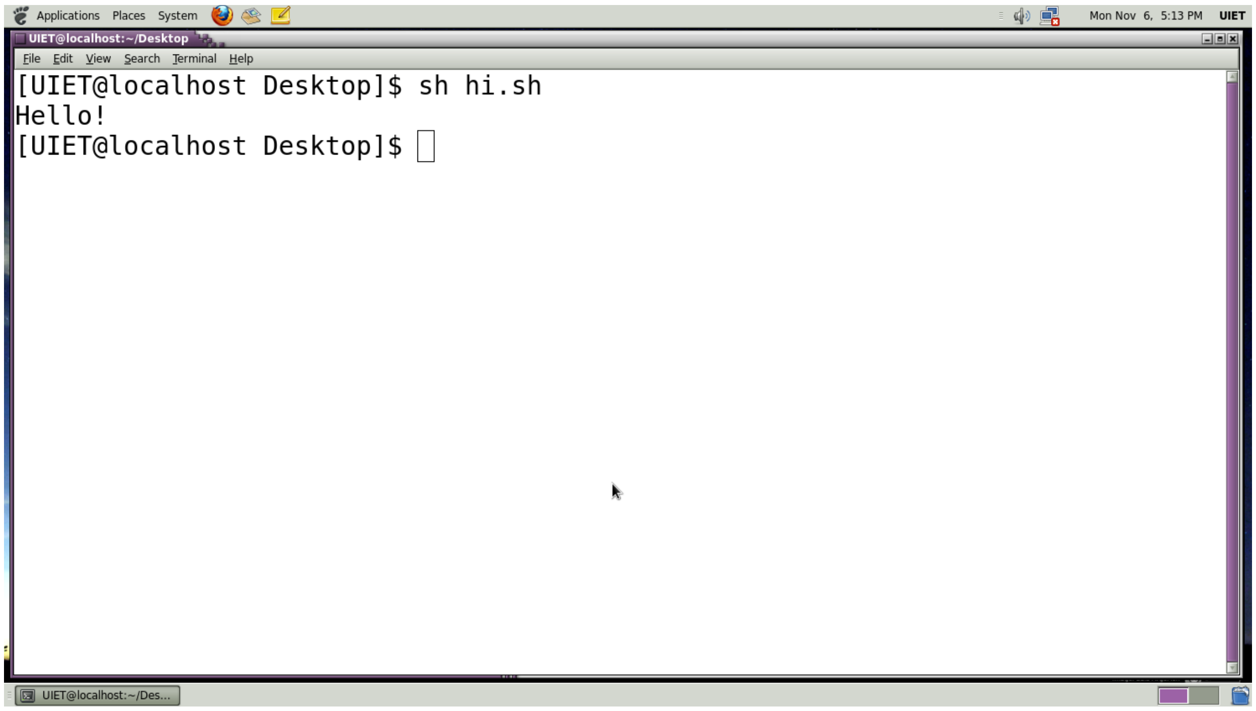
PROGRAM 4

Shell program to print a Sentence


Applications Places System
Mon Nov 6, 5:13 PM UIET
hi.sh (~/Desktop) - gedit
Change account settings and status
File Edit View Search Tools Documents Help
Open Save Undo
hi.sh

```
echo Hello!
```

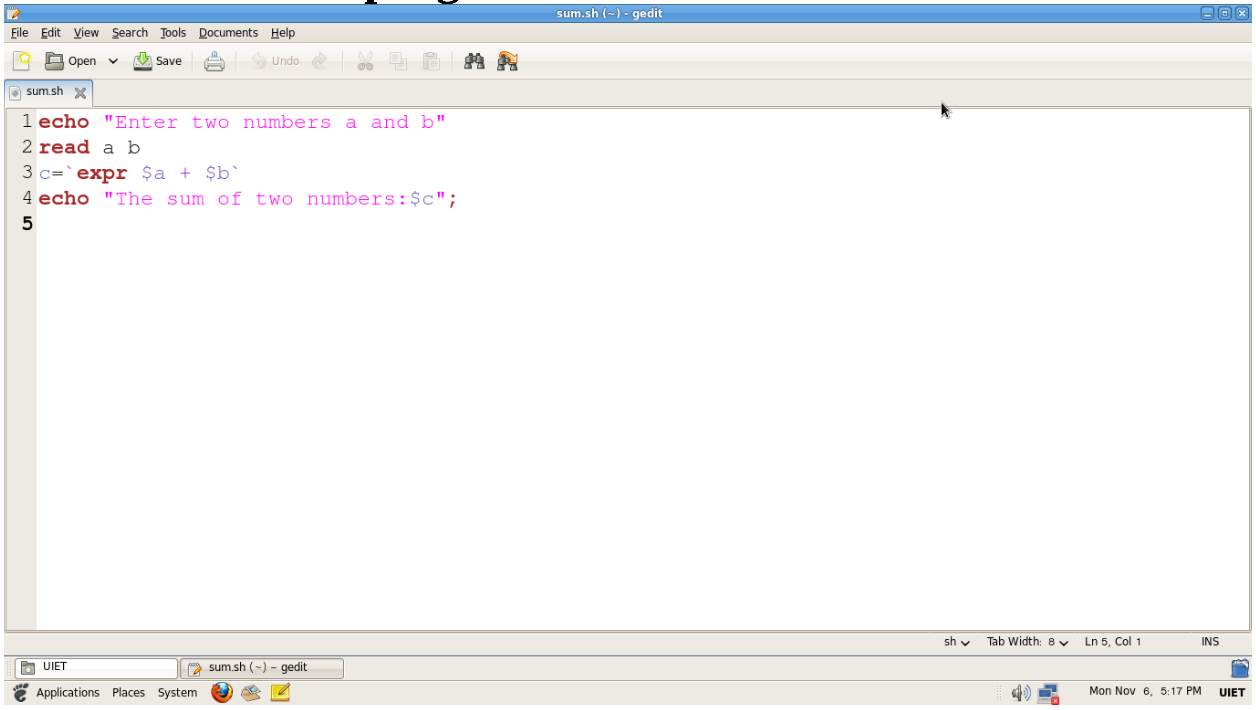
Loading file '/home/UIET/Desktop/hi.sh'...
sh
Tab Width: 8
Ln 1, Col 12
INS
hi.sh (~/Desktop) - gedit



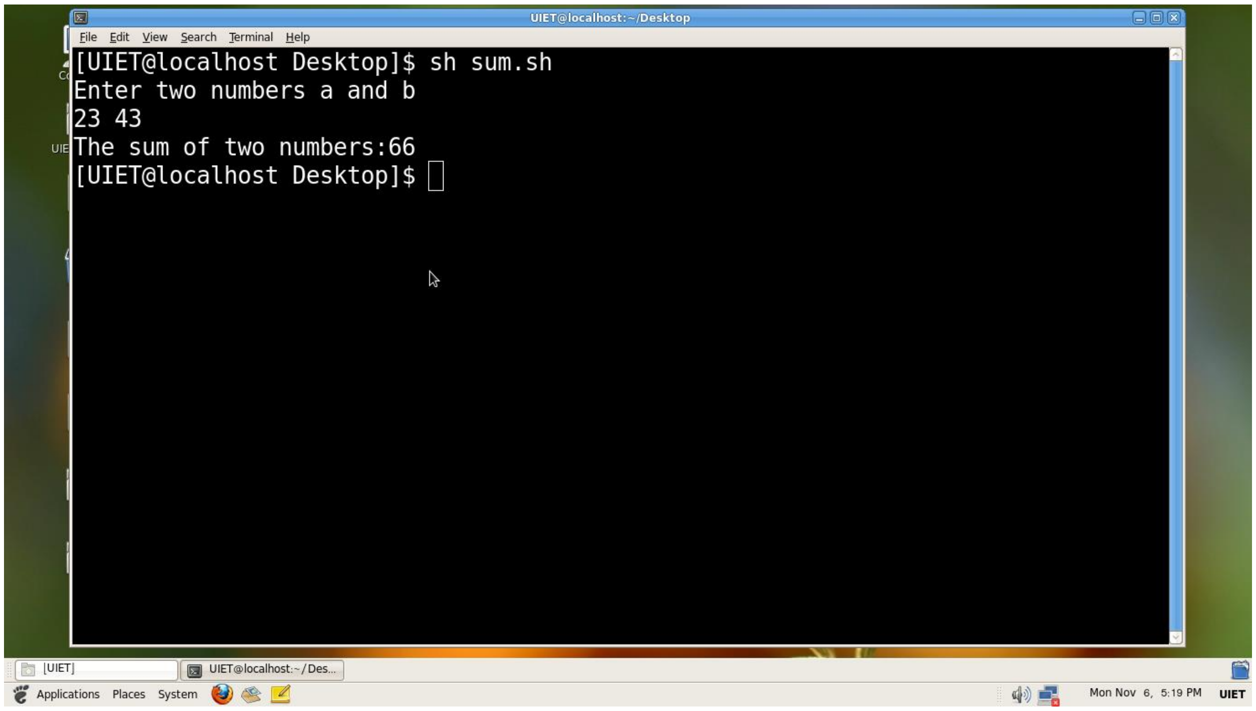
```
[UIET@localhost Desktop]$ sh hi.sh
Hello!
[UIET@localhost Desktop]$ 
```

PROGRAM 5

Shell program for Sum of 2 numbers


sum.sh (~) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
sum.sh
sh Tab Width: 8 Ln 5, Col 1 INS
UIET sum.sh (~) - gedit
Applications Places System
Mon Nov 6, 5:17 PM UIET

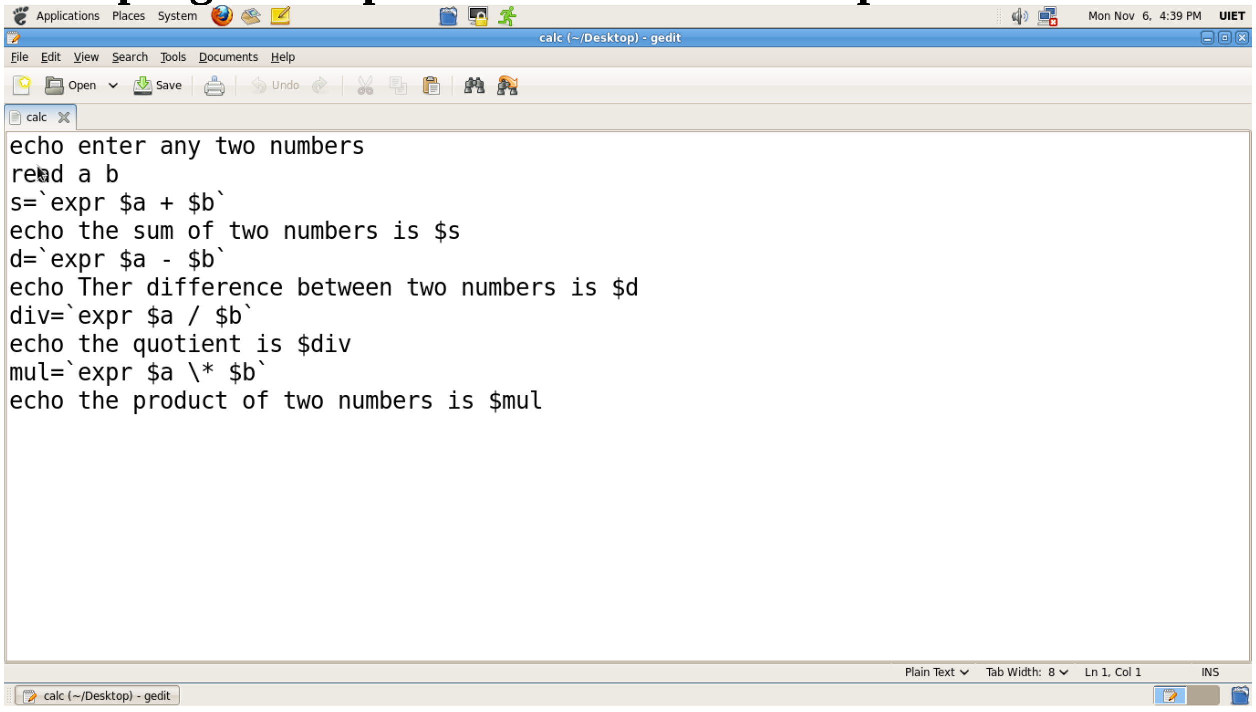
```
echo "Enter two numbers a and b"
read a b
c=`expr $a + $b`
echo "The sum of two numbers:$c";
```



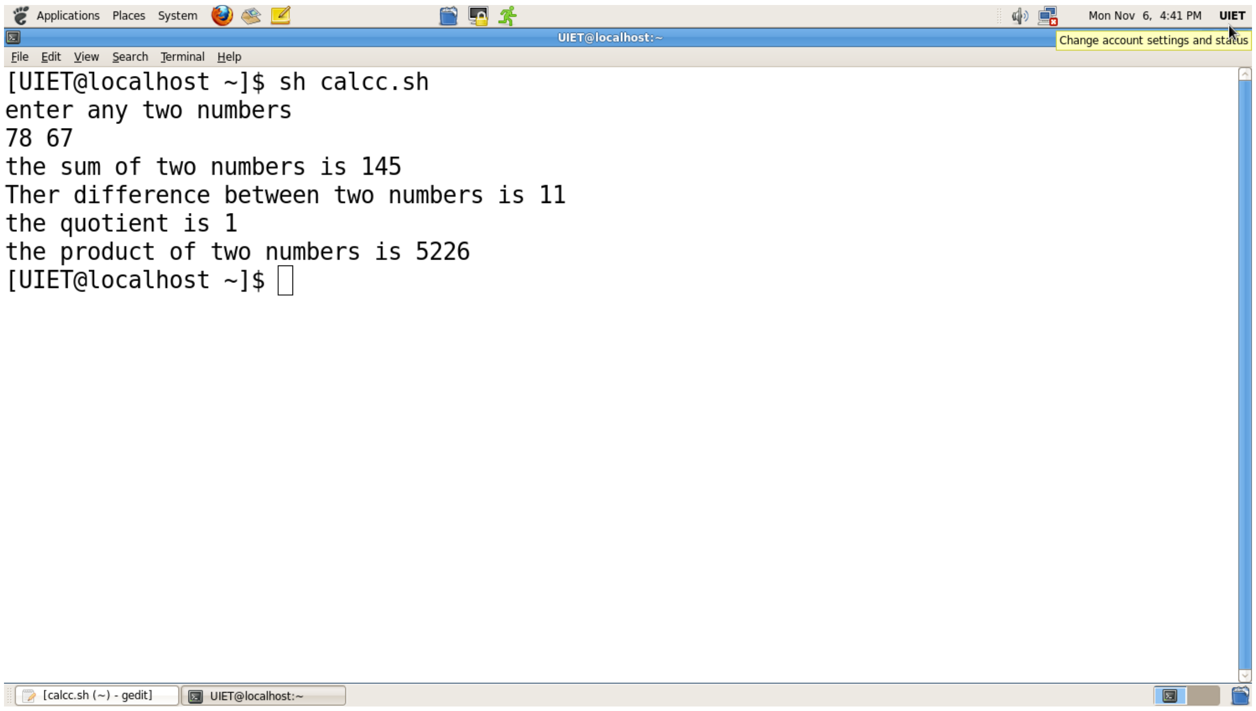
```
[UIET@localhost Desktop]$ sh sum.sh
Enter two numbers a and b
23 43
The sum of two numbers:66
[UIET@localhost Desktop]$
```

PROGRAM 6

Shell program to perform all arithmetic operations



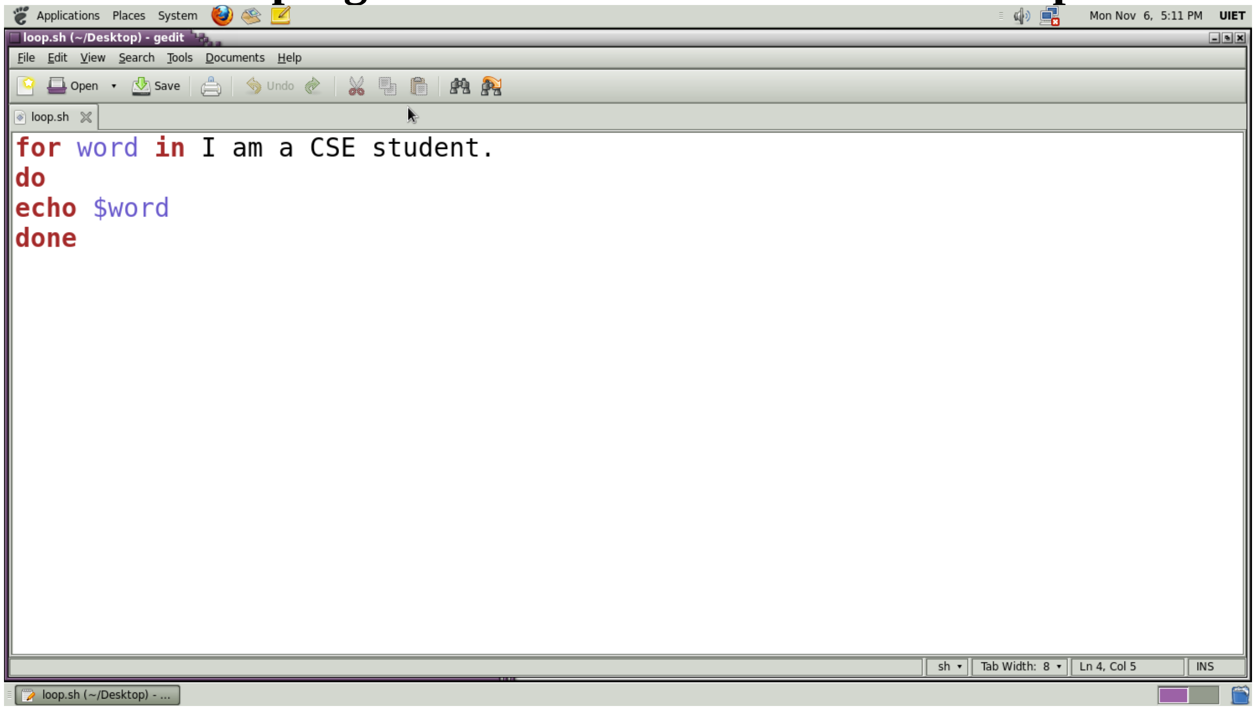
```
echo enter any two numbers
read a b
s=`expr $a + $b`
echo the sum of two numbers is $s
d=`expr $a - $b`
echo Ther difference between two numbers is $d
div=`expr $a / $b`
echo the quotient is $div
mul=`expr $a \* $b`
echo the product of two numbers is $mul
```



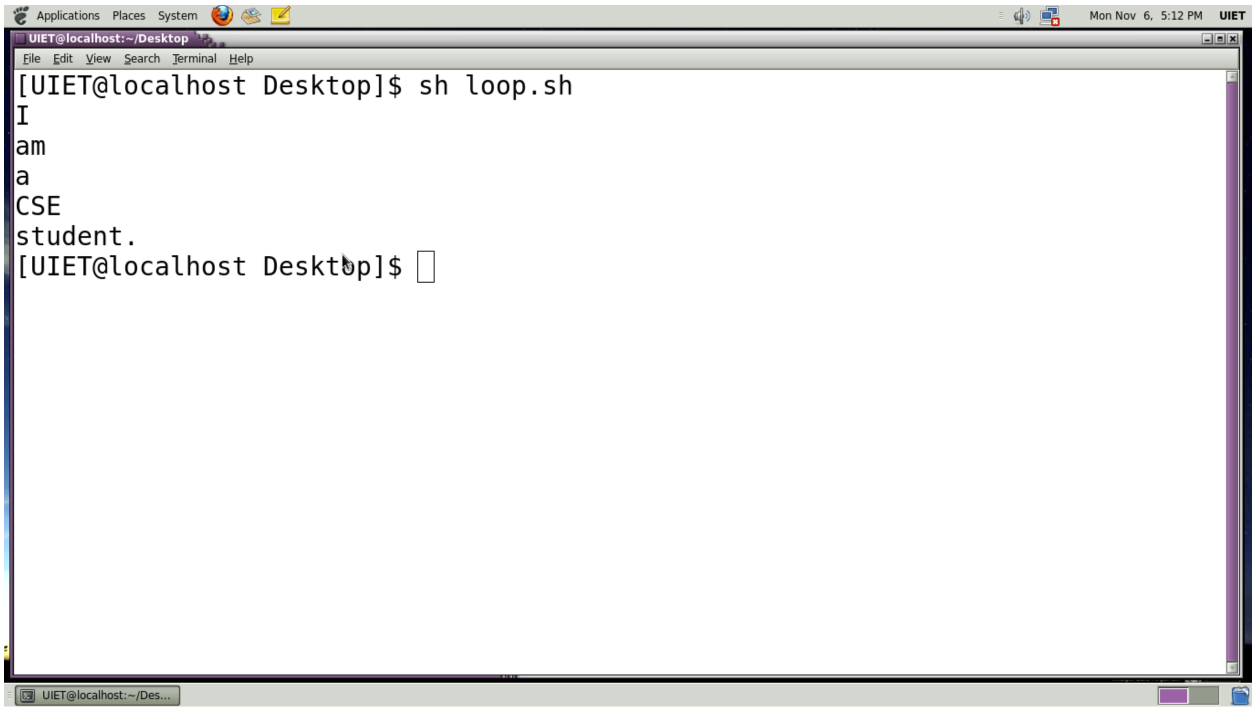
```
[UIET@localhost ~]$ sh calcc.sh
enter any two numbers
78 67
the sum of two numbers is 145
Ther difference between two numbers is 11
the quotient is 1
the product of two numbers is 5226
[UIET@localhost ~]$
```

PROGRAM 7(a)

Shell program to demonstrate the use of loops



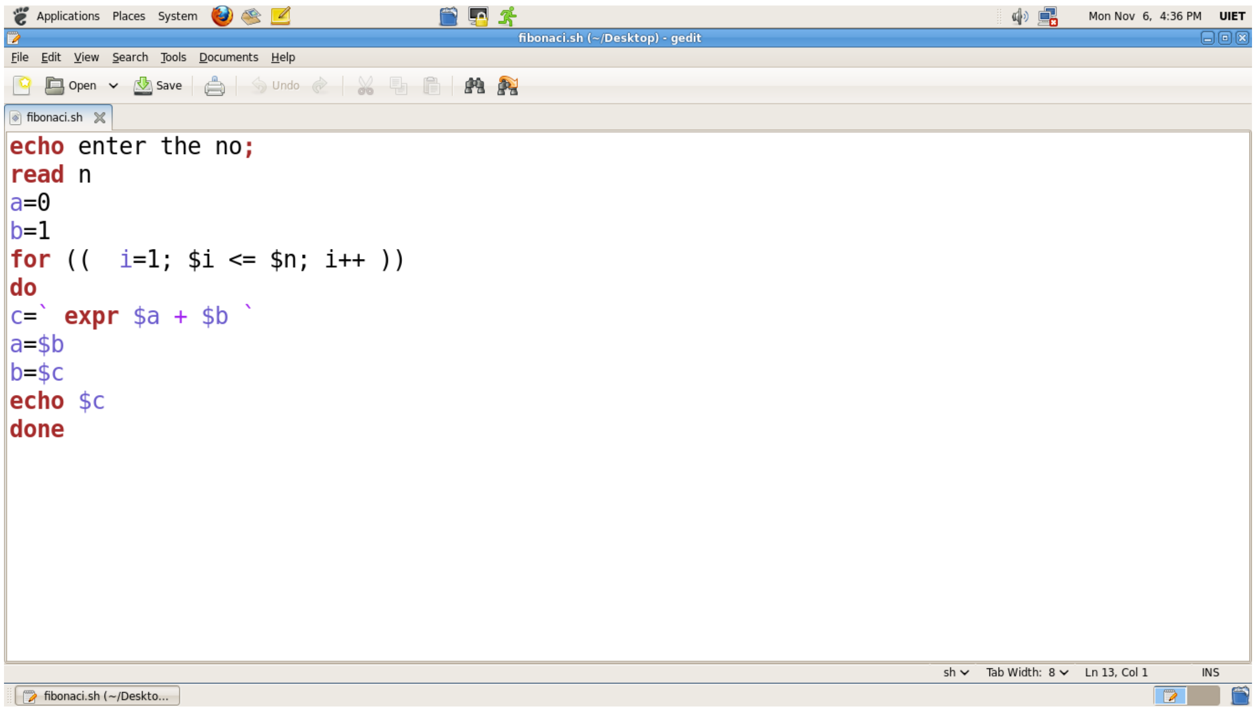
```
for word in I am a CSE student.
do
echo $word
done
```



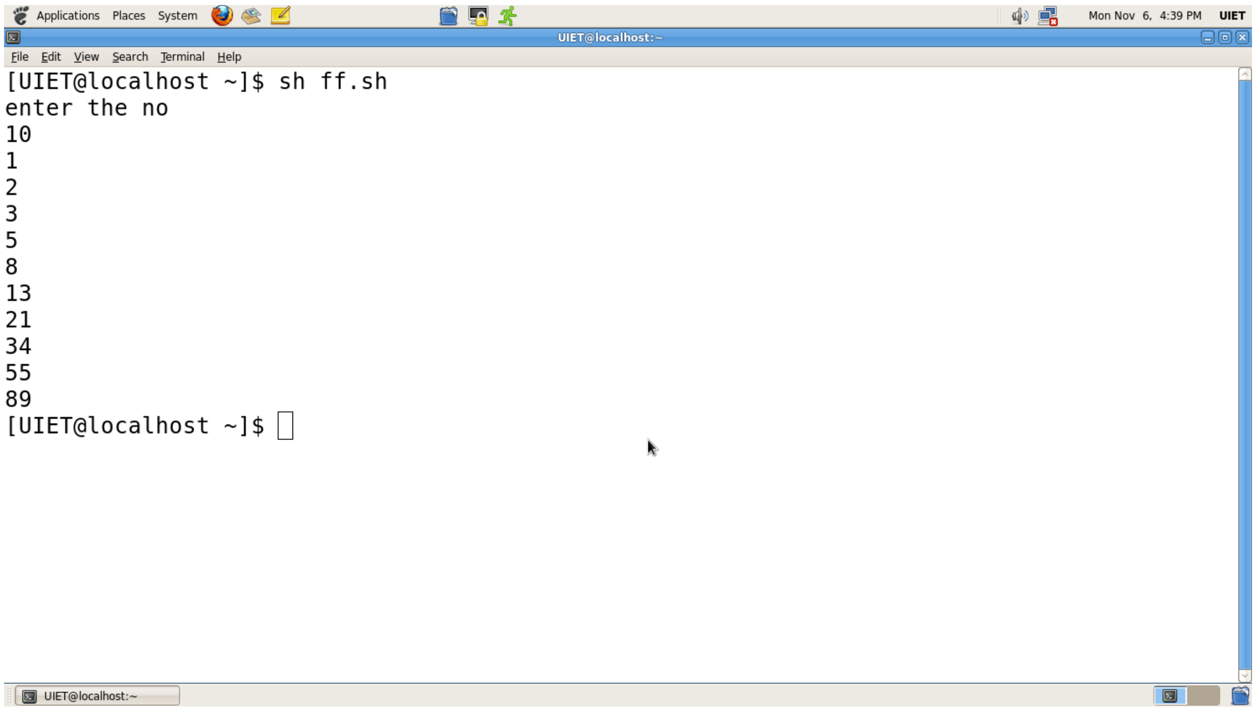
```
[UIET@localhost Desktop]$ sh loop.sh
I
am
a
CSE
student.
[UIET@localhost Desktop]$
```

PROGRAM 7(b)

Shell program to create a fibonacci series



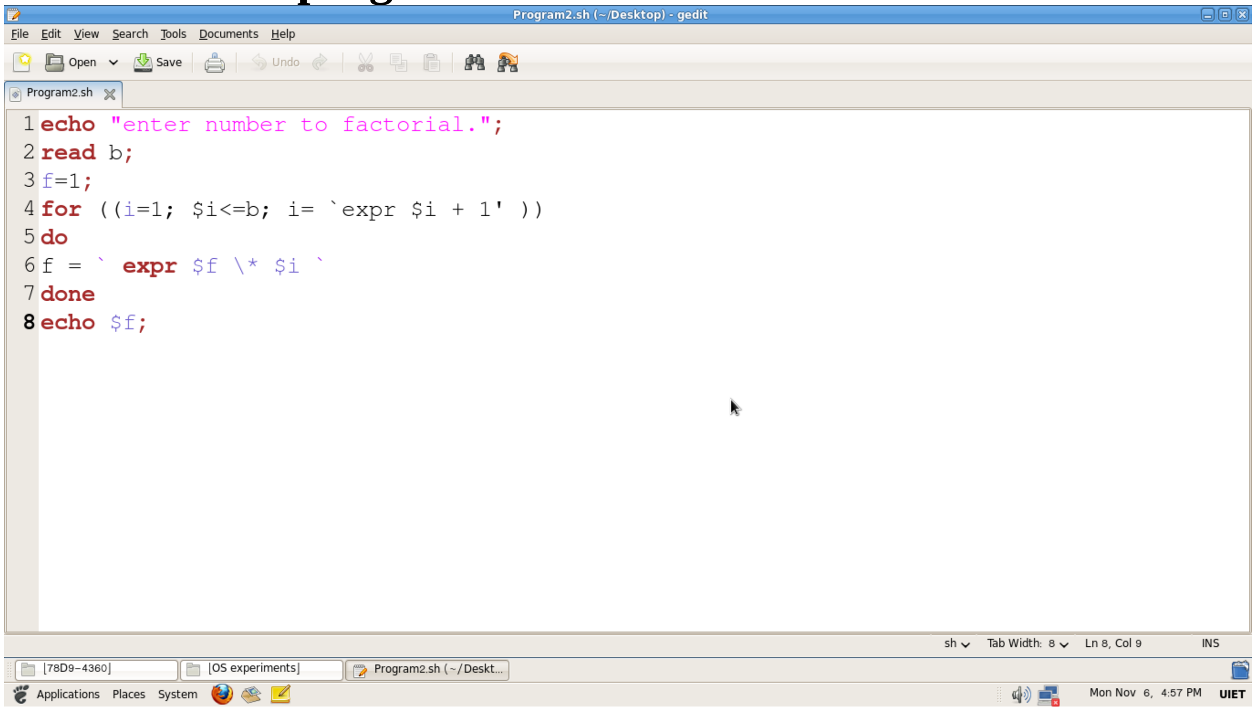
```
echo enter the no;
read n
a=0
b=1
for ((  i=1; $i <= $n; i++ ))
do
c=` expr $a + $b `
a=$b
b=$c
echo $c
done
```



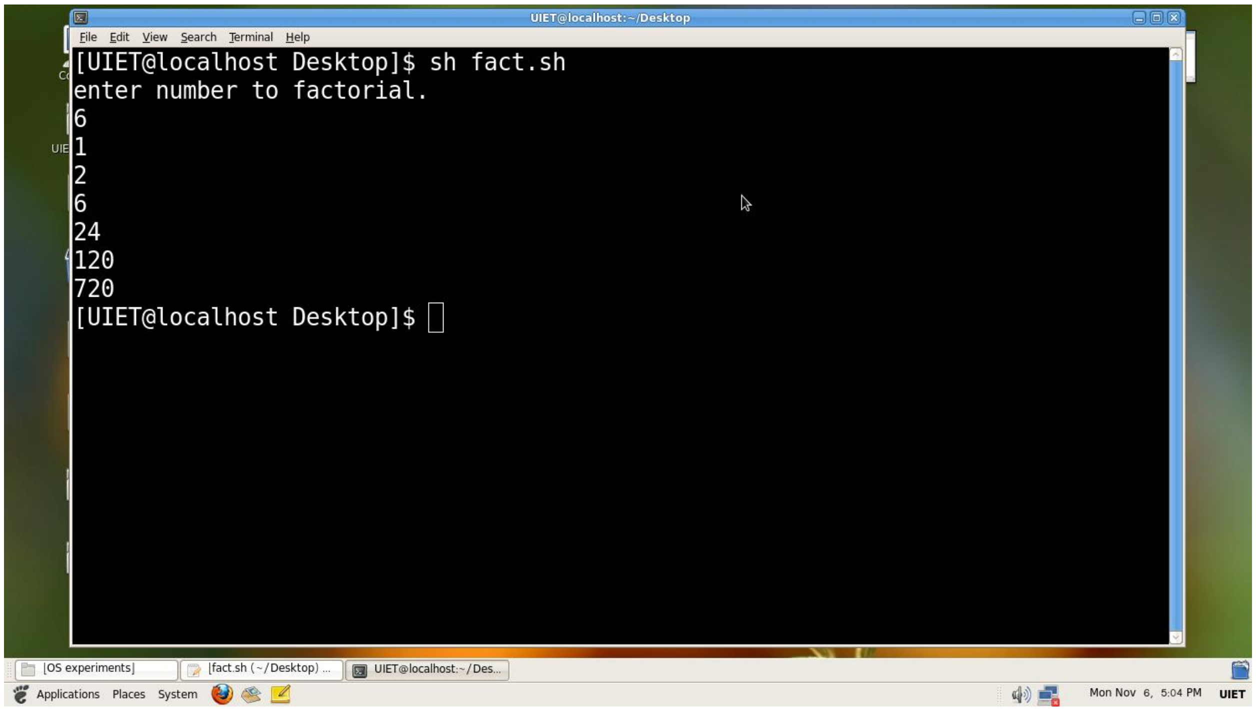
```
[UIET@localhost ~]$ sh ff.sh
enter the no
10
1
2
3
5
8
13
21
34
55
89
[UIET@localhost ~]$
```

PROGRAM 7(c)

Shell program to find factorial of a number



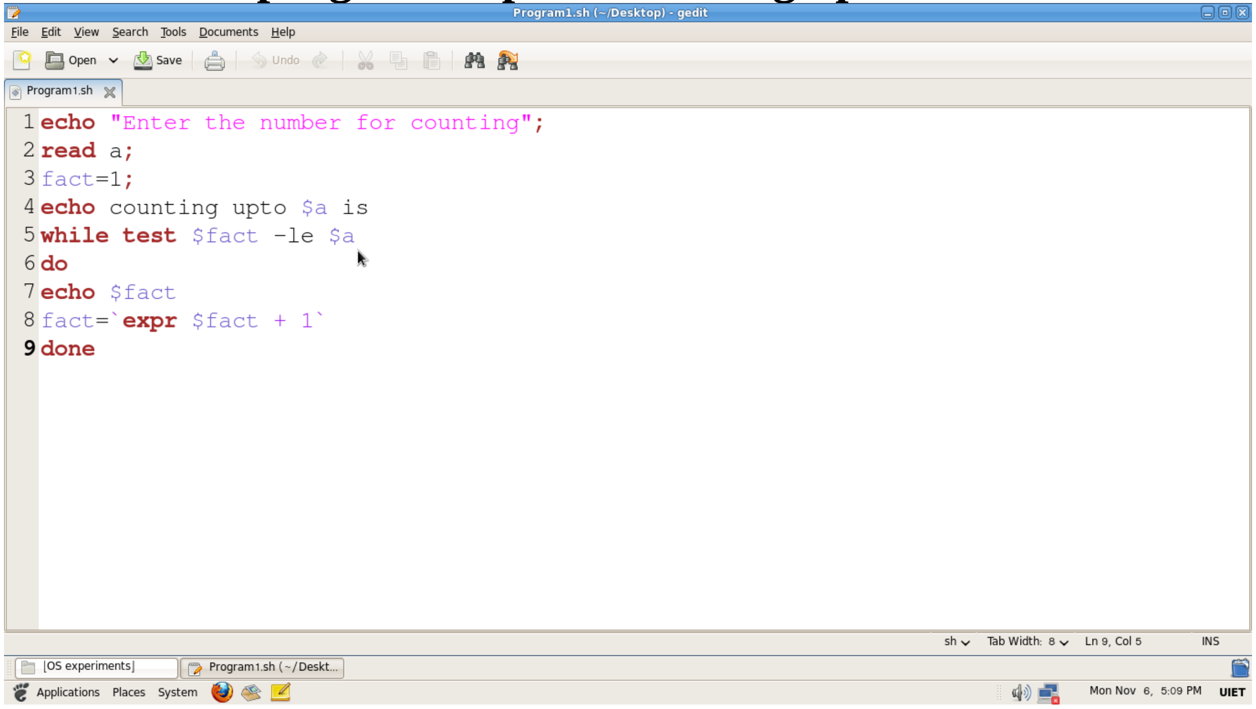
```
echo "enter number to factorial.";
read b;
f=1;
for ((i=1; $i<=b; i= `expr $i + 1' ))
do
f = ` expr $f \* $i `
done
echo $f;
```



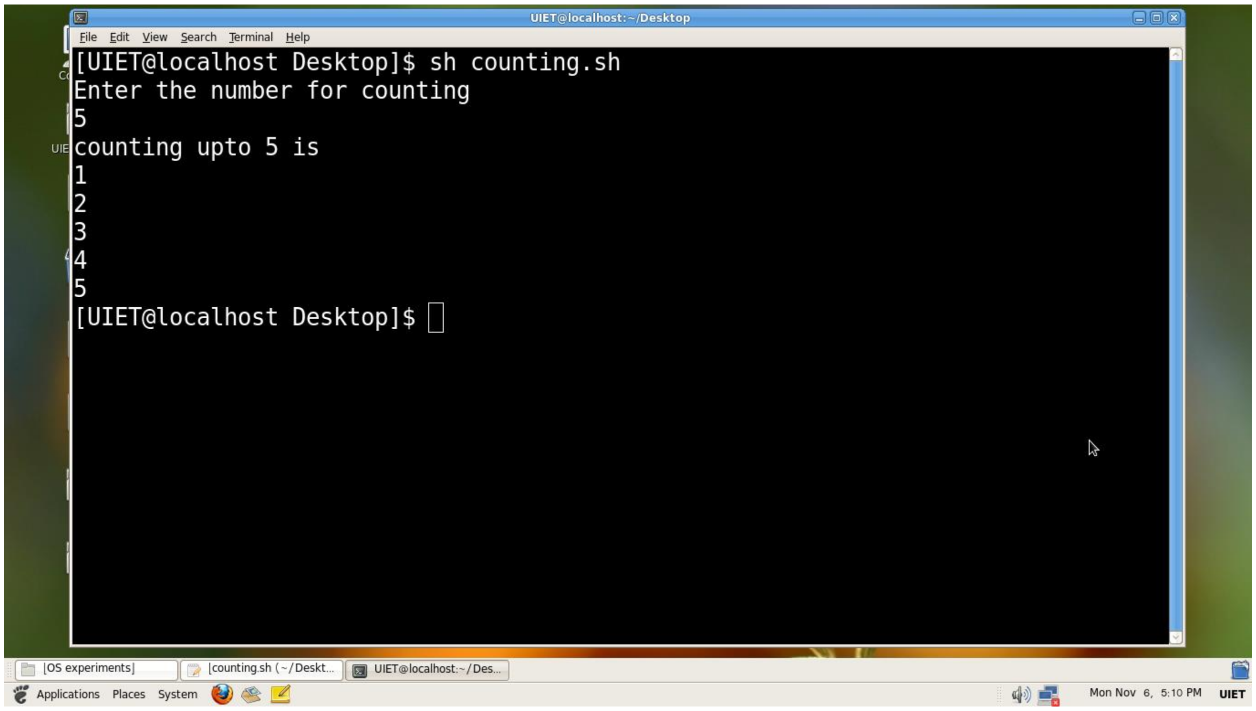
```
[UIET@localhost Desktop]$ sh fact.sh
enter number to factorial.
6
1
2
6
24
120
720
[UIET@localhost Desktop]$
```

PROGRAM 7(b)

Shell program to print counting upto a number



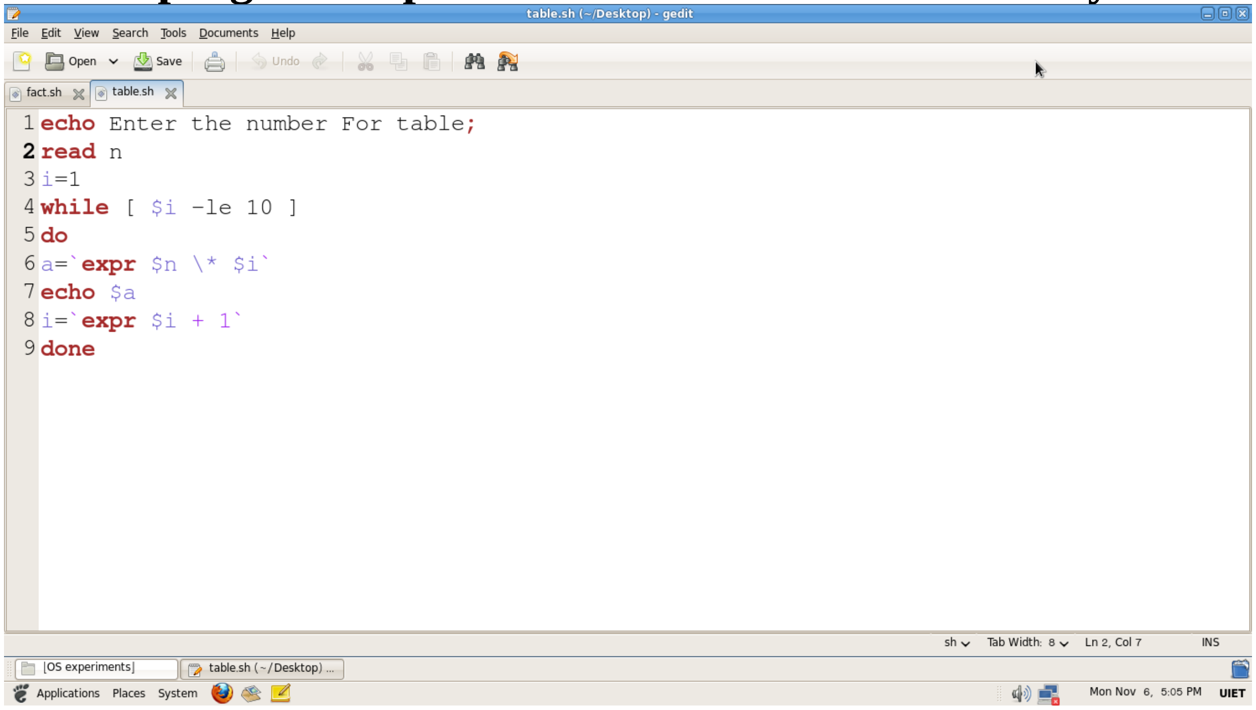
```
echo "Enter the number for counting";
read a;
fact=1;
echo counting upto $a is
while test $fact -le $a
do
echo $fact
fact=`expr $fact + 1`
done
```



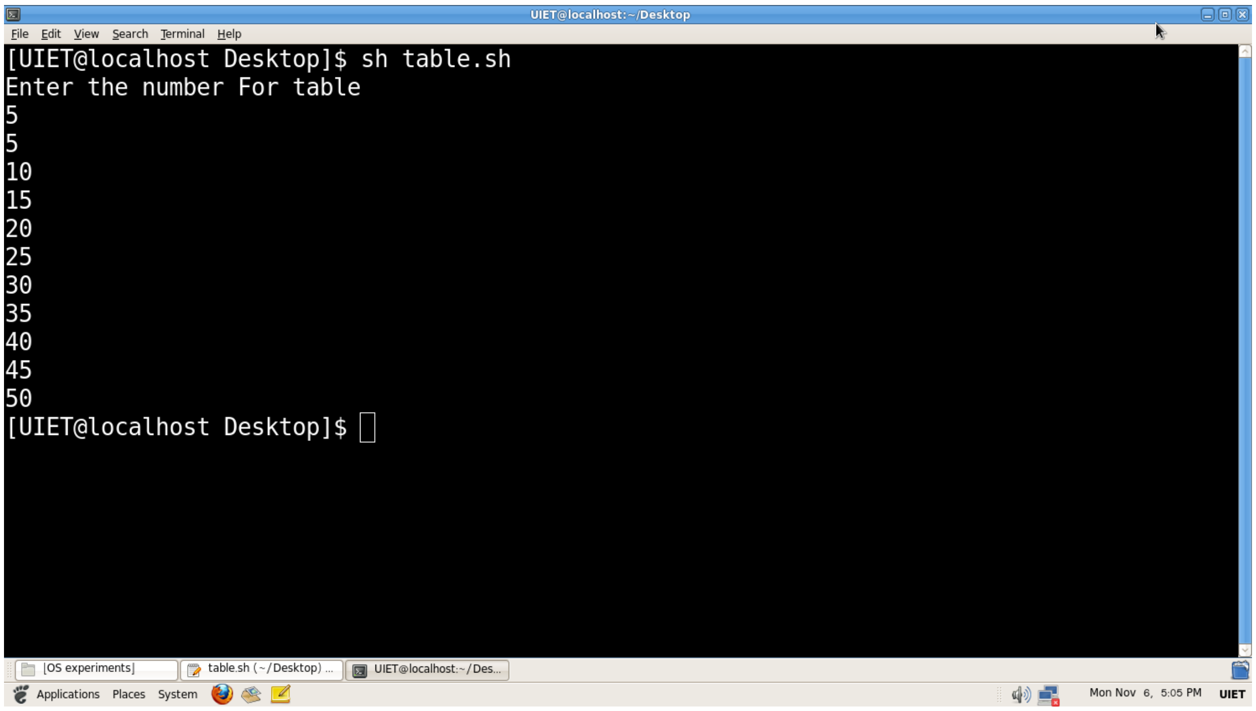
```
[UIET@localhost Desktop]$ sh counting.sh
Enter the number for counting
5
counting upto 5 is
1
2
3
4
5
[UIET@localhost Desktop]$
```

PROGRAM 7(e)

Shell program to print table of a number entered by user



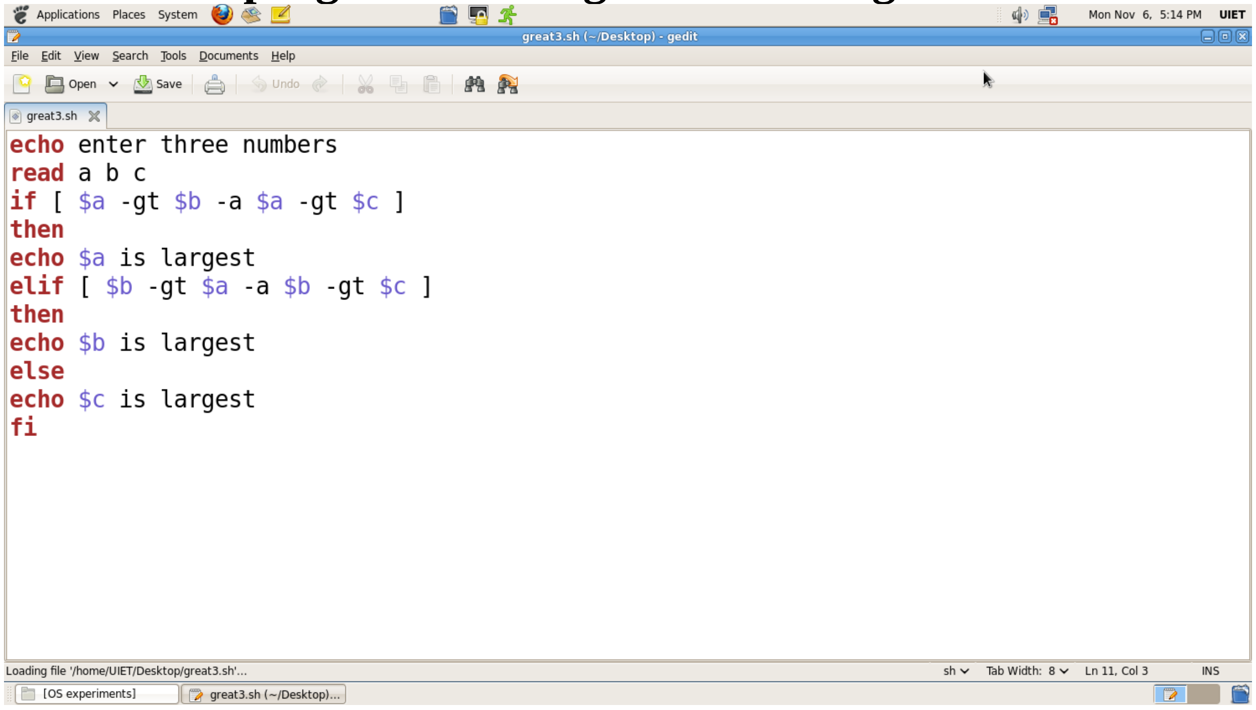
```
echo Enter the number For table;
read n
i=1
while [ $i -le 10 ]
do
a=`expr $n \* $i`
echo $a
i=`expr $i + 1`
done
```



```
[UIET@localhost Desktop]$ sh table.sh
Enter the number For table
5
5
10
15
20
25
30
35
40
45
50
[UIET@localhost Desktop]$ 
```

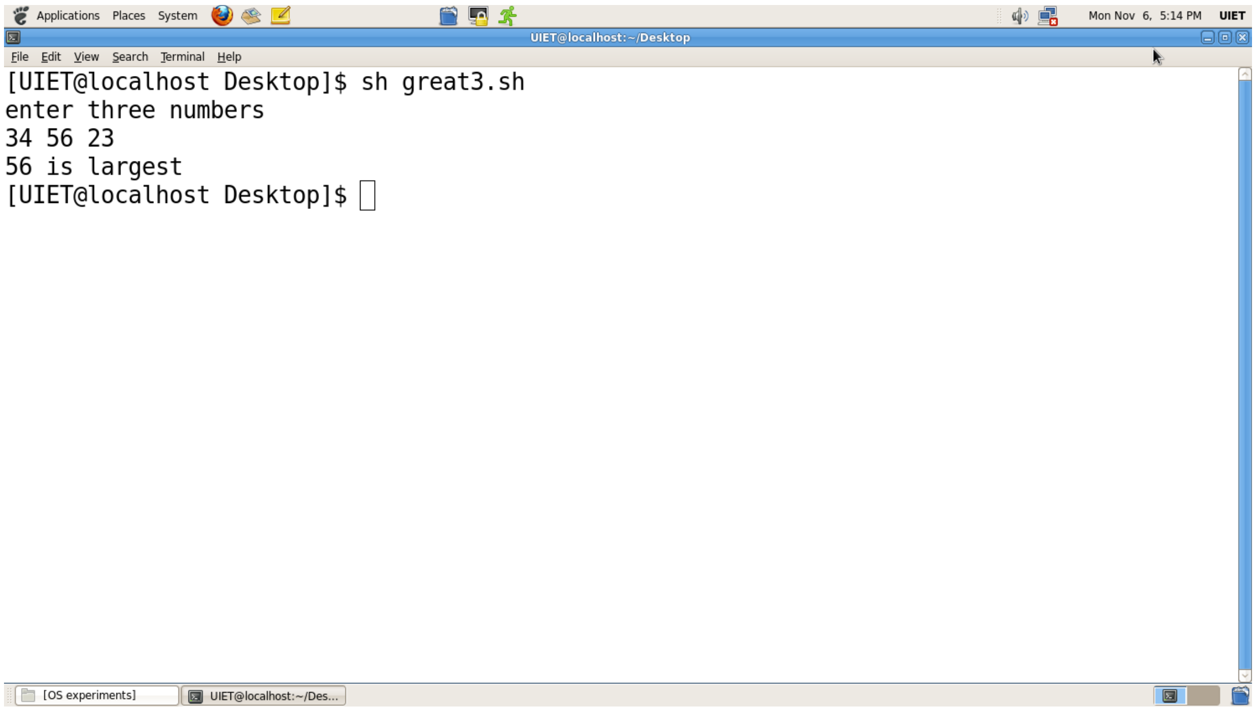
PROGRAM 8

Shell program to find greatest among 3 numbers



```
echo enter three numbers
read a b c
if [ $a -gt $b -a $a -gt $c ]
then
echo $a is largest
elif [ $b -gt $a -a $b -gt $c ]
then
echo $b is largest
else
echo $c is largest
fi
```

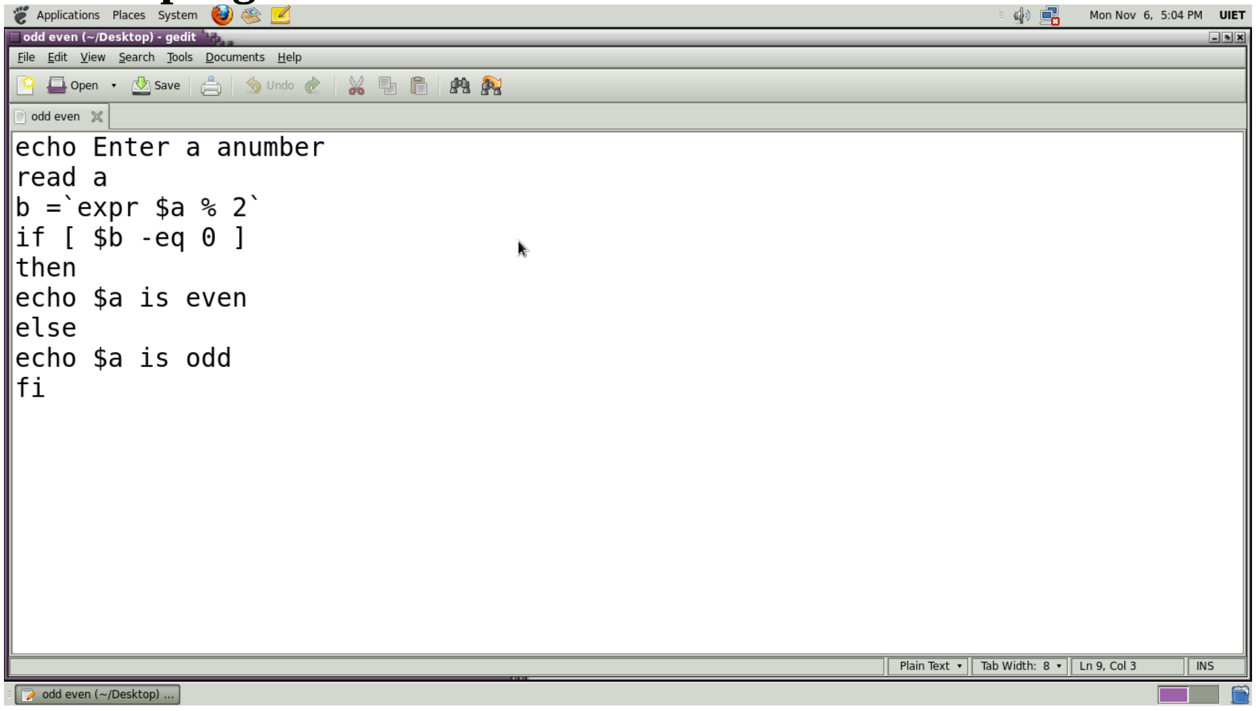
Applications Places System
Mon Nov 6, 5:14 PM
UIET
great3.sh (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
great3.sh
Loading file '/home/UIET/Desktop/great3.sh'...
sh
Tab Width: 8
Ln 11, Col 3
INS
[OS experiments]
great3.sh (~/Desktop)...



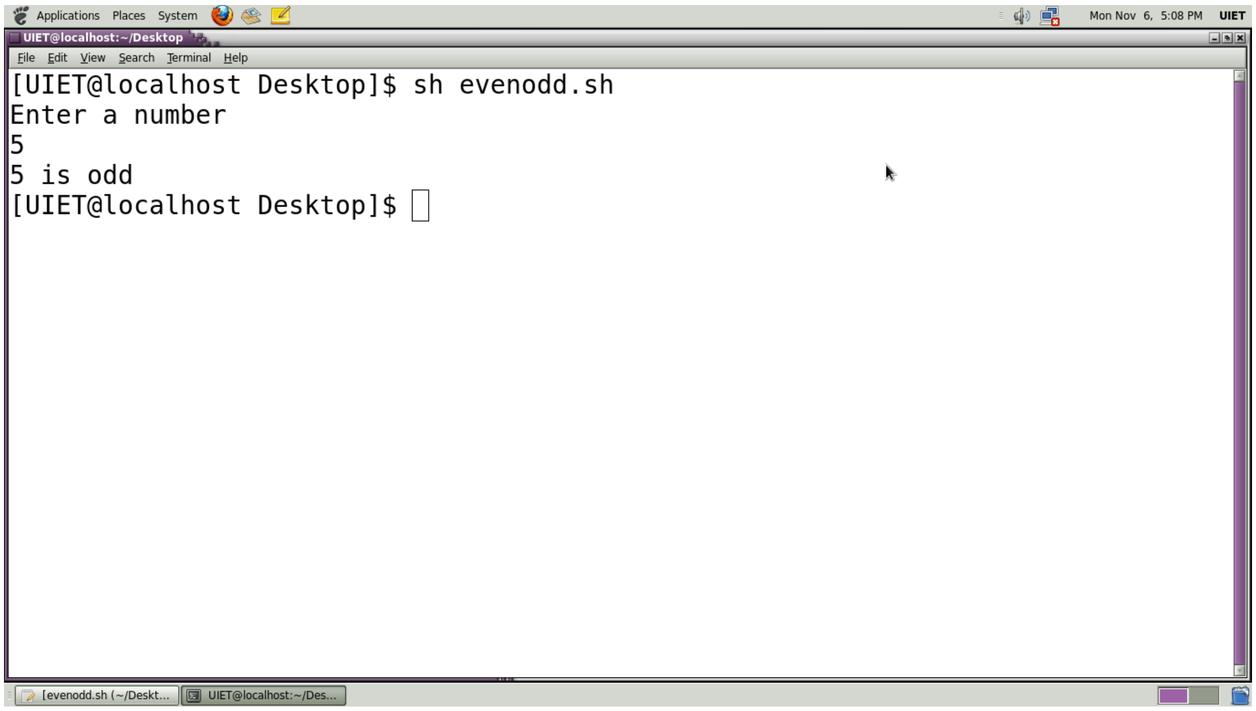
```
[UIET@localhost Desktop]$ sh great3.sh
enter three numbers
34 56 23
56 is largest
[UIET@localhost Desktop]$ 
```

PROGRAM 8(b)

Shell program to check whether a number is odd or even


Applications Places System
Mon Nov 6, 5:04 PM UIET
odd even (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
odd even
Plain Text Tab Width: 8 Ln 9, Col 3 INS
odd even (~/Desktop) ...

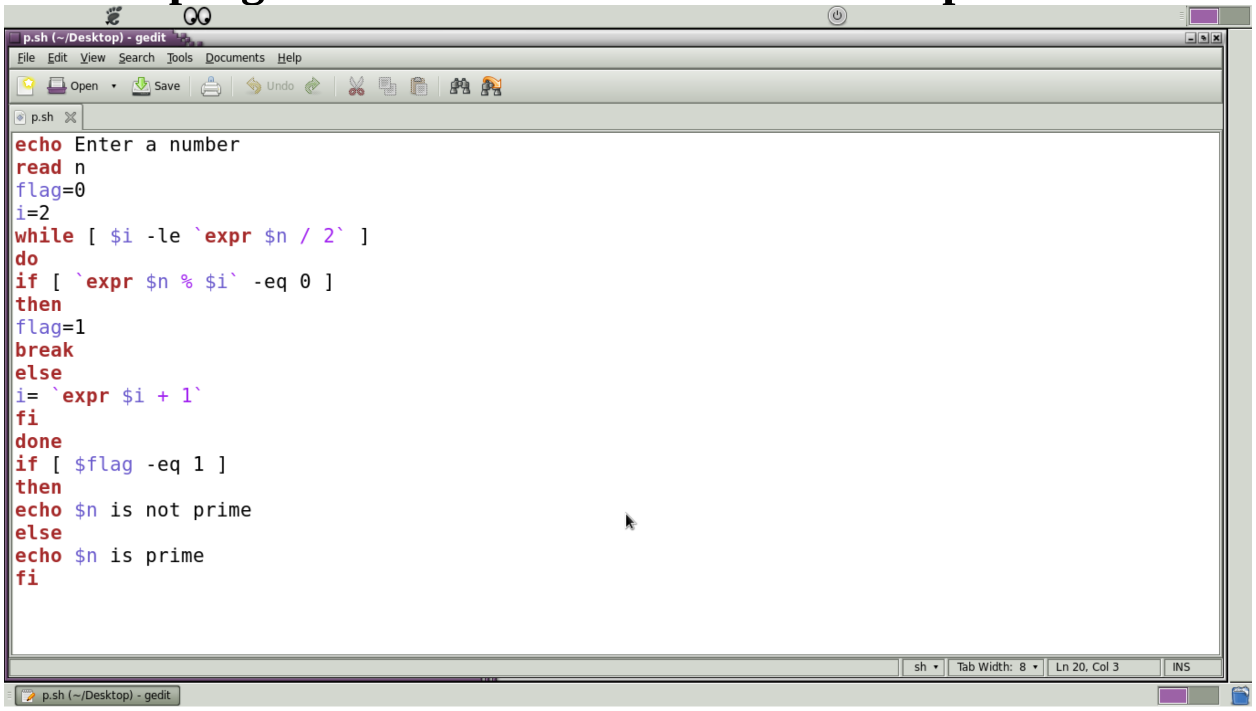
```
echo Enter a anumber
read a
b =`expr $a % 2`
if [ $b -eq 0 ]
then
echo $a is even
else
echo $a is odd
fi
```



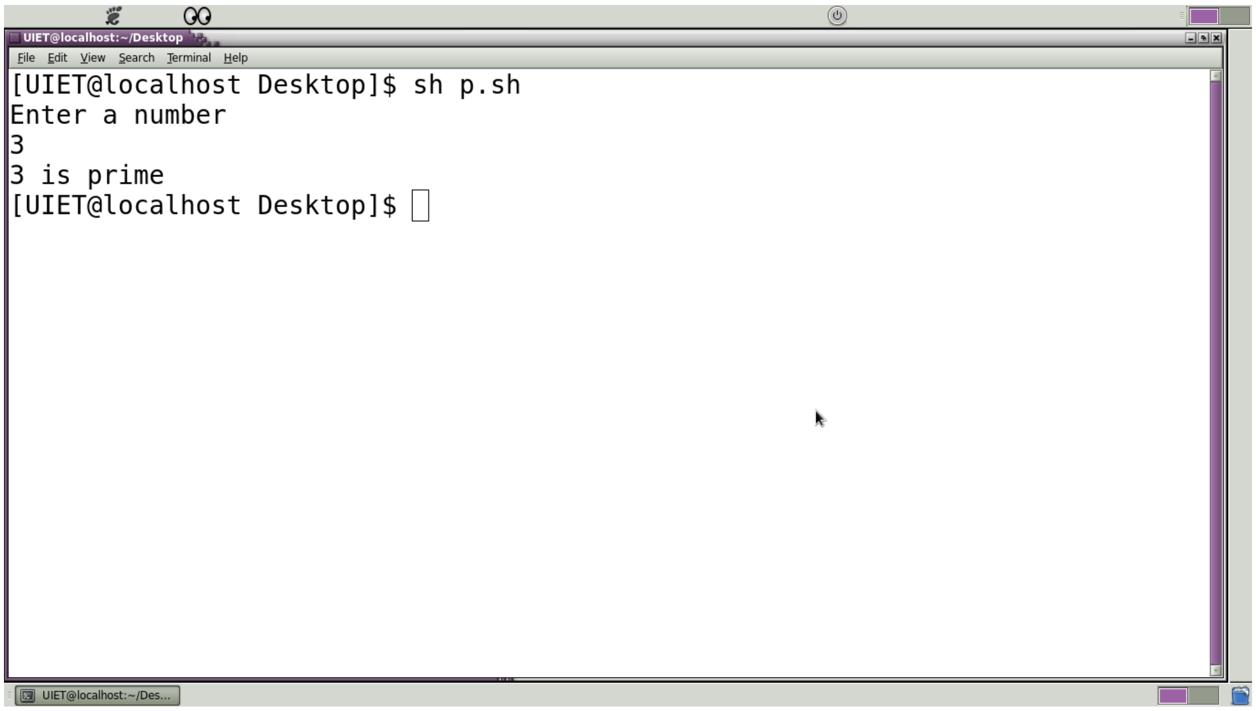
```
[UIET@localhost Desktop]$ sh evenodd.sh
Enter a number
5
5 is odd
[UIET@localhost Desktop]$ 
```

PROGRAM 8(c)

Shell program to check whether number is prime or not


p.sh (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open Save Undo
p.sh
sh Tab Width: 8 Ln 20, Col 3 INS
p.sh (~/Desktop) - gedit

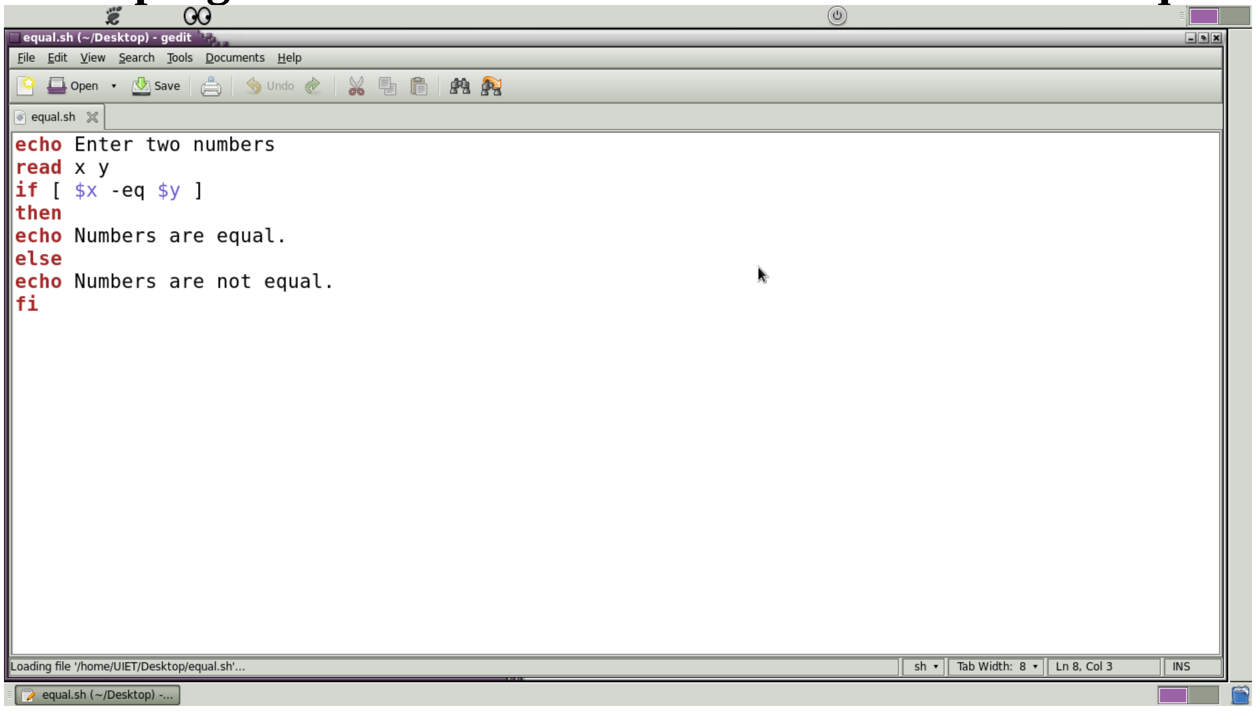
```
echo Enter a number
read n
flag=0
i=2
while [ $i -le `expr $n / 2` ]
do
if [ `expr $n % $i` -eq 0 ]
then
flag=1
break
else
i= `expr $i + 1`
fi
done
if [ $flag -eq 1 ]
then
echo $n is not prime
else
echo $n is prime
fi
```



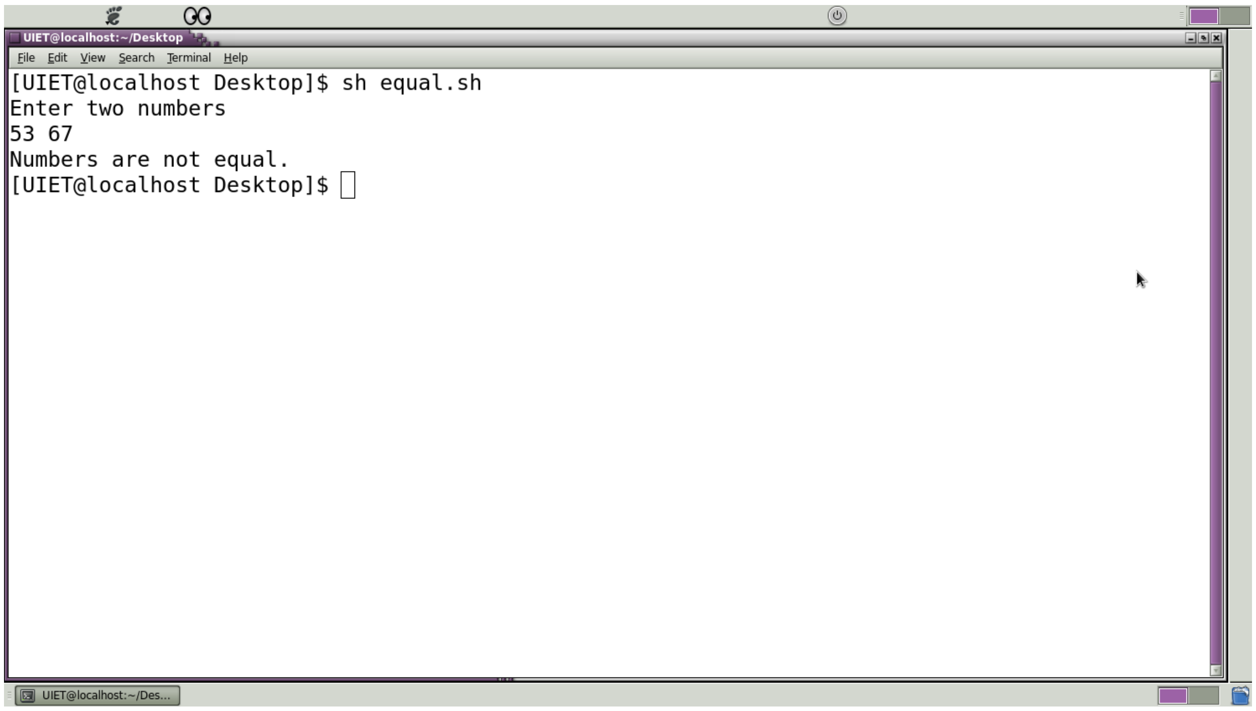
```
[UIET@localhost Desktop]$ sh p.sh
Enter a number
3
3 is prime
[UIET@localhost Desktop]$
```

PROGRAM 8(d)

Shell program to check whether the two numbers are equal



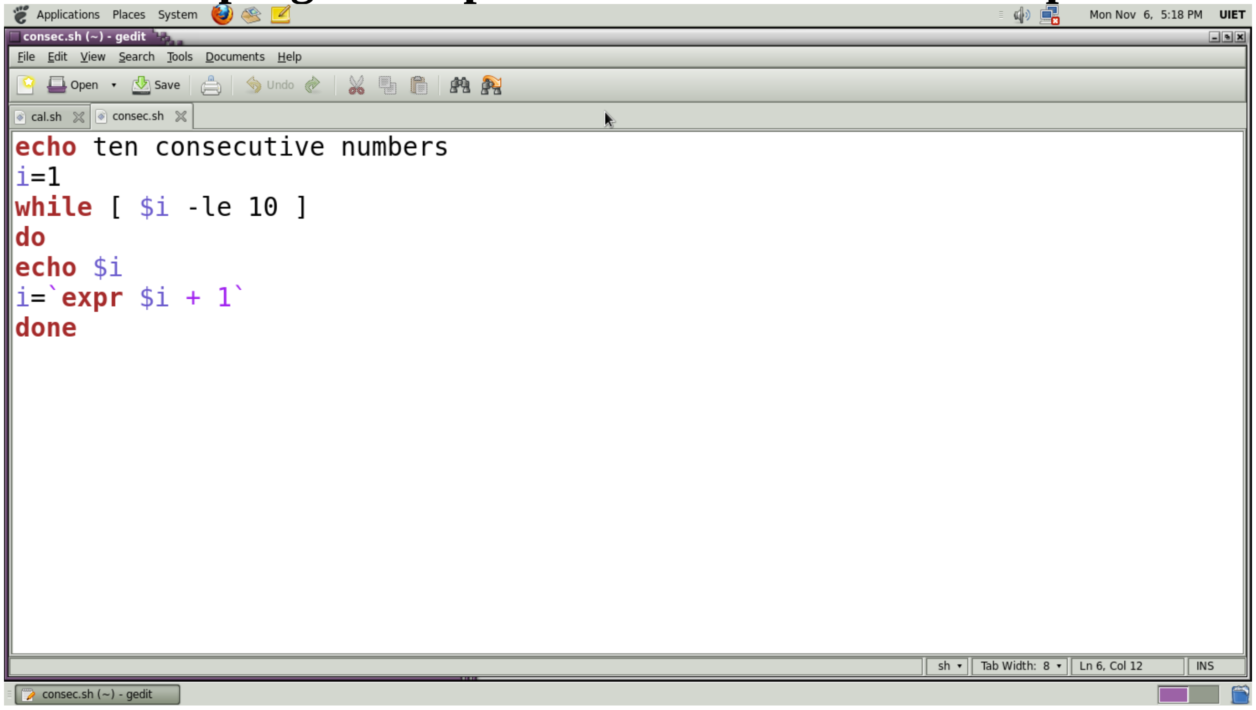
```
echo Enter two numbers
read x y
if [ $x -eq $y ]
then
echo Numbers are equal.
else
echo Numbers are not equal.
fi
```



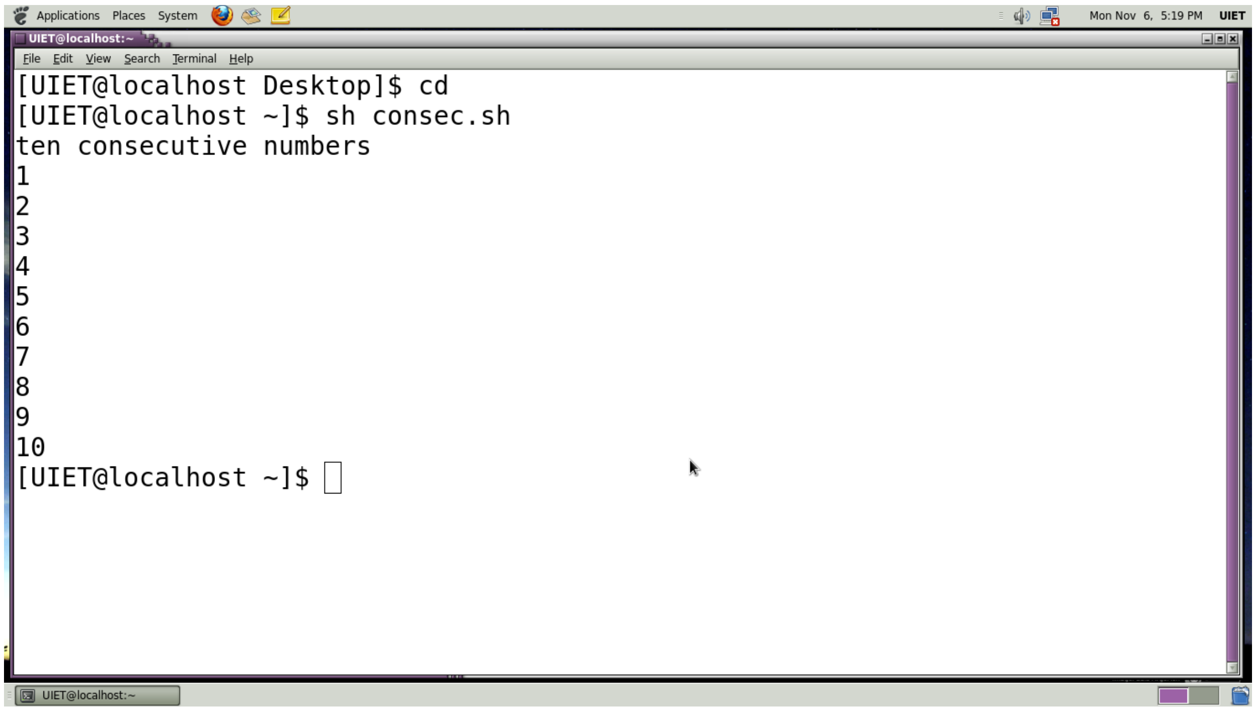
```
[UIET@localhost Desktop]$ sh equal.sh
Enter two numbers
53 67
Numbers are not equal.
[UIET@localhost Desktop]$ 
```

PROGRAM 9

Shell program to print consecutive numbers upto n



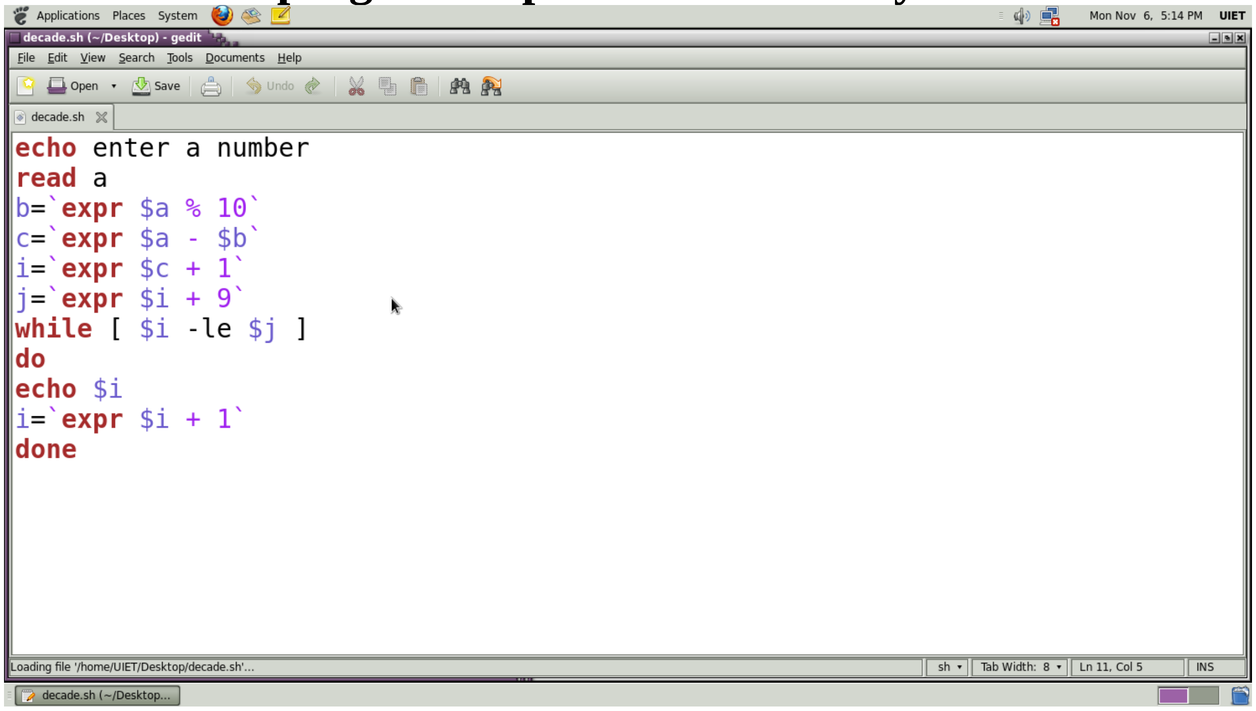
```
echo ten consecutive numbers
i=1
while [ $i -le 10 ]
do
echo $i
i=`expr $i + 1`
done
```



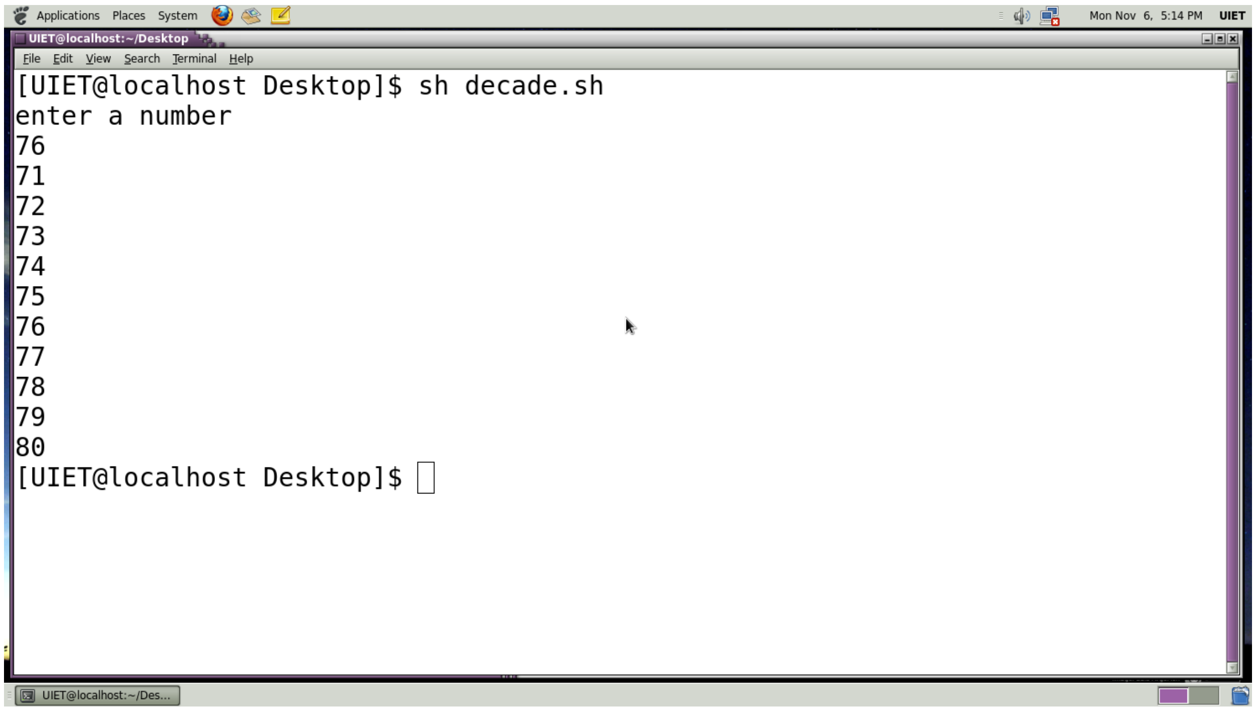
```
[UIET@localhost Desktop]$ cd
[UIET@localhost ~]$ sh consec.sh
ten consecutive numbers
1
2
3
4
5
6
7
8
9
10
[UIET@localhost ~]$
```

PROGRAM 10

Shell program to print decade of any number



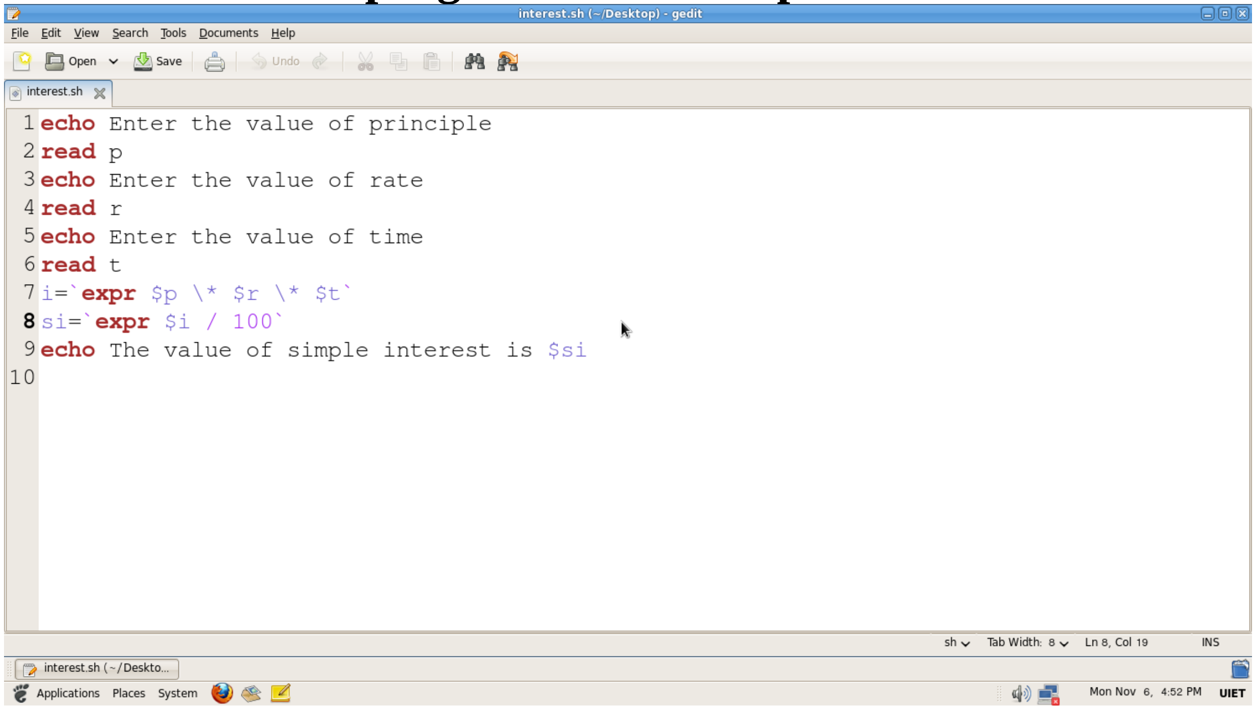
```
echo enter a number
read a
b=`expr $a % 10`
c=`expr $a - $b`
i=`expr $c + 1`
j=`expr $i + 9`
while [ $i -le $j ]
do
echo $i
i=`expr $i + 1`
done
```



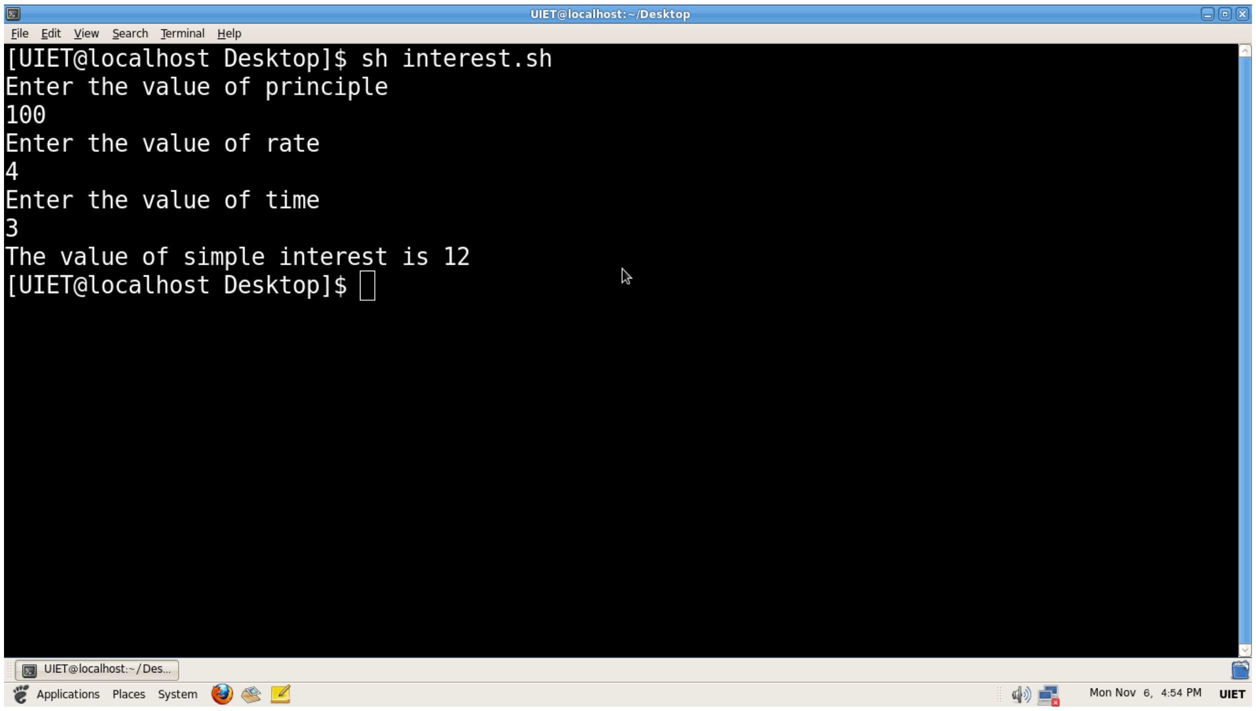
```
[UIET@localhost Desktop]$ sh decade.sh
enter a number
76
71
72
73
74
75
76
77
78
79
80
[UIET@localhost Desktop]$ 
```

PROGRAM 11

Shell program to find simple interest



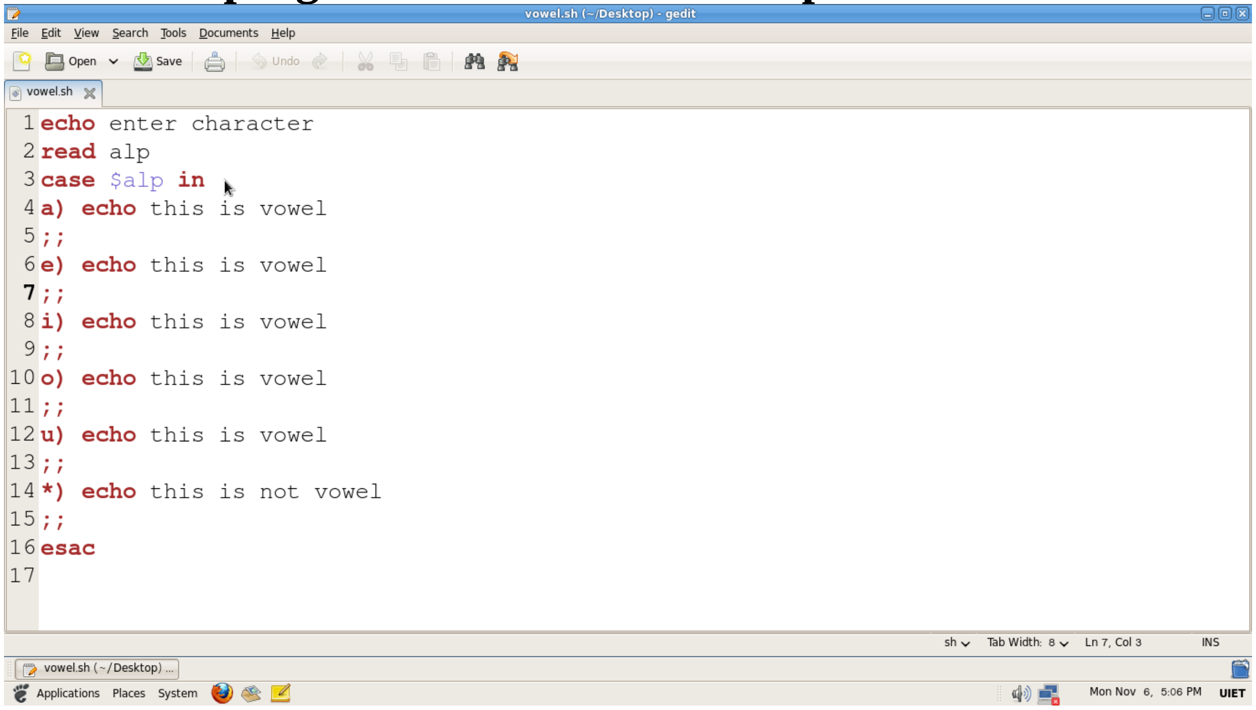
```
echo Enter the value of principle
read p
echo Enter the value of rate
read r
echo Enter the value of time
read t
i=`expr $p \* $r \* $t`
si=`expr $i / 100`
echo The value of simple interest is $si
```



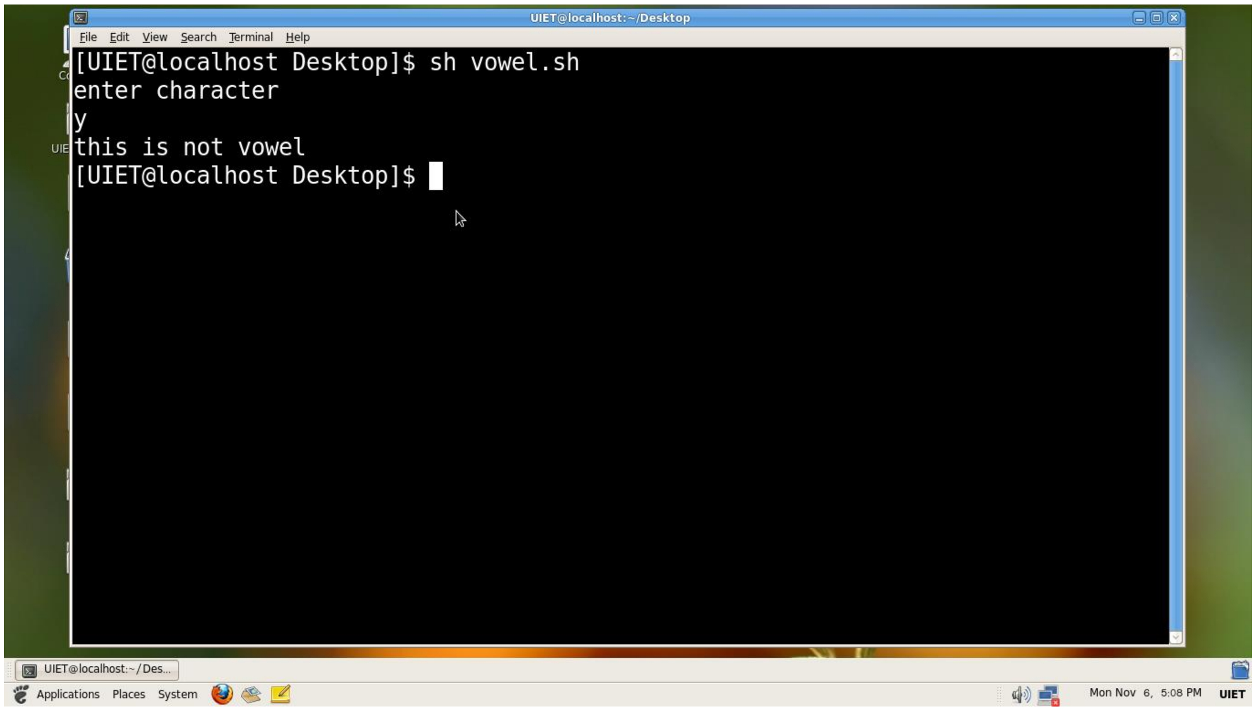
```
[UIET@localhost Desktop]$ sh interest.sh
Enter the value of principle
100
Enter the value of rate
4
Enter the value of time
3
The value of simple interest is 12
[UIET@localhost Desktop]$
```

PROGRAM 12

Shell program to check entered alphabet for vowel



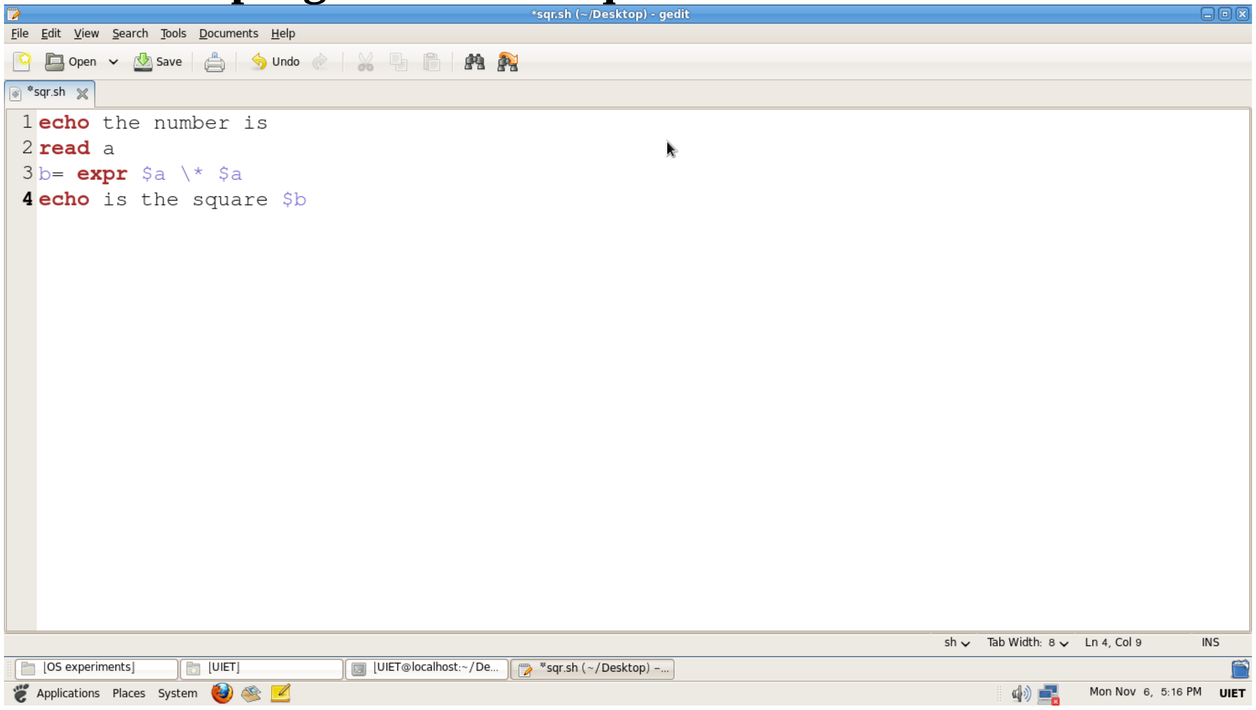
```
echo enter character
read alp
case $alp in
a) echo this is vowel
;;
e) echo this is vowel
;;
i) echo this is vowel
;;
o) echo this is vowel
;;
u) echo this is vowel
;;
*) echo this is not vowel
;;
esac
```



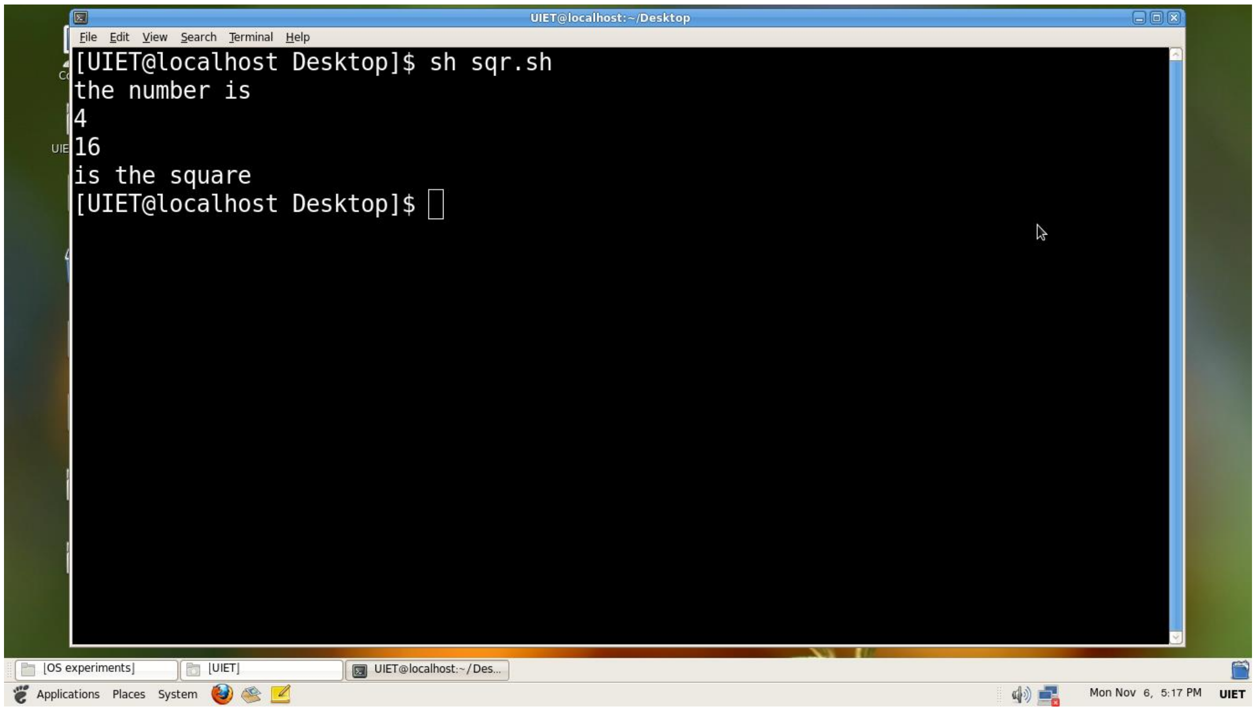
```
[UIET@localhost Desktop]$ sh vowel.sh
enter character
y
this is not vowel
[UIET@localhost Desktop]$
```

PROGRAM 13

Shell program to find square of a number entered



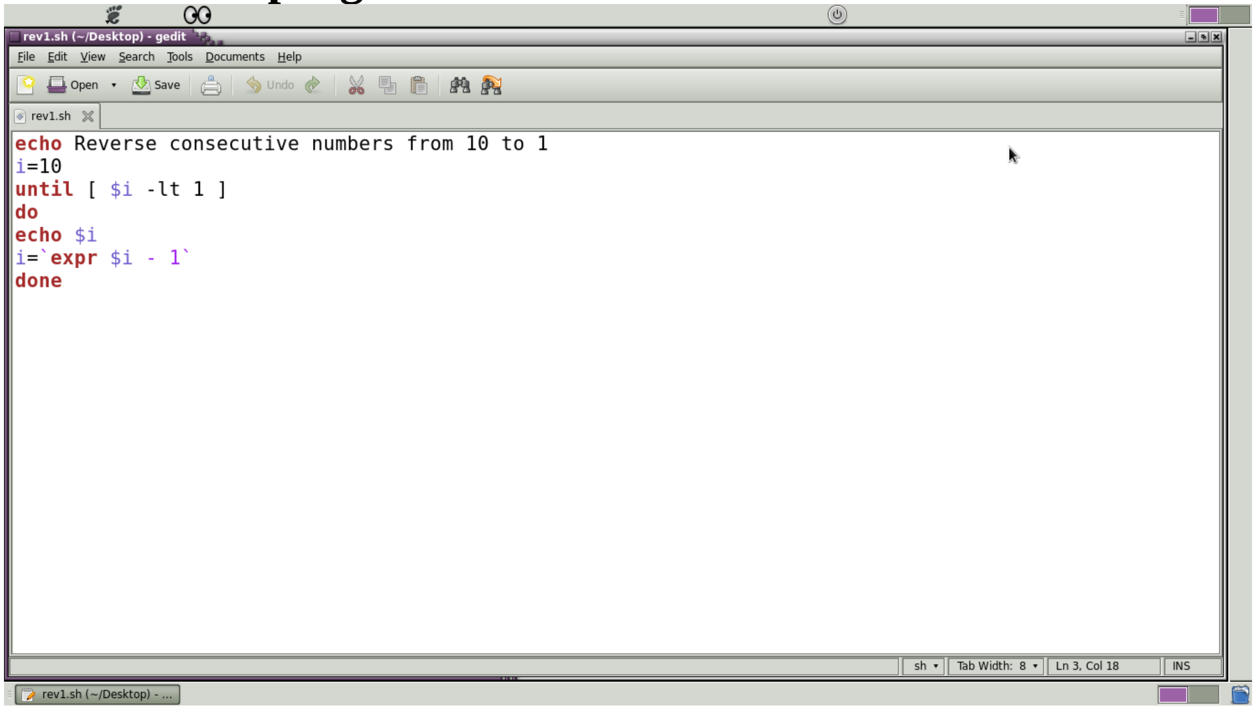
```
echo the number is
read a
b= expr $a \* $a
echo is the square $b
```



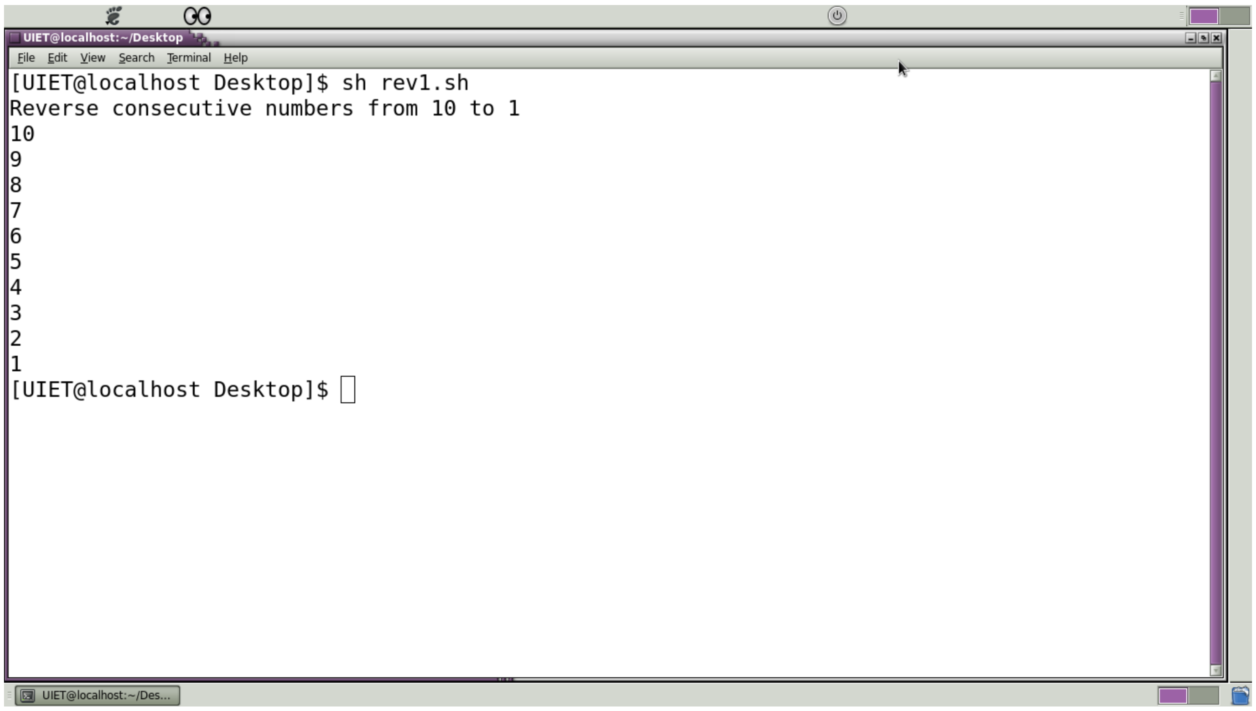
```
[UIET@localhost Desktop]$ sh sqr.sh
the number is
4
16
is the square
[UIET@localhost Desktop]$ 
```

PROGRAM 14(a)

Shell program to reverse consecutive numbers



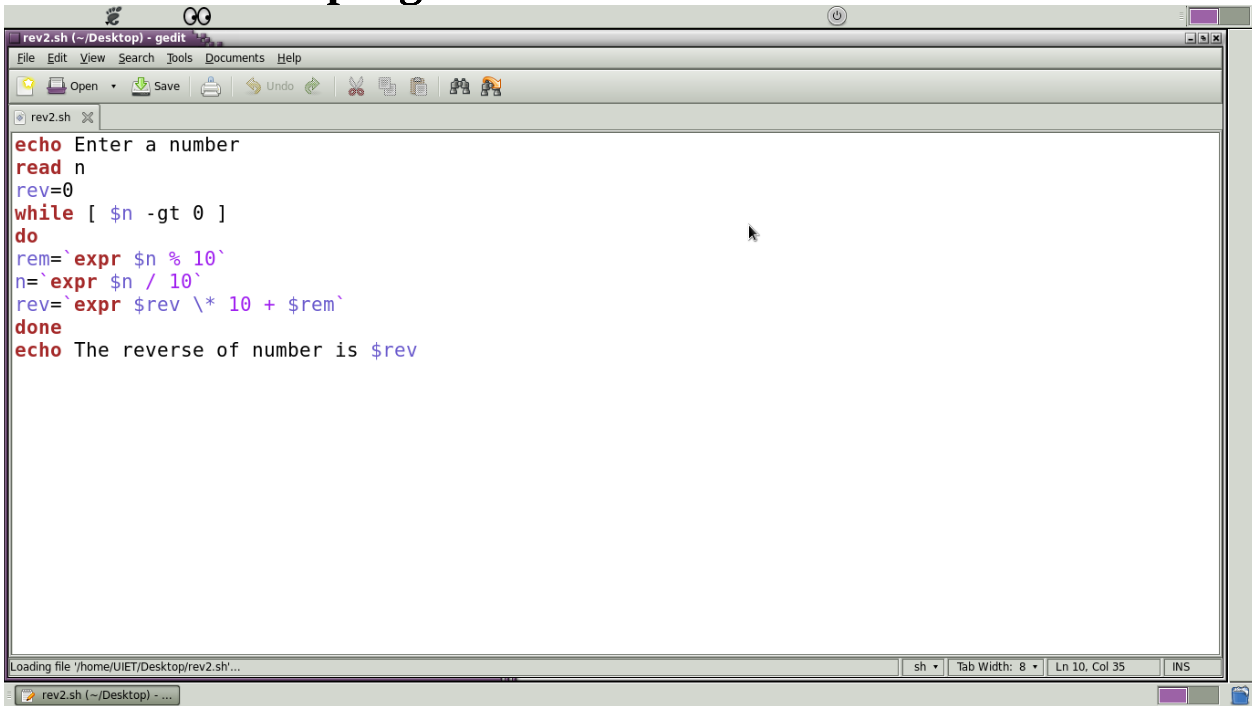
```
echo Reverse consecutive numbers from 10 to 1
i=10
until [ $i -lt 1 ]
do
echo $i
i=`expr $i - 1`
done
```



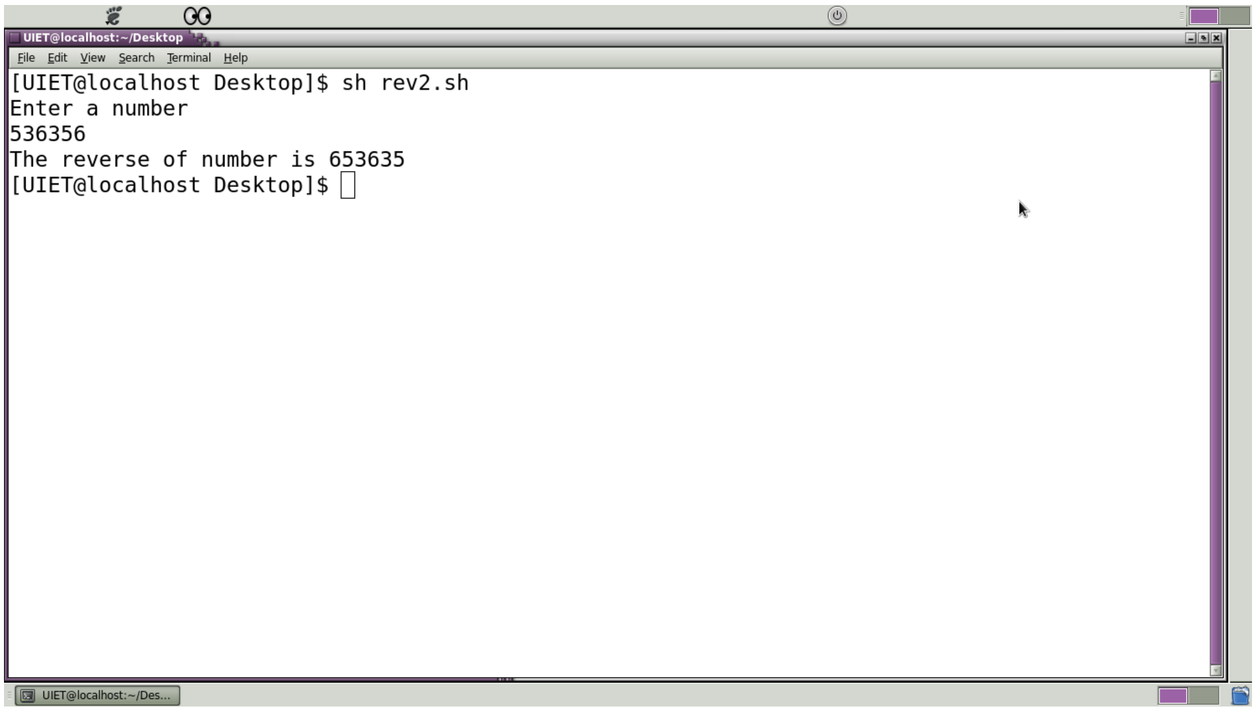
```
[UIET@localhost Desktop]$ sh rev1.sh
Reverse consecutive numbers from 10 to 1
10
9
8
7
6
5
4
3
2
1
[UIET@localhost Desktop]$ 
```

PROGRAM 14(b)

Shell program to reverse a number itself


rev2.sh (~/Desktop) - gedit
File Edit View Search Tools Documents Help
Open
Save
Undo
rev2.sh
Loading file '/home/UIET/Desktop/rev2.sh'...
sh
Tab Width: 8
Ln 10, Col 35
INS
rev2.sh (~/Desktop) - ...

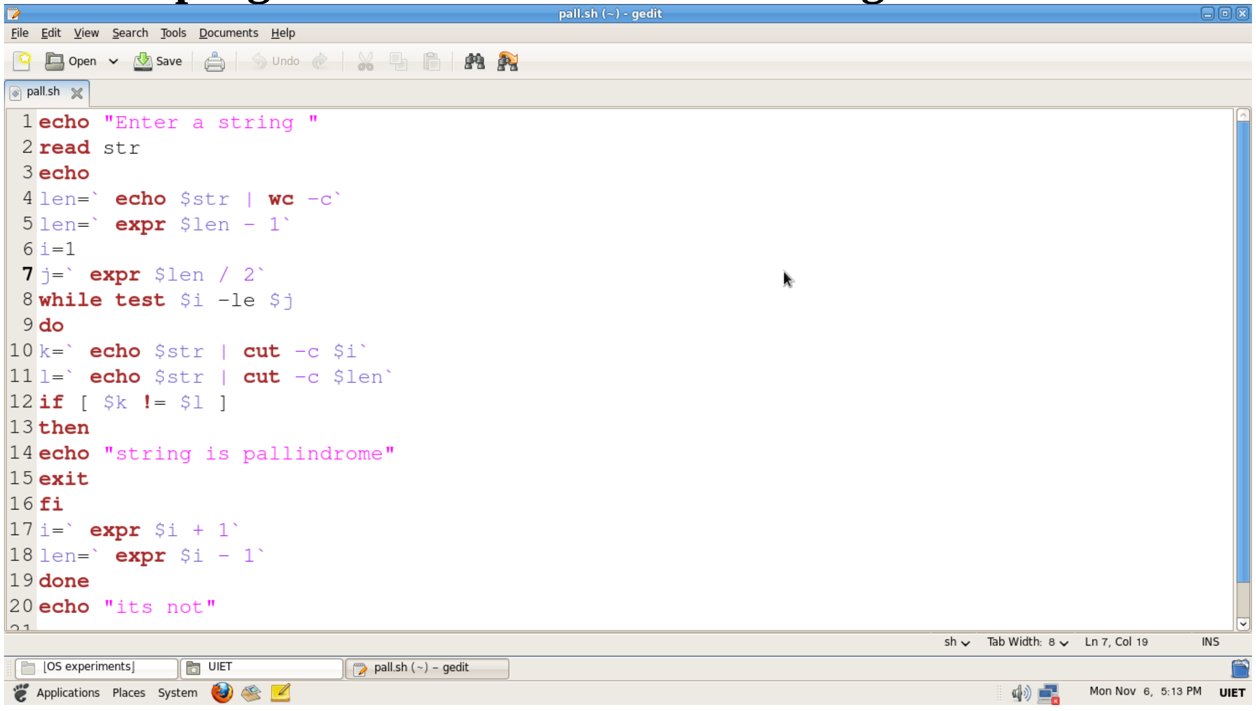
```
echo Enter a number
read n
rev=0
while [ $n -gt 0 ]
do
rem=`expr $n % 10`
n=`expr $n / 10`
rev=`expr $rev \* 10 + $rem`
done
echo The reverse of number is $rev
```



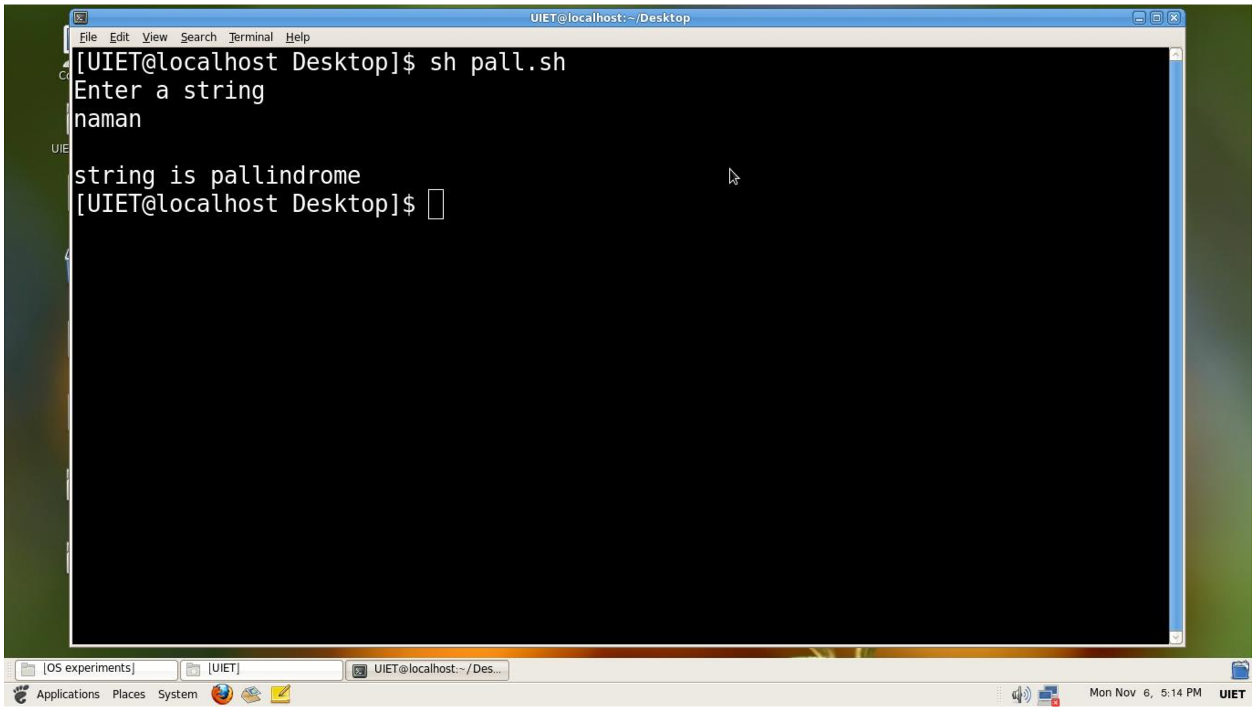
```
[UIET@localhost Desktop]$ sh rev2.sh
Enter a number
536356
The reverse of number is 653635
[UIET@localhost Desktop]$
```

PROGRAM 15

Shell program to check whether a string is Pallindrome



```
echo "Enter a string "
read str
echo
len=` echo $str | wc -c`
len=` expr $len - 1`
i=1
j=` expr $len / 2`
while test $i -le $j
do
k=` echo $str | cut -c $i`
l=` echo $str | cut -c $len`
if [ $k != $l ]
then
echo "string is pallindrome"
exit
fi
i=` expr $i + 1`
len=` expr $i - 1`
done
echo "its not"
```



```
[UIET@localhost Desktop]$ sh pall.sh
Enter a string
naman

string is pallindrome
[UIET@localhost Desktop]$ 
```