Varuvan Vadivelan
Institute of Technology

Dharmapuri – 636 703

LAB MANUAL

Regulation : 2013

Branch : B.E. – EEE

Year & Semester : I Year / II Semester

GE 6263 – COMPUTER PROGRAMMING LABORATORY

ANNA UNIVERSITY : CHENNAI

REGULATION : 2013

SYLLABUS

GE6263 - COMPUTER PROGRAMMING LABORATORY

1. UNIX COMMANDS **15**

   Study of UNIX OS - Basic Shell Commands - Unix Editor

2. SHELL PROGRAMMING **15**

   Simple Shell program - Conditional Statements - Testing and Loops

3. C PROGRAMMING ON UNIX **15**

   Dynamic Storage Allocation-Pointers-Functions-File Handling

TOTAL: 45 PERIODS

HARDWARE / SOFTWARE REQUIREMENTS FOR A BATCH OF 30 STUDENTS

Hardware

1 UNIX Clone Server

3 Nodes (thin client or PCs)

Printer – 3 Nos.

Software

OS – UNIX Clone (33 user license or License free Linux)

Compiler - C

INDEX

EX.NO:1 **STUDY OF UNIX OPERATING SYSTEM**

Date:

Objective: To learn the fundamentals of UNIX Operating System.

Outcome: Acquire knowledge about UNIX Operating System and its various aspects.

Pre-request/Theme: Students should know the basic concepts of a Computer operating system.

Description:

OPERATING SYSTEM

An operating system (commonly abbreviated OS and O/S) is the software component of a computer system that is responsible for the management and coordination of activities and the sharing of the limited resources of the computer. It is a set of programs that provides the interface between user and computer hardware. Controlled allocation of the processors, memories, and I/O devices among the programs are allocated by the operating systems efficiently. So the Operating System is also known as resource allocator. Operating systems offer a number of services to application programs and users. Applications access these services through application programming interfaces (APIs) or system calls.

TYPES OF OPERATING SYSTEM

1. Single User - The system will have its own hard disk, memory, CPU and other resources all dedicated to a single user. E.g. MS-DOS

2. Multi User- The users have access to a multi-user system will have just a terminal and a keyboard. The other resources such as hard disk, printers are centrally located. The user is expected to simply hook onto his account, perform the work, disconnect and leave quietly.

E.g. UNIX

UNIX HISTORY

The spade work for UNIX began at AT&T Bell Laboratories in 1969 by Ken Thompson and Dennis Ritchie. The OS was initially known as UNICS (UNiplexed Information and Computing System). In 1970 UNICS finally became UNIX. In 1973, UNIX was rewritten in Candauthored.UNIX operating systems are widely used in both servers and workstations.

UNIX was designed to be portable, multi-tasking and multi-user in a timesharing configuration. UNIX systems are characterized by various concepts: the use of plain text for storing data; a hierarchical file system; treating devices and certain types of inter-process communication (IPC) as files; and the use of a large number of software tools, small programs that can be combined together through a command line interpreter. These concepts are known as the UNIX philosophy.

UNIX FEATURES

1. **Multi-user system**—Multi-user capability of UNIX allows several users to use the same computer to perform their tasks. Several terminals [Keyboards and Monitors] are connected to a single powerful computer [UNIX server] and each user can work with their terminals.
2. **Multi-tasking system**—Multitasking is the capability of the operating system to perform various task simultaneously, i.e. a user can run multiple tasks concurrently.
3. **Programming Facility**—UNIX is highly programmable, the UNIX shell has all the necessary ingredients like conditional and control structures, etc.
4. **Security**—UNIX allows sharing of data; every user must have a single login name and password. So accessing another user's data is impossible without his permission.
5. **Portability**—UNIX is portable because it is written in a high level language. So, UNIX can be run on different computers.
6. **Communication**—UNIX supports communication between different terminals of the same server as well as between terminals on different servers. Apart from these features, UNIX has an extensive Tool kit, exhaustive system calls and Libraries and enhanced GUI (X Window).

ORGANIZATION OF UNIX

The UNIX system is functionally organized at three levels :

1. The kernel, which schedules tasks and manages storage.
2. The shell (Command Interpreter), which connects and interprets users' commands, calls programs from memory, and executes them.
3. The tools and applications that offer additional functionality to the OS.

COMPONENTS OF UNIX OPERATING SYSTEM

Kernel

The kernel is the heart of the system, a collection of programs written in C that directly communicate with the hardware. It manages the system resources, allocates time between user and processes, decides process priorities, and performs all other tasks. It's that part of UNIX system that is loaded into memory when the system is booted. So we can define kernel as the Master program or often called the Operating system.

Shell

The user cannot directly interact with the kernel. During the login of the user, the kernel starts an interactive program for each user. This program is known as shell. It is actually the interface between the user and the kernel. When user gives some command the shell analyses those commands and passes them to the kernel. The kernel then submits these commands to hardware and the required actions are carried out.

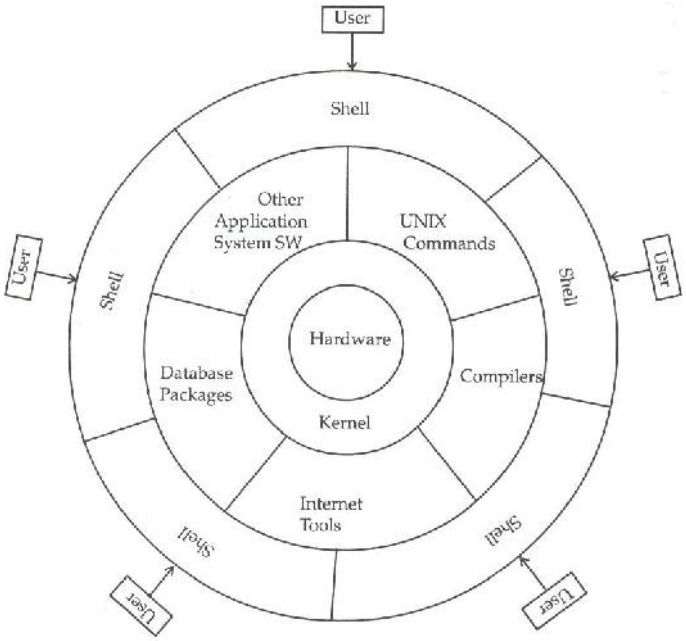


Fig 1.1: Structure of UNIX Operating System

FILES AND DIRECTORIES

A file is a collection of information that is assigned a name that is used to identify that file. The file is always stored in secondary storage medium. A directory is a special type file that contains a list of file names. All files are grouped together into directory for easier access. A directory can have one or more directories in it called sub-directories. In LINUX, files and directories are arranged in a hierarchical manner.

The files can be broadly classified as follows:

Ordinary files- Contains stream of data. All data, text, source programs, object and executable code, commands fall into this category.

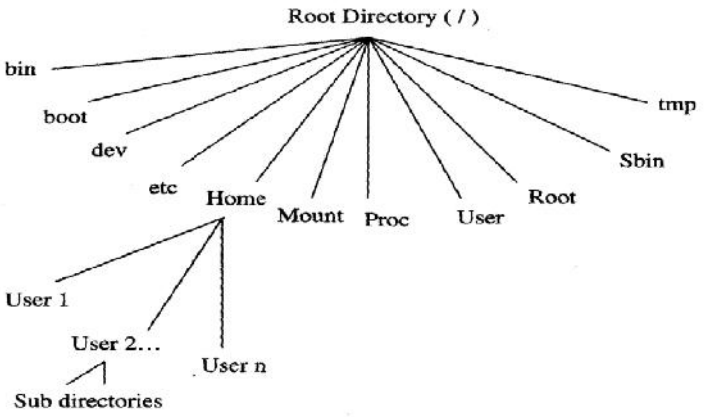
Directory files- Contains no external data. It contains an entry, name of the file and its identification number for each file and subdirectory under that directory. Directory files are not created by the user but by the UNIX system itself.

Device files - Even physical devices are treated as files. These are special in the sense that any output directed to it will be reflected onto the respective device.

UNIX FILE SYSTEM

All files in UNIX are related to one another. The file system of UNIX resembles a tree that grows from top to bottom. The file system begins with a directory called root (at the top). The root directory is denoted by a slash (\). Branching from root there are several directories such as bin, lib, etc, tmp, dev. Each of these directories contains several sub-directories and files.

***Fig 1.2:** UNIX File System*



The details of the different types of the Directories are given below:

Directory	Contents of Directories
/ bin	executable files
/ boot	Linux kernel
/ dev	device files
/ etc	administrative files
/ home	User area
/ lib	C compiler and subroutines
/ mnt	Mount information of floppy and CD
/ proc	process information files
/ root	home area
/ sbin	configuration and system running files
/ tmp	temporary files
/ usr	user file system
/ var	logs, spools for printing and other files.

CONCLUSION:

Thus the study of Unix Operating System has been completed successfully.

VIVA QUESTIONS:

1. Define Operating System.
2. What is kernel?
3. What is a UNIX shell?
4. What is the difference between multi-user and multi-tasking?
5. How is UNIX different from Linux?
6. List some features of UNIX.
7. What is piping?
8. How are the files and directories organized in UNIX?
9. How does the kernel differentiate device files and ordinary files?
10. What are the types of file system in UNIX?

MULTIPLE CHOICE QUESTIONS:

1. The operating system controls
   a) the hard drive b**)** the processor c**)** printers d**)**all of the above

2. Which of the following is not one of the 3 general components of the UNIX operating system?
   a**)** the kernel b**)** the shell c**)** the GUI d**)** the file system

3. The shell does the following in UNIX.
   a**)** Is the user interface b**)** Provides security to files
   c**)** Talks to the hardware d**)** Is the file manager

4. Which of the following statement is FALSE?
   a) UNIX supports multiple users
   b) Linux is an open source operating system and the source code is shared
   c) Shell takes care of inter process communication
   d) Shell provides the feature of I/O Redirection

5. Which among the following is used to write small programs to control Unix functionalities?
   a) Shell Commands b) Shell Script c) Filters d) C Language

6. Links are
   a**)** Like pipes b**)** point to other files
   c**)** Dummy filenames in the directory d**)** both B and C

7. Which of the following is not a component of a user account?
   a**)** Home directory b**)** password c**)** group ID d**)** kernel

8. The Superuser can

a) Create accounts b) Delete accounts

c) Read and write files in user accounts d) All the above

9. Who needs to possess the superuser account?

a) Ordinary users b) System administrator

c) Department manager d) President of the company

10. To use a Unix system with a GUI you need this type of window to enter Unix commands

a) Terminal b) Dialog box c) Dos d) Command

EX.NO:2 **STUDY OF BASIC SHELL COMMANDS**

Date:

Objective: To study and execute the shell commands.

Outcome: work with basic shell commands.

Pre-request/Theme: Students should know the basic commands of DOS.

Description:

UNIX is security conscious, and can be used only by those persons who have an account. *Telnet* (Telephone Network) is a Terminal emulator program for TCP/IP networks that enables users to log on to remote servers.

To *logon,* type telnet server_*ipaddress* in run window.

User has to authenticate himself by providing *username* and *password*. Once verified, a greeting and $ prompt appears. The shell is now ready to receive commands from the user. Options suffixed with a hyphen (–) and arguments are separated by space.

UNIX Shell:

A UNIX shell is a command-line interpreter that provides a user interface for the UNIX operating system . Users direct the operation of the computer by entering commands as text for a command line interpreter to execute or by creating text scripts of one or more such commands.

The commands can be combined using the pipeline (|) operator.

For example, number of users logged in can be obtained as → who | wc –l

COMMAND SHELL: It is a program that interprets commands. It allows a user to execute commands by typing them manually at a terminal, or automatically in programs called shell scripts. A shell is not an operating system. It is a way to interface with the operating system and run commands.

BASH = Bourne Again Shell.

Bash is a shell written as a free replacement to the standard Bourne Shell (/bin/sh) originally written by Steve Bourne for UNIX systems. It has all of the features of the original Bourne Shell, plus additions that make it easier to program with and use from the command line. Since it is Free Software, it has been adopted as the default shell on most Linux systems.

General commands:

Command	Function
date	Used to display the current system date and time.
date + %D	Displays date only
date + %T	Displays time only
date + %Y	Displays the year part of date
date +%H	Displays the hour part of time
cal	Calendar of the current month
cal *year*	Displays calendar for all months of the specified year
cal *month year*	Displays calendar for the specified month of the year
who	Get the information about all the users currently working in the system
who am i	It is used to know in which terminal the user is currently logged on
id	It is used to display the login name.
Tty	Used to display the terminal name
uname	Displays the Operating System
uname–r	Shows version number of the OS (kernel).
uname–n	Displays domain name of the server
echo "*txt*"	Displays the given text on the screen
echo $HOME	Displays the user's home directory
Bc	Basic calculator. Press **Ctrl+d** to quit
Lp *file*	Allows the user to spool a job along with others in a print queue.
man *cmdname*	Manual for the given command. Press **q** to exit
history	To display the commands used by the user since logon.
Exit	Exit from a process. If shell is the only process then logs out

Directory commands:

Command	Function
Pwd	Path of the present working directory
mkdir *dir*	A directory is created in the given name under the current directory
mkdir *dir1dir2*	A number of sub-directories can be created under one stroke
cd *subdir*	Change Directory. If the *subdir* starts with/then path starts from **root**(absolute)otherwise from current working directory.
cd	To switch to the home directory.
cd/	To switch to the root directory.
cd ..	To move back to the parent directory
rmdir subdir	Removes an empty sub-directory.

File commands:

Command	Function
cat >*filename*	To create a file with some contents. To end typing press **Ctrl+d**. The >symbol means redirecting output to a file.(< for input)
cat *filename*	Displays the file contents.
cat >>*filename*	Used to append contents to a file
cp *src des*	Copy files to given location. If already exists, it will be overwritten
cp–i*src des*	Warns the user prior to overwriting the destination file
cp–r*src des*	Copies the entire directory, all its sub-directories and files.
mv *old new*	To rename an existing file or directory. –i option can also be used
mv *f1 f2 f3 dir*	To move a group of files to a directory.
mv–v *old new*	Display name of each file as it is moved.
file *filename*	Determine the type of file
rm *file*	Used to delete a file or group of files. –i option can also be used
spell *filename*	Find the spelling errors in the file
rm*	To delete all the files in the directory.
rm–r*	Deletes all files and sub-directories
rm–f*	To forcibly remove even write-protected files
ls	Lists all files and subdirectories (blue colored) in sorted manner.
ls *name*	To check whether a file or directory exists.
ls *name**	Short-hand notation to list out file names of a specific pattern.
ls–a	Lists all files including hidden files (files beginning with**.**)
ls–x *dir name*	To have specific listing of a directory.
ls–R	Recursive listing of all files in the subdirectories
ls–l	Long listing showing file access rights (read/write/execute-**rwx** for user/group/others-**ugo**).
cmp *file1file2*	Used to compare two files. Displays nothing if files are identical.
wc *file*	It produces a statistics of lines (**l**), words (**w**), and characters(**c**).
chmod *perm file*	Changes permission for the specified file.(r=4,w=2,x=1) chmod740*file*setsallrightsforuser,readonlyforgroups and no rights for others

CONCLUSION:

Thus the study and execution of basic shell commands has been completed successfully.

VIVA QUESTIONS:

1. What is the difference between soft and hard links?
2. What is the difference between $* and $@?
3. Write the command to display the date in the form *dd/mm/yyyy*
4. How will you record your login session in the file session.lst?
5. How will you ensure that *bc* displays the result of all divisions using 3 decimal places?
6. State True or False: "/" can be used in a filename
7. In how many ways you can find out what your home directory is ?
8. What does cd do when used without arguments?
9. If rmdirc_progs fails, what could be the possible reasons ?
10. What is the difference between **cat foo** and **cat >foo** ?

MULTIPLE CHOICE QUESTIONS:

1. What will be output of following command?

   $ **echo** "The process id is" $$$$

   a**)** The process **id** is $$ b**)** The process **id** is $<pid>$<pid>

   c**)** The process **id** is <pid><pid> d**)** The process **id** is $$$$

2. What would be the current working directory at the end of the following command sequence?

   $ **pwd**

   /home/user1/proj

   $ **cd** src

   $ **cd** generic

   $ **cd** .

   $ **pwd**

   a) /home/user1/proj b) /home/user1/proj/src

   c) /home/user1 d) /home/user1/proj/src/generic

3. Write the command to display the current date in the form dd/mm/yyyy.

   a) date +%d/%m/%Y b) date +"%d/%m/%Y"

   c) date +/%d/%m/20%y d) date +"/%d/%m/20%y"

4. The command syntax to display the file 'sample.txt' one page at a time is

   a) man sample.txt>more b) cat sample.txt<more

   c) cat sample.txt|more d) None of the above

5. Which one shows the name of the operating system?

   a) uname –n b) uname –r c) uname –o d) uname –m

6. How do you add (append) a file "file1″ to the example.tar file

a) no you cannot add a file to example.tar

b) tar -cvf example.tar file1

c) tar -rvf file1 example.tar

d) tar -evf file1 example.tar

7.In UNIX "echo" is

a) A SHELL variable b) An external command

c) One of the many shells d) An internal command

8.Which option of the kill command sends the given signal name to the specified process?

a) –l b) –n c) –s d) -a

9. Which command removes a directory from directory stack?

a) dirs b) popd c) pushd d) rm

10. Which command puts a script to sleep until a signal is received?

a) sleep b) suspend c) disown d) break

EX.NO:3 **STUDY OF FILTERS AND SEARCHING**

Date:

Objectives: a) To query a data file using filter commands in UNIX

b) To search for a regular expression in a file using grep command in UNIX.

Outcome: solve problems using filtering commands and grep commands in UNIX

Pre-request/Theme: Students should know about File structure and File commands in UNIX.

Description:

Filters are the central commands of the UNIX tool kit. It acts on data file where lines are *records*, *fields* delimited by a character not used by the data (mostly |, default is white space). The output is a set of records and the input file is unaltered by these commands.

Command	Function
head	used to display the first few records (10 records by default)
head stud	Displays first 10 records by default
head -5 stud	Displays first 5 records
head -1 stud | wc –c	length of first record
tail	used to display the last few records (10 records by default)
tail stud	Displays last 10 records by default
tail -5 stud | tee last5	Last 5 records listed & stored in file *last5* using *tee*
cut	Used to extract specific fields. The **d option** specifies the delimiter and **f for** specifying the field list. The **c option** may be used if extraction is done character wise
cut –d | -f 1,3,4 stud	Fields 1,3,4 listed
paste –d | list1 list2	merges two cut files *list1* and *list2*
sort	Reorders the file as per ASCII sequence. The **t** option is used to specify delimiter and **k** option to specify the field
sort stud	Sorted on 1st column by default
sort –t | -k 3 stud	Sort as per 3rd column
sort –c stud	Check if file is sorted using **c** option
sort -t | -k 4,4 -k 3,3 stud	Sorting on secondary keys
sort -t | -nr –k 4 stud	Sort on numeric field using **n** option, **r** for reverse
uniq stud	Display unique entries in a sorted file
tr	Translates characters. Can be used to change text case. It works with standard input <
tr '[a-z]' '[A-Z]' < stud	Changes text to upper case
nl	Display file content with lines numbered. The **s** option is used to specify separator
nl –s "|" stud	Displays entries numbered with separator |
pr [options] <file name >	It is used to display the contents of the file by separating them into
join –a1 f1 f2	It is used to extracts common lines from two sorted files

A frequent requirement is to look for a pattern or expression in a file. UNIX handles this feature through **grep** and **egrep**. **grep** uses an regular expression to display lines that match and **egrep**enables searching for multiple patterns. Its usage isgrep*optionssearchtext filename*

Command	Function
grep this demo	Lists the lines that contains the string *this*
grep 'end of' demo	Quotes mandatory for text containing space
grep this demo*	Search *this* in multiple files
grep –c to demo	Number of occurrence of the word *to* in the file
grep –n sequence demo	Display line numbers along with matching lines
grep –v word demo	Displays lines that does not contain the text *word*
grep –l vim *	Displays files containing text *vim*
grep –i WORD demo	Search for text ignoring case differences
grep '^[0-9]' demo	Lines that start with a number
grep '[0-9]$' demo	Lines that end with a number
ls -l | grep "^d"	Display the subdirectory names
grep –c "^$" demo	Display count of blank lines in the file.
grep "2....$" stud	Display lines that ends in the range 20000–29999
egrep "lower|UPPER" demo	Display lines that match either *lower* or *upper*
egrep "(previous|current) word" demo	Display lines that match either *previous word* or *current word*

CONCLUSION:

Thus information retrieval using filters has been completed successfully

Thus searching text patterns in files using grep has been completed successfully.

VIVA QUESTIONS:

1. How do you search for a string inside a given file & directory?
2. How will you display only the lines common to two files?
3. What are filters?
4. What does the command in UNIX "$who | sort –logfile > newfile" do?
5. What is the significance of the "tee" command?
6. Man is a filter command. Say true or false.
7. What is the use of 'grep' command in UNIX?
8. Write a command to print the lines that has the the pattern "july" in all the files in a particular directory?
9. What is the command used to translate characters?
10. What is the command to count the number of blank lines?

MULTIPLE CHOICE QUESTIONS:

1. How do you print the lines between 5 and 10, both inclusive

   a) cat filename | head | tail -6 b) cat filename | head | tail -5

   c) cat filename | tail +5 | head d) cat filename | tail -5 | head -10

2. Create a new file "new.txt" that is a concatenation of "file1.txt" and "file2.txt"

   a) cp file.txt file2.txt new.txt b) cat file1.txt file2.txt > new.txt

   c) mv file[12].txt new.txt d) ls file1.txt file2.txt | new.txt

3. The command to search the pattern "Hi there" in file I would be

   a) Grep "Hi there" file 1 b)Grep Hi there file 1

   c)Grep \<Hi there\> file1 d)Grep 'Hi there' file 1

4. The command head f1 would display

   a) First line of the file f1 b) Nothing

   c) First 10 lines of the file f1 d) The whole file f1

5. To delete 5 limes from a file that you are editing and copy them to a buffer named x you would use the command

   a)"x5dd b)"dd5x c)"5xdd d)"d5xd

6. The father of all processes is

   a)Root b)Sh c)Sched d)Init

7. Which option will be used with sort command to start sorting after the nth column of the (m+1)th field

   a) –m.n b)+m.n c)+n.m+1 d)+(m+1).n

8. Which of the following keys is used to move the cursor to the end of the paragraph

   a) } b) { c) | d) $

9. The command which transcribes the standard input to the standard output and also makes a copy of the same in a file is

   a) Tee b) Tr c) Sort d) Grep

10. The message of the day is stored in a file called

    a) /etc/profile b) /etc/motv c) profile d) Autoexec.bat

EX.NO:4 **STUDY OF VI EDITOR**

Date:

Objective: To learn the various features and controls of VI editor.

Outcome: Acquire knowledge about editing text using VI editor and execute UNIX commands.

Pre-requisite: Students should know the various types of commands in UNIX.

vi Editor:

A *text editor* is one of the most common and useful tools in all Operating Systems. Unix provides a versatile editor vi, a full-screen editor and owes its origin to Bill Joy. "vi" stands for *visual* editor. A vi session begins by invoking vi with or without a filename

$vi

$vi*filename*

The user is presented with a full empty screen, each line beginning with a ~. This is vi's way of indicating non-existent lines. Out of 25 lines on the terminal, 24 can be used to enter text. The last line is reserved for commands and also used by the system to display messages. vi functions in three modes namely:

1. **Input mode**—where any key depressed is entered as text
2. **Command mode**—where keys are used as commands to act on text (initial mode)
3. **ex mode**—ex mode commands that can be entered in the last line to act on text

INPUT MODE:

vi starts with command mode. To insert text any of the following commands should be used.

Commands	Function
i	Inserts text to the left of the cursor.
I	Inserts text at the beginning of line.
a	Appends text to right of cursor
A	Appends text at end of line
o	Opens line below
O	Opens line above

In *Input* mode the editor displays **INSERT** in the last line. Press *Enter* key to start a fresh line of text in Input mode and the ~ disappears. To quit *input* mode press *Esc* key.

COMMAND MODE:

EDIT COMMANDS

Command	Function
R	Replaces more than a single character. The editor displays **REPLACE** in the last line. The existing text is overwritten as they are typed.
S	Deletes a single character to the left and switches to **Input**
X	Deletes the character in the current cursor position
?*text*	Locates the *text* in the file. If not found, the message "Pattern not found" appears. Use **n** to repeat the forward search and **N** for backward search.
U or u	Reverses the last change made to the buffer.
Dd	Cuts the entire line
Dw	Cuts the entire word
d$	Cuts a line from cursor position to the end of the line
d0	Cuts from the cursor position to the start of the line
Yy	Copies (yanks) the entire line
Yw	Copies the entire word
P	Pastes the text

NAVIGATION COMMANDS

Command	Function
B	Moves back to beginning of a word
W	Moves forward to beginning of word
|	Moves to start of the line
$	Moves to end of the line
K	Up one line
J	Down one line
H	Left one character
L	Right one character
Ctrl+f	Scrolls a page forward
Ctrl+b	Scrolls a page backward
*l*G	To move to the specific line

One of the most notable features of **vi** is the facility of prefixing a number to most commands. When prefixed, commands interpret the instruction to be repeated as many times.
For example **3x** deletes the next three characters.

THE EX MODE:

The essential save and exit commands form the features of ex mode. Press : (colon) in command mode to switch to ex mode. The : is displayed in the last line. Type the command and press *Enter* key to execute the same.

Command	Function
W	Saves file, displays the number of lines & characters and returns to Input mode. If it is an unnamed file then vi puts a message.
w *file*	The file is saved under the given name
L1,*L2* w *file*	Used to write specific line numbers to some file. The . (dot) represents current line, 0 for first line and $ could be used to represent last line.
Q	Quits vi session and returns to $ prompt. vi has a safety mechanism that warns if changes made to file are not saved.
q!	Quits vi session without saving any changes made since the last
Wq	Save and exit
Sh	Escape to shell
%s/*Sstr*/*Rstr*/g	This is yet another powerful feature known as substitution. It is similar to Find and Replace. % represents all lines, g makes it global. To make vi ask for confirmation before replacing use gc instead of g.
r *file*	To insert another file into the current file.
new *file*	Splits screen into multiple windows and opens the file.

Starting with VI editor:

Syntax:vi filename

Editing the file:

- Open the file using $ vi filename
- To add text at the end of the file, position the cursor at the last character of the file.
- Switch from command mode to text input mode by pressing ‘a’.
- Here ‘a’ stands for append.
- Inserting text in the middle of the file is possible by pressing ‘i’. The editor accepts and inserts the typed character until Esc key is pressed.

Deleting Text

For deleting a character, move the cursor to the character, press ‘x’. The character will disappear.

Conditional Statements in Shell Programming:

The format for the various conditional statements and looping statements and the relational operators used for those conditions are given below.

if condition

```
if condition
then
        execute commands
fi
```

if – else condition

```
if condition
then
        execute commands
else
        execute commands
fi
```

if – elif condition – multi way branching

```
if condition
then
        execute commands
elif condition
then
        execute commands
else
        execute commands
fi
```

Relational Operators:

Operator	*Meaning*
-eq	Equal to
-ne	Not equal to
-gt	Greater than
-ge	Greater than or equal to
-lt	Less than
-le	Less than or equal to

Shell supports a set of loops such as for, while and until to execute a set of statements repeatedly. The body of the loop is contained between do and done statement.

Case condition:

```
case expression in
        pattern1) execute commands ;;
        pattern2) execute commands ;;
        pattern3) execute commands ;;
```

……..

esac

while looping: The **while** loop executes the *statements* as long as the condition remains true.

While *expression*

*d*o

execute commands

done

until: while's complement

The **until** loop complements the while construct in the sense that the *statements* are executed as long as the condition remains false.

until [condition]

for looping: The **for** loop doesn't test a condition, but uses a list instead.

for *variable* in list

*d*o

execute commands

done

CONCLUSION:

Thus the study of text manipulation using vi editor has been completed successfully.

VIVA QUESTIONS:

1. What is the format for vi command?
2. What are the different modes of vi editor?
3. What is the difference between lettered buffer and temporary buffer in vi editor?
4. How to append a file to current file?
5. What is the difference between ZZ and :wq commands?
6. How to enter form command mode to insertion mode in vi editor?
7. What does c$ command do from command mode?
8. How to return to shell without leaving vi editor?
9. Name three ways of exiting **vi** session after saving your work
10. How will you quickly move to the fifth word of a line?

MULTIPLE CHOICE QUESTIONS:

1. Which one of the following statement is not true?

a) vim editor is the improved version of vi editor

b) vi editor commands are not case sensitive

c) vi editor has two modes of operation: command mode and insert mode

d) vi stands for visual editor

2. Which command is used to close the vi editor?

a) q b) wq c) both (a) and (b) d) none of the mentioned

3. In vi editor, the key combination CTRL+f

a) moves screen down one page b) moves screen up one page

c) moves screen up one line d) moves screen down one line

4. Which vi editor command copies the current line of the file?

a) yy b) yw c) yc d) none of the mentioned

5. Which command is used to delete the character before the cursor location in vi editor?

a) X b) x c) D d) d

6. Which one of the following statement is true?

a) autoindentation is not possible in vi editor

b) autoindentation can be set using the command ‘:set ai’

c) autoindentation can be set using the command ‘:set noai’

d) autoindentation is set by default in vi editor

7. Which command searches the string in file opened in vi editor?

a) / or ? b) f or F c) t or T d) none of the mentioned

8. In vi editor, which command reads the content of another file?

a) read b) r c) ex d) none of the mentioned

9. Which command shows all the abbreviations in vi editor?

a) ab b) abb c) show d) none of the mentioned

10. Which command sets the number for all lines?

a) :set li b) :set ln c) :set nu d) :set nl

EX.NO:5 **SHELL PROGRAMMING – SIMPLE SHELL PROGRAM**

Date:

Objective: Learn to write a simple shell program.

.

Outcome: Realize how to solve simple problems using shell script.

Pre-requisite: Students should know the basic commands of shell programming.

Description:

The original UNIX came with the Bourne shell (sh) and it is universal even today. Then came a plethora of shells offering new features. Two of them, C shell (csh) and Korn shell (ksh) has been well accepted by the UNIX fraternity. Linux offers Bash shell (bash) as a superior alternative to Bourne shell.

UNIX system will usually offer a variety of shell types:

- **sh** or Bourne Shell: The original basic shell, a small program with few features still used on UNIX systems and in UNIX-related environments
- **Bash** or Bourne Again shell: The standard GNU shell, intuitive and flexible. Probably most advisable for beginning users while being at the same time a powerful tool for the advanced and professional user. On Linux, **bash** is the standard shell for common users. This shell is a so-called *superset* of the Bourne shell, a set of add-ons and plug-ins.
- **csh** or C shell: The syntax of this shell resembles that of the C programming language.
- **tcsh** or TENEX C shell: a superset of the common C shell, enhancing user-friendliness and speed. That is why some also call it the Turbo C shell.
- **ksh** or the Korn shell: sometimes appreciated by people with a UNIX background. A superset of the Bourne shell; with standard configuration a nightmare for beginning users.

The activities of a shell are not restricted to command interpretation alone. The shell also has rudimentary programming features. When a group of commands has to be executed regularly, they are stored in a file (with extension .sh). All such files are called shell scripts or shell programs. Shell programs run in interpretive mode.

Key factors:

1. Comments in shell script start with #. It can be placed anywhere in a line; the shell ignores contents to its right. Comments are recommended but not mandatory
2. Shell variables are loosely typed i.e. not declared. Their type depends on the value assigned. Variables when used in an expression or output must be prefixed by $.

3. The read statement is shell's internal tool for making scripts interactive.
4. Output is displayed using echo statement. Any text should be within quotes. Escape sequence should be used with –e option.
5. Commands are always enclosed with ` ` (back quotes).
6. Expressions are computed using the expr command. Arithmetic operators are + - * / %. Meta characters * ( ) should be escaped with a \.
7. Multiple statements can be written in a single line separated by ;
8. The shell scripts are executed using the sh command (sh *filename*).

Sample program title: ADDITION OF TWO NUMBERS

ALGORITHM:

Step 1- Open vi editor and enter the program

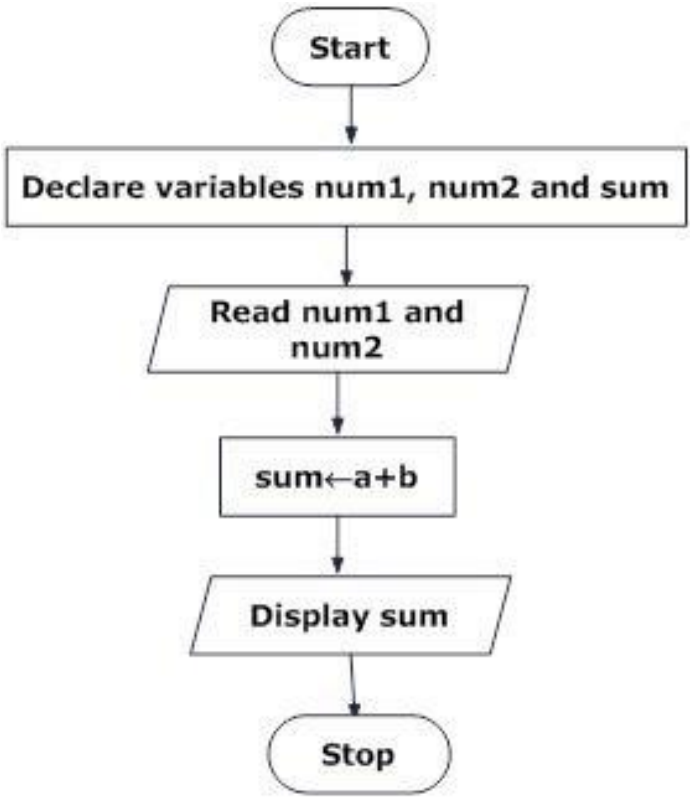
Step 2- Read two numbers from the user.

Step 3- Add both the numbers.

Step 4- Print the summation of two numbers.

Step 5- Stop.

FLOWCHART:



SHELL SCRIPT:

```
echo –n “Enter the 1st number”
read n1
echo –n “Enter the 2nd number”
read n2
sum = `expr $n1 + $n2`
echo “The sum is = $sum”
```

INPUT	OUTPUT
$ Enter the 1st number : 23 Enter the 2nd number : 45	The sum is = 68

CONCLUSION:

Thus the shell script to compute the summation of the two numbers was executed successfully.

Practice Exercises:

1. Write a Shell program to display a given message.
2. Write a Shell Program to find the roots of the quadratic equation.
3. Write a shell script to perform integer arithmetic operations.
4. Write a shell script to getting input details like name, roll number and marks and print them.
5. Write a Shell program to area and circumference of a circle.
6. Write a Shell program to conversion of Celsius to Fahrenheit.
7. Write a Shell program to swap two values.
8. Write a Shell program to simple interest and compound interest.
9. Write a shell program to find the area of a triangle.
10. Write a shell program to find the square and cube of a number.

Viva Questions:

1. Mention the difference between echo "$SHELL" and echo '$SHELL'.
2. Mention the difference between echo * and ls.
3. When we log in, a program starts executing at our terminal. What is this program known as? Name four types of this program that are available on a system.
4. Which command is used to display the processes?
5. State True or False : The wc command is designed to count characters, lines and words only in files
6. State True or False : We can run UNIX commands in uppercase
7. How do you mention the current directory value in $PATH?
8. What is the difference between an argument and an option?
9. If a command doesn't seem to complete, which key will you press to interrupt it?
10. What is the command to execute a shell script?

Multiple Choice Questions:

1. Shell Program is stored in a file called

   a)Unix b)Sh c)Dd d)Cc

2. Which shell offers a command history feature

   a)C shell b)Visual shell c)Bourne shell d)Korn shell

3. After you have entered text in your file, to save and quit vi you will use

   a)Esc shift ZZ b)Shift ZZ c)Esc:q Enter d)None of the above

4. Which of the following assignment is illegal

   a)a='cat file' b)a=100 b=50 c)age=25 d)all the above

5. Which of the following is NOT a shell keyword

   a)Shift b)Readonly c)Unset d)ls

6. When we are executing a shell script the shell acts as

   a)An interpreter b)A compiler c)An operating system d)None of the above

7. A shell variable cannot start with

   a)An alphabet b)A number

   c)A special symbol other than an underscore d)Both b and c above

8. Which of the following statements is correct

   a)a=expr $b + $c b)a=’expr $b * $c’

   c)a=’expr $b * ($c + $d)’ d)a=’expr $b * \($c + $d \)’

9. The expression expr -7 % 2evaluates to

   a)1 b)-1 c)-3.5 d)0

10. In UNIX “echo” is

    a)A SHELL variable b)an external command

    c)one of the many shells d)an internal command

EX.NO:6 **SHELL PROGRAMMING – CONDITIONAL STATEMENTS**

Date:

Objective: Recognize the usage of Conditional Statements in Shell Programming.

Outcome: Get to know how to solve problems involving certain conditions.

Pre-request/Theme: Students should know to solve basic problems using Shell scripts.

Description:

Shell supports decision-making using if statement. The if statement has the following formats. The first construct executes the *statements* when the condition is true. The second construct adds an optional else to the first one that has different set of statements to be executed depending on whether the condition is true or false. The last one is an elif ladder, in which conditions are tested in sequence, but only one set of statements is executed.

Type I :

if [*condition*]
then
statements
fi

Type II :

if [*condition*]
then
statements
else
statements
fi

Type III :

if [*condition*]
then
statements
elif [*condition*]
then
statements
.. .
else
statements
fi

The set of relational and logical operators used in conditional expression is given below. The numeric comparison in the shell is confined to integer values only.

Operator Description

-eq	Equal to
-ne	Not equal to
-gt	Greater than
-ge	Greater than or equal to
-lt	Less than
-le	Less than or equal to
-a	Logical AND
-o	Logical OR
!	Logical NOT

Sample program title: BIGGEST AMONG THREE NUMBERS

ALGORITHM:

Step 1- Open vi editor and start the program

Step 2- Read values of a, b and c

Step 3- If a > b and a > c then

Print "A is the biggest" else if b > c then

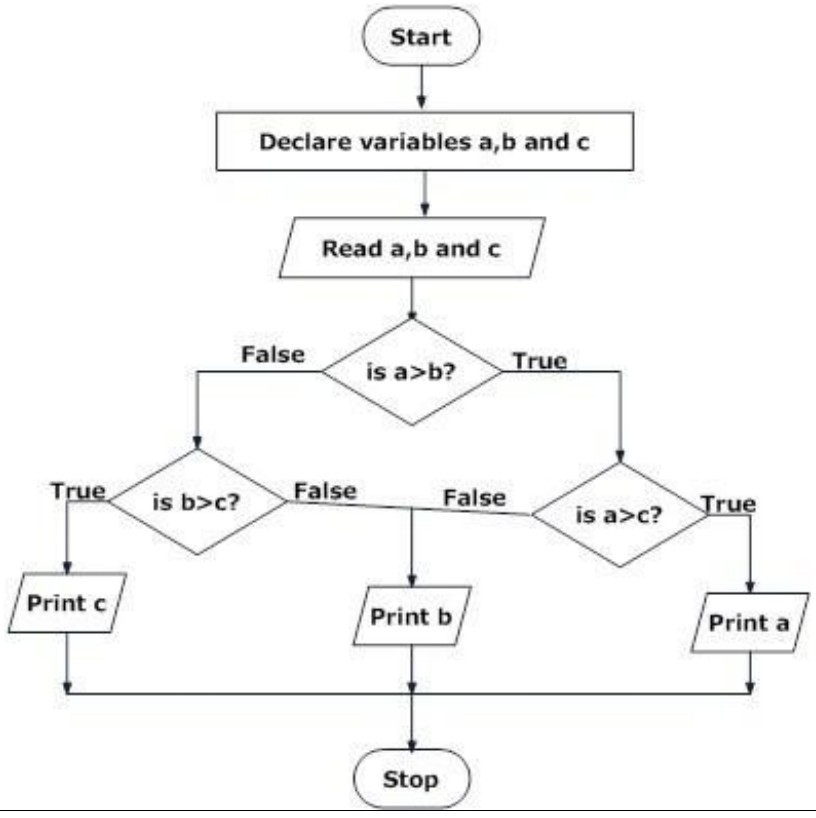
Print "B is the biggest"

else

Print "C is the biggest"

Step 4- Stop

FLOWCHART:



SHELL SCRIPT

```
echo -n "Give value for A B and C: " read a b c
if [ $a -gt $b -a $a -gt $c ]
then
echo "A is the Biggest number" elif [ $b -gt $c ]
then
echo "B is the Biggest number" else
echo "C is the Biggest number"
fi
```

INPUT	OUTPUT
[vijai@localhost decision]$ sh big3.sh Give value for A B and C: 4 3 4	C is the Biggest number

CONCLUSION:

Thus the shell script to find the biggest among three numbers was executed.

Practice Exercises:

1. Write a shell program to check whether the given number is odd or even.
2. Write a shell program to find the minimum among four values.
3. Write a shell program to check whether the input number is prime or not.
4. Write a shell program to determine the given number is positive or negative or zero.
5. Write a shell program to check whether the given year is leap year or not.
6. Write a shell program to determine the grade of a student.
7. Write a shell program to compare the equality of strings.
8. Write a shell program to calculate the employee pay. *(Note check basic pay and change the percentage of HRA, DA, TAX).*
9. Write a shell program to compute the rental charge for travelling in a car, van, jeep and bus.
10. Write a shell program to display the greeting based on system time using nested if statement.

VIVA QUESTIONS:

1. Write the syntax for "if" conditionals in linux
2. What is the difference between the following commands?

```
if [ -e /usr/local/bin/bash ] ; then /usr/local/bin/bash ; fi

if [ -x /usr/local/bin/bash ] ; then /usr/local/bin/bash ; fi
```

3. Given the following variable declarations,

```
HOME=/home/ranga

BINDIR=/home/ranga/bin
```

4. what is the output of the following if statement?

```
if [ $HOME/bin = $BINDIR ] ; then

echo "Your binaries are stored in your home directory."

fi
```

5. What test command should be used in order to test whether /usr/bin is a directory or a symbolic link?

6. Given the following if statement, write an equivalent case statement:

```
if [ "$ANS" = "Yes" -o "$ANS" = "yes" -o "$ANS" = "y" -o "$ANS" = "Y" ] ; then
 ANS="y"
else
ANS="n"
fi
```

7. What is exit status?
8. Represent the if statement using flowchart.
9. What is the function of test command?
10. Mention the usage of logical operators.

MULTIPLE CHOICE QUESTIONS:

1. Flow control in a shell program allows parts of a shell program to be executed
   a) Repeatedly b) Conditionally
   c) All of the above d) None of the above

2. Some lines in an "if statement" are indented because
   a) Required by the shell b) For better readability
   c) Both A and B d) None of the above

3. In "bash", "if" constructs are ended with
   a)Fi b) Endif c)End d)None of the above

4. Variable names may begin with
   a)Numbers b)Underscores c)Alphabetic characters d) Both B and C

5. Which of the following is not a iterative construct in "bash"?
   a)IF b)For c)Until d)While

6. In elif ladder, the conditions are tested
   a)Sequentially b)inorder c)based on size d)none of the above

7. The commands between else and fi block is called
   a)Fi block b) else block c) if block d) none of the above

8. The –ne command means
   a)Equal to b) less than or equal to c) not equal d) not logical

9. The Unix slang for " #! " is
   a)Hash-bang b) Shboom c)Shebang d)More than one of the above

10. What is the output of the following:

```
Echo" enter a number from 1 to 10
Read num
If test  $num lt 6
then
echo " less than 6
echo – Anonymous
```

a) Less than 6 Anonymous b) 6 c) Anonymous d) none

EX.NO:7 **SHELL PROGRAMMING – MULTI WAY BRANCHING**

Date:

Objective: To introduce the concept of Multi-way Branching in shell scripts.

Outcome: Know how to solve problems using Multi-way Branching statements.

Pre-request/Theme: Students should know about conditional Shell scripts.

Description:

The case statement is used to compare a variables value against a set of constants (integer, character, string, range). If it matches a constant, then the set of statements followed after) is executed till a ;; is encountered. The optional *default* block is indicated by *. Multiple constants can be specified in a single pattern separated by |.

Syntax:

```
case variable in
        constant1)
        statements;;
        constant2)
        statements;;
        . . .
        constantN)
        statements;;
        *)
        statements;;
esac
```

Sample program title: SIMPLE CALCULATOR

ALGORITHM:

Step 1- Open vi editor and start the program

Step 2- Read operands a and b

Step 3- Display operation menu

Step 4- Read option

Step 4.1- If option = 1 then

Step 4.2- Calculate c = a + b

Step 4.3- else if option = 2 then Calculate c = a – b

Step 4.4- else if option = 3 then

Step 4.5- Calculate c = a * b

Step 4.6- else if option = 4 then Calculate c = a / b

Step 4.7- else if option = 5 then

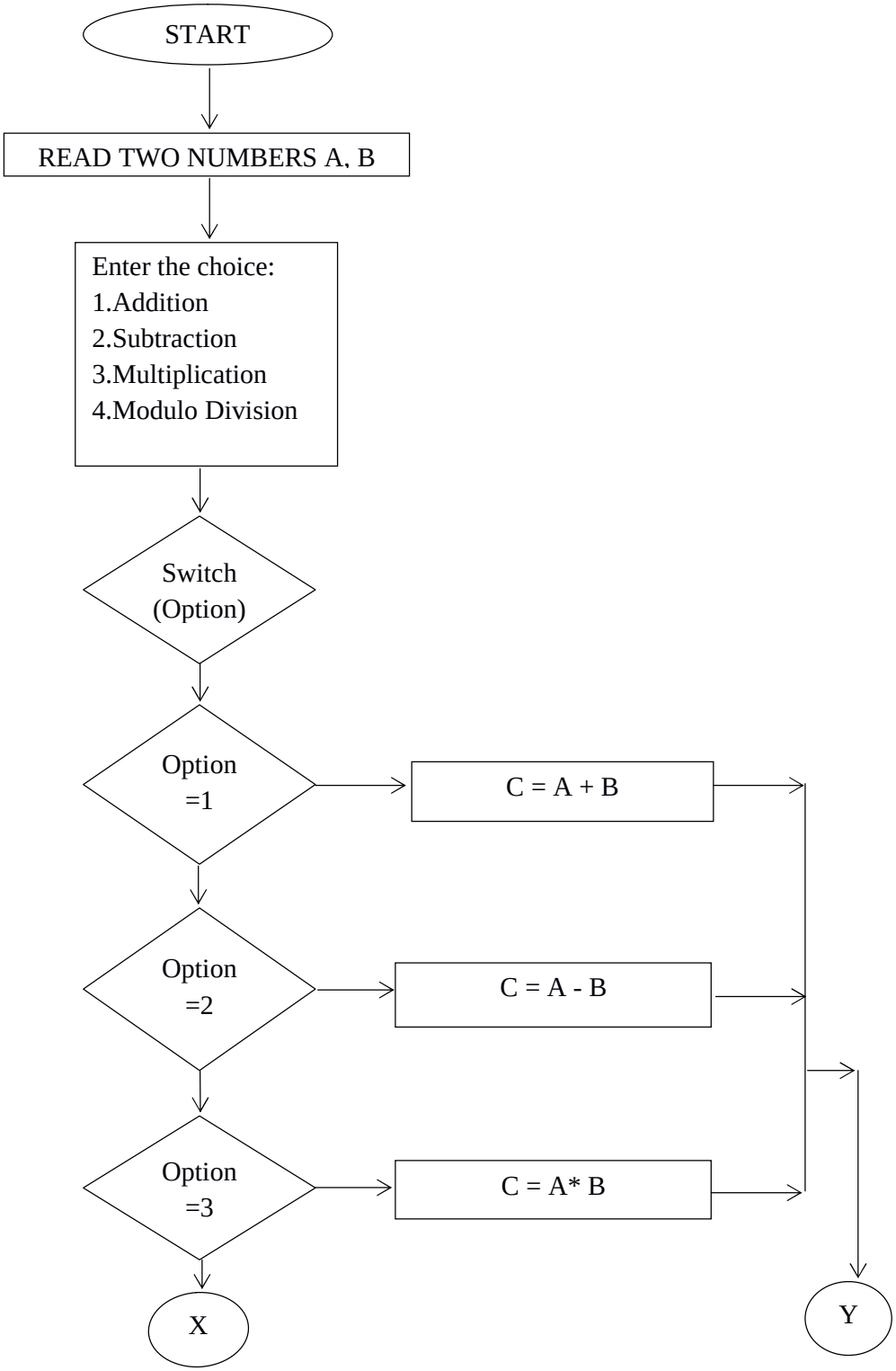
Step 4.8- Calculate c = a % b

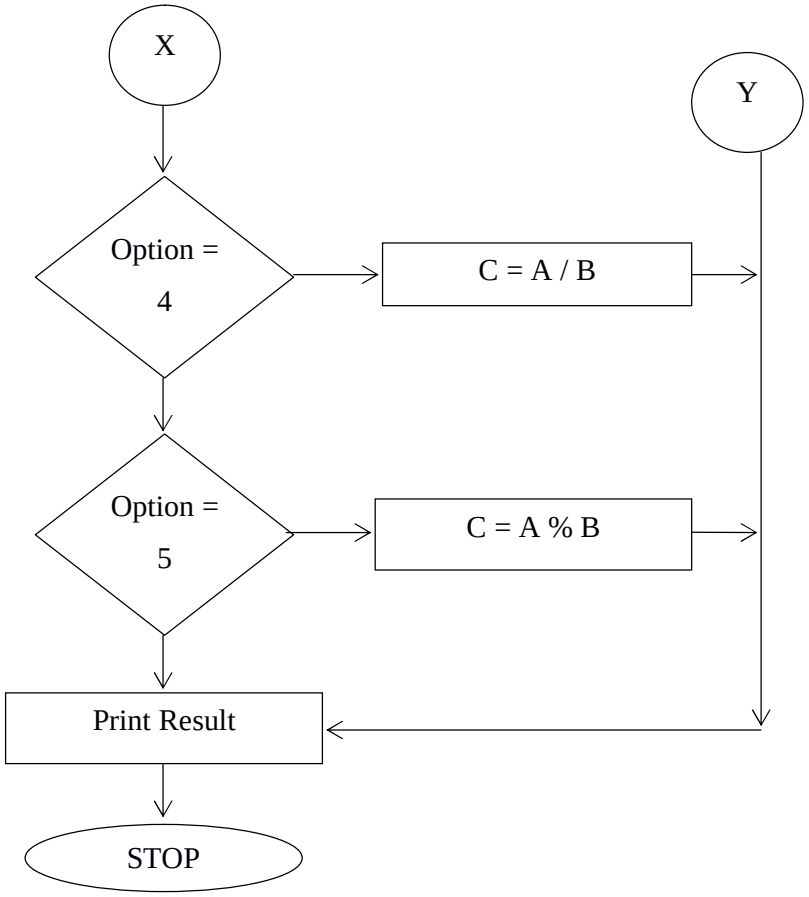
Step 4.9- elsePrint "Invalid option"

Step 5- Print c

Step 6- Stop

FLOWCHART:





SHELL SCRIPT

```
echo -n "Enter the two numbers : "
read a, b
echo " 1. Addition"
echo " 2. Subtraction"
echo " 3. Multiplication"
echo " 4. Division"
echo " 5. Modulo Division"
echo -n "Enter the option : "
read option
case $option in
        1) c=`expr $a + $b`
echo "$a + $b = $c";;
        2) c=`expr $a - $b`
echo "$a - $b = $c";;
        3) c=`expr $a \* $b`
echo "$a * $b = $c";;
```

```
        4) c=`expr $a / $b`
echo "$a / $b = $c";;
        5) c=`expr $a % $b`
echo "$a % $b = $c";;
        *) echo "Invalid Option"
esac
```

INPUT	OUTPUT
[vijai@localhostmultway]$ sh calc.sh Enter the two numbers : 2 4 1. Addition 2. Subtraction 3. Multiplication 4. Division 5. Modulo Division Enter the option : 1	2 + 4 = 6

CONCLUSION:

Thus the shell script to perform arithmetic operations using simple calculator was executed.

Practice Exercises:

1. Write a shell program to find the area of circle, square, rectangle and triangle using case statements.
2. Write a shell program to execute any five UNIX command using case construct.
3. Write a shell program to display the given integer in words using case statements.
4. Write a Shell Script to generate UNIX commands with menu driven program which consists of the following: Menu should contain
    a) To list all file
    b) Current working directory
    c) Calendar
    d) vi editor
    e) Today's date
    f) Current user
5. Write a shell program to perform the various conversions using case constructs
    a) Dollar to rupees
    b) Centimeters to meters
    c) Centigrade to Fahrenheit
6. Write a shell program using case construct to evaluate the hard disk space usage stating the recommended actions to be taken.
7. Write a shell program to identify the range of characters (like lowercase or uppercase or digit, special symbols etc.) while key press using case construct.
8. Write a shell program using case construct to display several messages in contrast with the traffic signals.
9. Write a shell program using case construct to test and implement command line parameter handling.
10. Write a shell program using case construct to display the list of trains after selecting the source and destination.

Viva Questions:

1. How does a case statement look in shell scripts?
2. How do you define a function in a shell script?
3. What are wildcards?
4. What is the keyword used to end a case statement?
5. What will happen if break is not given in case statements?
6. Give simple example illustrating the use of case construct.
7. Justify the limitations of case statements.
8. Differentiate case constructs in c program and shell program.
9. What is conditional follow up?
10. Compare break and continue statement.

Multiple choice questions:

1. The case statements are ended with

   a); b) ;; c) - d) :

2. The optional default block is represented by

   a)* b) & c)# d)@

3. A Sticky bit applies to a file would mean that

   a) No one can remove it

   b) It would stick around in memory even when its execution is over

   c) Next time you login it would get executed on its own

   d) None of the above

4. To find the block size on your file system the command is

   a) Blksz b) Szblk c) Chksz d) Cmchk

5. To the statement *read v1 v2 v3* we can supply

   a) maximum 3 values

   b) only 1 value

   c) exactly 3 values

6. Multiple patterns can be seperated by

   a**) |** b) [c) ! d) < >

7. Which of the following is not a command separator?

   a**) |** **b)** ; c) & d) #

8. Say true or False - UNIX filenames are case sensitive.

9. The pattern is terminated by _____operator

   a) * b) ? c)) d) ()

10. What is the output for the following program?

```
#! /bin/sh
TEST=1111
case $TEST in
1111) echo "1111" ;;
```

```
1112) echo "1112" ;;
*) echo "None"
Esac
```

a) 1111 b)1112 c) both 1111 and 1112 d) error

EX.NO:8 **SHELL PROGRAMMING – TESTING AND LOOPS**

Date :

Objective: Realize the Looping constructs in Shell programming.

Outcome: Know how to solve problems requiring testing and repetitions of statements.

Pre-request/Theme: Students should know about Decision Making and Conditional manipulations.

Sample program title: FIBONACCI SERIES

ALGORITHM:

Step 1- Open vi editor and start the program

Step 2- Read number of terms as n

Step 3- Initialize 0 to f1, 1 to f2 and 2 to i

Step 4- Print initial Fibonacci terms f1, f2

Step 5- Generate next term using the formula f3 = f1 + f2

Step 6- Print f3

Step 7- Increment i by 1

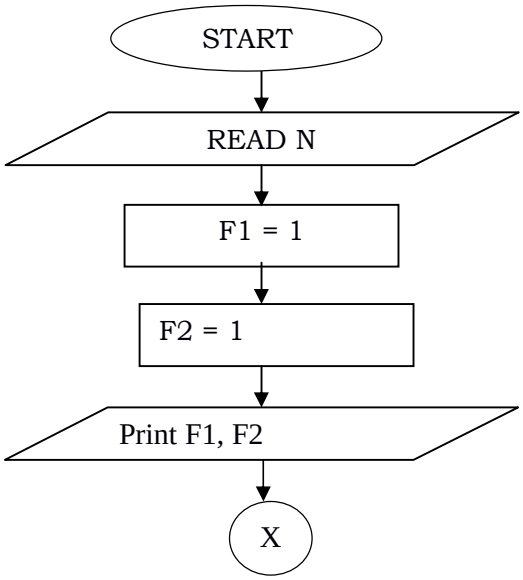
Step 8- Assign f2 to f1

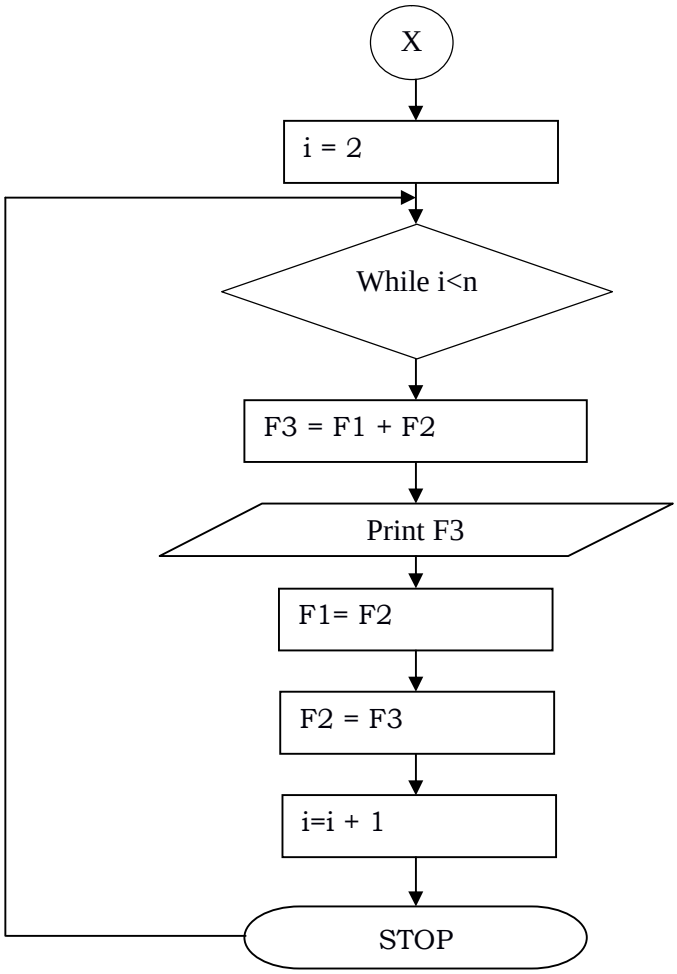
Step 9- Assign f3 to f2

Step 10- Repeat steps 5–9 until i less than n

Step 11- Stop

FLOWCHART:





SHELL SCRIPT:

```
echo -n "Enter number of terms : "
read n
echo "Fibonacci Series"
f1=0
f2=1
echo -n "$f1 "
echo -n " $f2 "
i=2
while [ $i -lt $n ]
do
        f3=`expr $f1 + $f2`
        echo -n " $f3 "
        f1=$f2
        f2=$f3
        i=`expr $i + 1`
done
```

INPUT	OUTPUT
[vijai@localhost loops]$ sh fibo.sh Enter number of terms : 8	Fibonacci Series 0 1 1 2 3 5 8 13

CONCLUSION:

Thus the shell script to generate the Fibonacci series was successfully executed.

Practice Exercises:

1. Write a shell program to check whether the entered is Armstrong or not.
2. Write a shell program to find the factorial of a given number.
3. Write a Shell script to generate Multiplication Table.
4. Write a Simple Shell script to print the sum of n natural numbers.
5. Write a shell script to generate all combinations of 1 2 3 using for loop.
6. Write a shell program to find the smallest digit of a value
7. Write a shell program to count the number of digits of a value.
8. Write a shell program to reverse a number.
9. Write a shell program to count the number of vowels in a string.
10. Write a shell program to find the sum of even and odd numbers in an array.
11. Write a shell program to find the sum of $1^3+2^3+33+\ldots..+N^3$
12. Write a shell program to find the number of blank spaces in a line of text

Viva questions:

1. What are the three methods of repeating a part of a program?
2. Differentiate while and do while
3. How long will the while loop be executed?
4. Give the syntax of for loop in shell programming.
5. What is IFS?
6. Differentiate while and until.
7. What is a control variable in for loop? Give example
8. Give examples for nesting of loops.
9. What is the purpose of break statement?
10. Write a shell program to generate prime numbers between 10 and 100

Multiple choice questions:

1. The shell script has name ls. If you execute ls
   a) The script is generated
   b) Ls command is executed.
   c) Whether script is executed or command is executed depending upon the path
   d) Both script and command executed.
2. **Say true or false:**

   Using for loop, we can calculate the factorial of a number.
3. The break statement is used to exit from

   a) An if statement b) a for loop c) a program d) none of the above
4. An until loop ensure that statements within the loop gets executed.

a) Only once b) Atleast once c) Not even once d) None of the above

5. A loop that continues indefinitely is called a

a) Indefinite loop b) Non-stop loop c) Infinite loop d) None of the above

6. The break statement

a) Jumps out of the innermost for loop

b) Jumps out of the innermost while loop

c) Jumps out of the innermost do-while loop

d) all of the mentioned

7. Which of the following is not logical operator?

a) & b) && c) || d)!

8. What will be the output of the program?

```
#include<stdio.h>
int main()
{
   int i=0;
   for(; i<=5; i++);
      printf("%d", i);
   return 0;
}
```

a)0, 1, 2, 3, 4, 5 b)5 c)1, 2, 3, 4 d)6

9. How many times is a do while guaranteed to loop?

a)0 b) Infinetely c)1 d)variably

10. Which of the following cannot be used as LHS of the expression in for (exp1;exp2; exp3) ?

a) Variable b) Function c) typedef d) macros

EX.NO:9 C PROGRAMMING ON UNIX – DYNAMIC STORAGE ALLOCATION

Date :

Objective: To learn the implementation of Dynamic Storage allocation on UNIX.

Outcome: Realize how to run C program on UNIX.

Pre-request/Theme: Students should know about C programming constructs.

Description:

Memory allocated using arrays is insufficient or abundant, thereby inefficient. To overcome this, memory could be allocated at run-time instead at compile time. The process of allocating memory at run time is known as dynamic memory allocation. C inherently does not have this facility but supports with memory management functions malloc, calloc and realloc, which can be used to allocate and free memory using free during the program execution.

The memory space located between program and local variable is available known as *heap* for dynamic allocation during the execution of the program. The size of heap keeps changing when program is executed due to creation and death of variables. Therefore it is possible to encounter *memory overflow* during dynamic allocation process. In such situations, the memory allocation functions will return a null pointer.

malloc

The malloc function reserves a block of memory of specified size and returns a pointer of type void containing the first byte of the allocated region. Thus it could be casted to any type of pointer. The allocated space contains garbage data.

ptr = (*cast-type**) malloc(*bytesize*);

calloc

calloc is another memory allocation function that is normally used to request multiple blocks of storage each of the same size and then sets all bytes to zero.

ptr =(*casttype**) calloc(*blockcount*, *blocksize*);

free

Compile time storage of a variable is allocated and released by the system in accordance with its storage class. With the dynamic runtime allocation, it is our responsibility to release the space when it is not required, using the free function. The release of storage space becomes important when the storage is limited. free(*ptr*);

realloc

The memory allocated by using calloc or malloc might be insufficient or excess sometimes. In both the situations the memory size already allocated could be changed with the help of

function realloc. This process is called reallocation of memory.

ptr = realloc(*ptr*, *newsize*);

Sample program title: Sort Elements in ascending order using dynamic memory allocation

ALGORITHM:

Step 1- Open vi editor and start the program

Step 2-Declare pointer variables p and a.

Step 3-Read the size s.

Step 4-Using malloc function allocates the memory dynamically to pointer p.

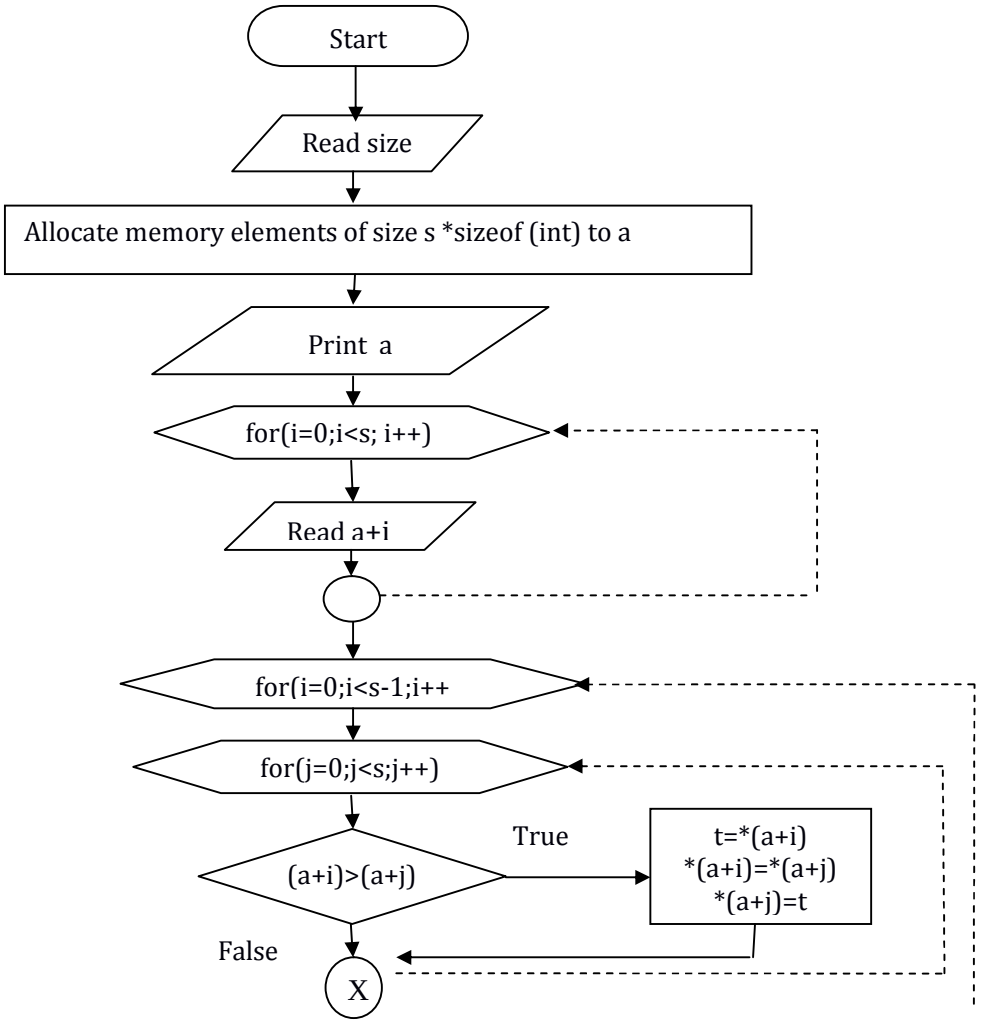
Step 5- Set a for loop to enter the elements.

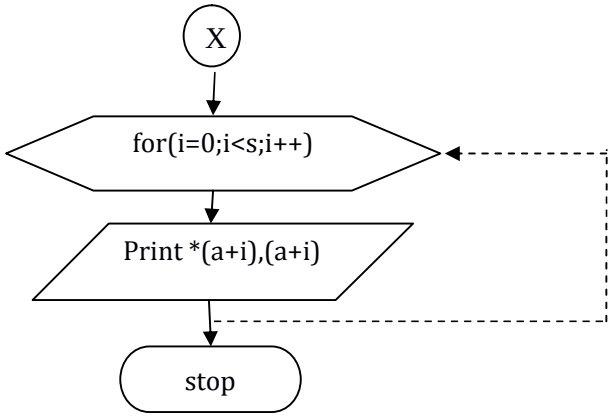
Step 6-Sort the elements in ascending order.

Step 7-Print the elements and their address.

Step 8-Stop.

FLOWCHART:





PROGRAM:

```
/* Dynamic Memory Allocation using malloc() */
#include <stdlib.h>
#include <malloc.h>
#include <stdio.h>
main()
{
    int n, i, *a;
    floatavg, sum=0;
    printf("Enter the No. of students : ");
    scanf("%d", &n);
    a = (int*) malloc(n * sizeof(int));
    if (a == NULL)
    {
        printf("\n memory allocations not possible");
        exit(-1);
    }
    printf("Enter %d marks : ", n);
    for(i=0; i<n; i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0; i<n; i++)
    sum += a[i];
    avg = sum / n;
```

```
    printf("Average mark  : %.2f\n",avg);
    free(a);
}
```

INPUT	OUTPUT
[vijai@localhost pointer]$ gccdynalloc.c [vijai@localhost pointer]$ ./a.out Enter the No. of students : 6 Enter 6 marks : 56 67 78 89 90 98	Average mark : 79.67

CONCLUSION:

Thus the C program for Sorting Elements in ascending order using dynamic memory allocation was successfully executed.

Practice Exercises:

1. Write a C program on UNIX to reallocate memory for strings.
2. Write a C program on UNIX to allocate memory for two dimensional arrays.
3. Write a C program to find the sum of two matrices using dynamic memory allocation
4. Write a C program to combine two arrays without duplicates using dynamic memory allocation
5. Write a C program to add the content of two equal size array and store it into another array using dynamic memory allocation
6. Write a C program to print the common values of two arrays with dynamic memory allocation
7. Write a C program to search an element from a list of values with dynamic memory allocation
8. Write a C program to copy the content of one array to the other in reverse order with dynamic memory allocation
9. Write a C program to transpose of a matrix with dynamic memory allocation
10. Write a C program print the numbers which are divisible by 3 and not divisible by 2 from an array of values with dynamic memory allocation

Viva questions:

1. On freeing a dynamic memory, if the pointer value is not modified, What will the pointer point to?
2. Why do we write (int *) before malloc?
3. What is Heap for memory allocation?
4. Which header file should be included to use functions like malloc() and calloc()?
5. What function should be used to free the memory allocated by calloc() ?
6. Specify the 2 library functions to dynamically allocate memory?
7. What will be the output of the program?

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int *p;
   p = (int *)malloc(20); /* Assume p has address of 1314 */
   free(p);
   printf("%u", p);
   return 0;
}
```

8. When we dynamically allocate memory is there any way to free memory during run time?

Say true or False:

9. malloc() allocates memory from the heap and not from the stack.
10. malloc() returns a NULL if it fails to allocate the requested memory.

Multiple choice questions:

1. The function ____ obtains block of memory dynamically.

a) calloc b) malloc c) Both a & b d) free

2. void * malloc(size_t n) returns

a) Pointer to n bytes of uninitialized storage

b) NULL if the request cannot be satisfied

c) Nothing

d) Both a & b are true

3. calloc() returns a storage that is initialized to.

a) Zero b) Null c) Nothing d) One

4. In function free(p), p is a

a) int b) Pointer returned by malloc()

c) Pointer returned by calloc() d) Both b & c

5. What is the output of this C code?

```
#include <stdio.h>
void main()
{
   char *p = calloc(100, 1);
   p = "welcome";
   printf("%s\n", p);
}
```

a) Segmentation fault b) Garbage c) Error d) welcome

6. Memory allocation using malloc() is done in?

a) Static area b) Stack area c) Heap area d) Both b & c

7. Why do we write (int *) before malloc?

```
int *ip = (int *)malloc(sizeof(int));
```

a) It is for the syntax correctness b) It is for the type-casting

c) It is to inform malloc function about the data-type expected d) none of the mentioned

8. Which one is used during memory deallocation in C?

a) remove (p); b) delete(p); c) free(p); d) terminate(p);

9. calloc initialises memory with all bits set to zero.

a) true b) false c)Depends on the compiler d)Depends on the standard

10. realloc(ptr, size), where size is zero means

a) Allocate a memory location with zero length b) Free the memory pointed to by ptr

c) Undefined behavior d) doesn't do any reallocation of ptr

EX.NO:10 **C PROGRAMMING ON UNIX – POINTERS**

Date :

Objective: To implement the concept of Pointers using C program on UNIX.

Outcome: Realize how to run C program on UNIX with Pointers.

Pre-request/Theme: Students should know about C programming constructs.

Sample program title: REFERENCE AND DEREFERENCEOPERATOR

ALGORITHM:

Step 1- Open vi editor and start the program

Step 2- Initialize*x*to10

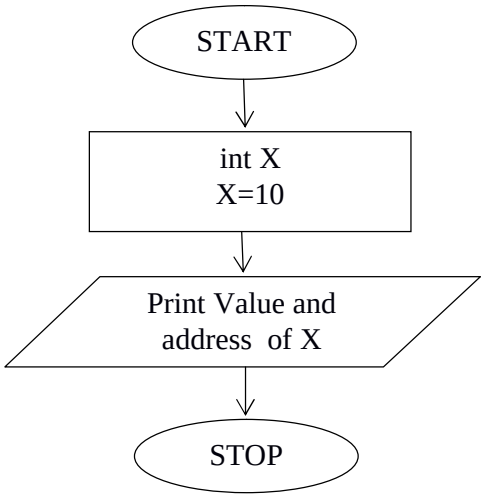
Step 3- Print the value of *x*

Step 4- Print the address of *x* using address operator

Step 5- Print the value of *x* by dereferencing the address operator

Step 6- Stop

FLOWCHART :



PROGRAM:

```
/*Data and address*/
#include <stdio.h>
main ()
{
```

```
    int x;
    x=10;
    printf("Value of x is %d" ,x);
    printf("\nAddress of x is %u", &x);
    printf("\nValue of x is %d\n",*(&x));
}
```

INPUT	OUTPUT
[vijai@local host pointer]$ gcc refderef.c [vijai@localhost pointer]$ ./a.out	Value of x is 10 Address of x is 3221216948 Value of x is 10

CONCLUSION:

Thus the C program to print value & address of a variable using pointer was successfully executed.

Practice Exercises:

1. Write a C program to implement Pass by reference.
2. Write a C program to add given two numbers using pointers
3. Write a 'C' program to sort 'N' numbers. Use pointers.
4. Write a 'C' program to demonstrate Pointer to Pointer (double pointer).
5. Write a 'C' program using pointers to find the sum of one dimensional array.
6. Write a 'C' program to copy a string using pointers
7. Write a 'C' program to display the student details.
8. Write a 'C' program to generate employee payroll using Pointers and Unions.
9. Write a 'C' program to illustrate Pointers and structures.
10. Write a 'C' program to access elements of an array using Pointers.

Viva questions:

1. Define pointer
2. What is constant pointer?
3. Give examples for initialization of pointer
4. What is the purpose and advantage of pointers?
5. What is a double pointer?
6. What is dangling pointer?
7. Can we assign a pointer to a function?
8. Are the expression *ptr++ and ++*ptr are same?
9. What is null pointer?
10. Give simple example of Pointers to arrays.

Multiple choice questions:

1. A pointer is

   a)keyword to create variables b) a variable that stores the address of instruction

   c) a variable that stores the address of another variable d) all the above

2. The operator used for dereferencing is ____

   a) * b) ^ c) $ d) &

3. What is (void*)0?

   a) Representation of NULL pointer b) Representation of void pointer

   c) Error d) none of above

4. Can you combine the following two statements into one?

```
char *p;
p = (char*) malloc(100);
```

a) char p = *malloc(100); b) char *p = (char) malloc(100);

c) char *p = (char*)malloc(100); d) char *p = (char *)(malloc*)(100);

5. In which header file is the NULL macro defined?

a) stdio.h b) stddef.h c) stdio.h and stddef.h d) math.h

6. How many bytes are occupied by near, far and huge pointers (DOS)?

a) near=2 far=4 huge=4 b) near=4 far=8 huge=8

c) near=2 far=4 huge=8 d) near=4 far=4 huge=8

7. If a variable is a pointer to a structure, then which of the following operator is used to access data members of the structure through the pointer variable?

a) . b) & c) * d) ->

8. Correct syntax to pass a Function Pointer as an argument

a) void pass(int (*fptr)(int, float, char)){}

b) void pass(*fptr(int, float, char)){}

c) void pass(int (*fptr)){}

d) void pass(*fptr){}

9. Which of the following is not possible in C?

a) Array of function pointer b) Returning a function pointer

c) Comparison of function pointer d) none of the mentioned

10. Which of the following does not initialize ptr to null (assuming variable declaration of a as int a=0;?

a) int *ptr = &a; b) int *ptr = &a – &a;

c) int *ptr = a – a; d) All of the mentioned

EX.NO:11 **C PROGRAMMING ON UNIX – FUNCTIONS**

Date :

Objective: To solve problems with the idea of functions using C program on UNIX.

Outcome: Learn to implement C Functions on UNIX.

Pre-request/Theme: Students should know about C programming constructs.

Sample program title: SUM OF DIGITS

ALGORITHM:

Step 1- Open vi editor and start the program

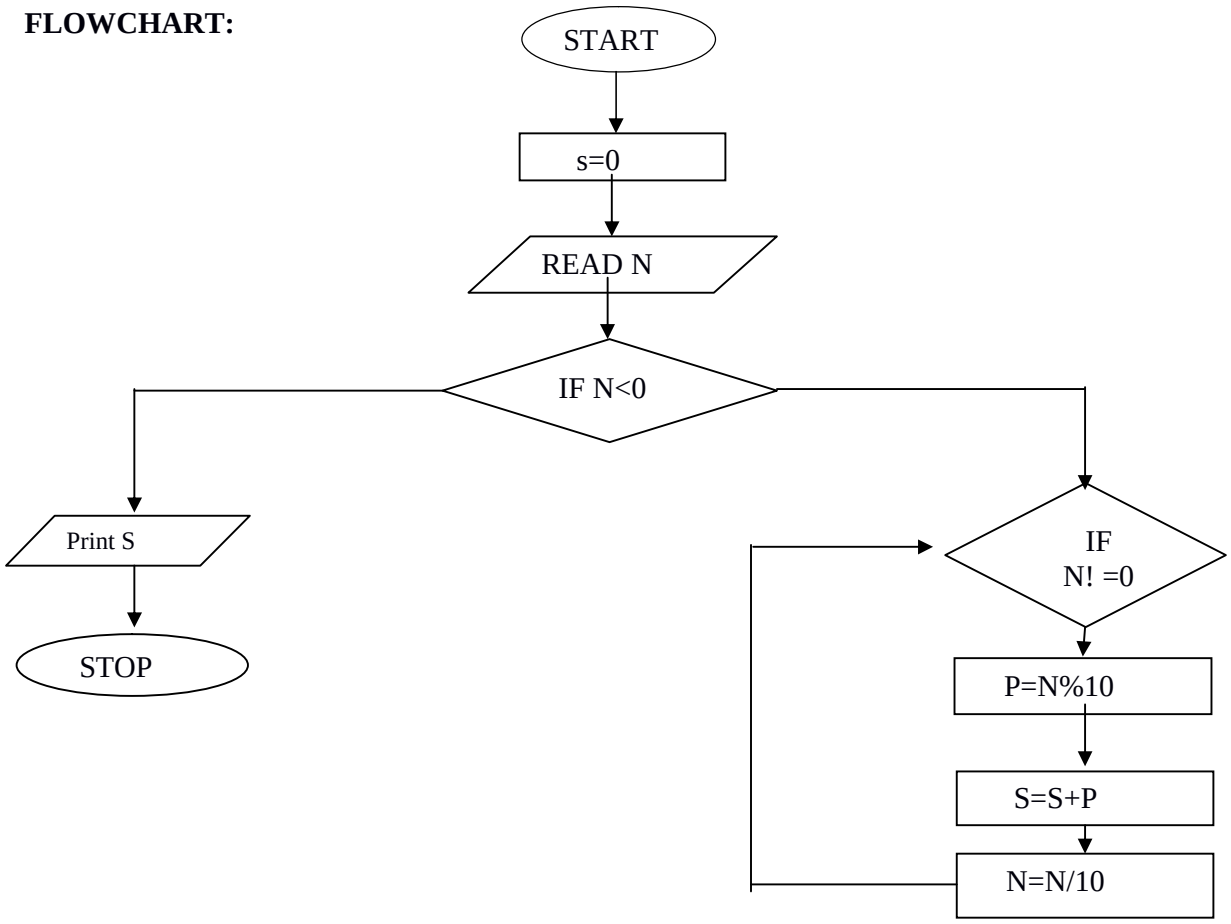
Step 2- Get number from user. Pass it as argument to function sum.

Step 3- Find remainder out each individual digit by means of modulo division and calculate sum by adding them.

Step 4- Display output.

Step 5- Stop the program.

FLOWCHART:



PROGRAM:

```
#include<stdio.h>
void sum(int);
main()
{
 int no;
 printf("Enter your number\n");
 scanf("%d",&no);
 sum(no);
}

void sum(int num)
{
 int a,sum=0;
 while(num>0)
 {
  a=num%10;
  sum=sum+a;
  num=num/10;
}
 printf("Sum=%d",sum);
}
```

INPUT	OUTPUT
[com35@localhost ~]$ gcc sumdigi.c [com35@localhost ~]$ ./a.out Enter your number 12345	Sum=15

CONCLUSION:

Thus the program to find the sum of digits was successfully executed and output was obtained.

Practice Exercises:

1. Write a C program to perform linear search using function.
2. Write a C program to find the reverse of a number using functions.
3. Write a C program to find the length of a string.
4. Write a C program to check the given number is Palindrome or not using Functions.
5. Write a C program to find the factorial of a number using recursive functions.
6. Write a c program to find the sum of all the digits of a given multi-digit number using "function with arguments and return values".
7. Write a C program to find the greatest common divisor (GCD) and least common multiple (LCM) of two integers.
8. Write a C program to find the sum of 'n' numbers.

Viva questions:

1. Define function prototype
2. What are the types of function prototypes?
3. Can functions return structure in c?
4. Give the syntax for declaring a Function.
5. What is a Overloaded function?
6. Define recursive function.
7. Give simple example for performing aaddition operation using functions.
8. What is call by value and call by reference?
9. What is nesting of functions?
10. How many times the program will print "Salem" ?

```
#include<stdio.h>
int main()
{
printf("Salem");
main();
return 0;
```

Multiple choice questions:

1. Which is true about function tolower?
   a) The function tolower is defined in b) Converts an upper case letter to lower case
   c) returns other characters untouched d)None of the mentioned
2. Which of the following function can be used to terminate the main function from another function safely?
   a) return(expr); b) exit(expr); c) abort(); d) Both b and c

3. The keyword used to control from a function back to the calling function is

   a) switch b) goto c) go back d) return

4. Which of the following is a correct format for declaration of function?

   a) return-type function-name(argument type);

   b) return-type function-name(argument type){}

   c) return-type (argument type)function-name;

   d) Both (a) and (b)

5. Which of the following function declaration is illegal?

   a) int 1bhk(int); b) int 1bhk(int a);

   c) int 2bhk(int*, int []); d) All of the mentioned

6. Can we use a function as a parameter of another function? [Eg: void wow(int func())]

   a) Yes, and we can use the function value conveniently

   b) Yes, but we call the function again to get the value, not as convenient as in using variable

   c) No, C does not support it.

   d) This case is compiler dependent

7. The value obtained in the function is given back to main by using _____ keyword?

   a) return b) static c) new d) volatile

8. What is the return-type of the function sqrt()

   a) int b) float c) double d) Depends on the data type of the parameter

9. What is the problem in the following declarations?

```
int func(int);
double func(int);
int func(float);
```

   a) A function with same name cannot have different signatures

   b) A function with same name cannot have different return types

   c) A function with same name cannot have different number of parameters

   d) All of the mentioned

10. What is the default return type if it is not specified in function definition?

    a) void b) int c) double d) short int

EX.NO:12 **C PROGRAMMING ON UNIX –FILE HANDLING**

Date :

Objective: To perform File handling operations using C program on UNIX.

Outcome: Realize how to handle files with C program on UNIX.

Pre-request/Theme: Students should know about C programming constructs.

Description:

Many applications require information stored on auxiliary storage device. Such information is stored permanently as a data file that allows accessing and altering the information whenever necessary.

Opening a File

Prior to performing any activity of a file, the file should be opened. By opening a file, link between the program and the operating system is established. This link exists by means of a structure named FILE, in header file *stdio*.h. Therefore, it is mandatory for programs pertaining to file should include <stdio.h>.When a request is made for a file to be opened, the operating system returns a pointer to the structure FILE. The pointer is declared as FILE **fileptr*;

fopen():

A file is opened using the standard function fopen(). The file to be opened is searched in the disk. If a file could not be opened, a NULL is returned. *fileptr* = fopen("*filename*", "*mode*");

The file mode specifies the purpose of opening a file. Some of them are

Mode Description

r - Open an existing file for reading only

w - Open a new file for writing only. If the file exists, its contents are destroyed

a - Open an existing file for appending. If the file doesn't exist, a new file is created

fread() and fwrite() :

The functions fread/fwrite is used for reading/writing data from/to the file opened by fopen function. These functions accept three arguments. The first argument is a pointer to buffer used for reading/writing the data. The data read/written is in the form of ‘nmemb’ elements each ‘size’ bytes long.

In case of success, fread/fwrite return the number of bytes actually read/written from/to the stream opened by fopen function. In case of failure, a lesser number of byes (then requested to read/write) is returned.

fseek():

The fseek() function is used to set the file position indicator for the stream to a new position. This function accepts three arguments. The first argument is the FILE stream pointer returned by the

fopen() function. The second argument 'offset' tells the amount of bytes to seek. The third argument 'whence' tells from where seek of 'offset' number of bytes is to be done.

fclose():

*int fclose(FILE *fp);*

The fclose() function first flushes the stream opened by fopen() and then closes the underlying descriptor. Upon successful completion this function returns 0 else end of file (eof) is returned. In case of failure, if the stream is accessed further then the behavior remains undefined.

The I/O operation is done using any of the following functions.

getc & putc

The *getc* function is used to read a character from the file opened in read mode and assigns it to a character variable. The getc will return an EOF marker when the end-of-file is reached. Thereafter reading is not possible. The *putc* function writes the character contained in the variable onto a file opened in write mode. The file pointer moves by one character position for every character I/O operation. *charvar* = getc(*fileptr*); *putc*(*charvar*, *fileptr*);

fgets & fputs

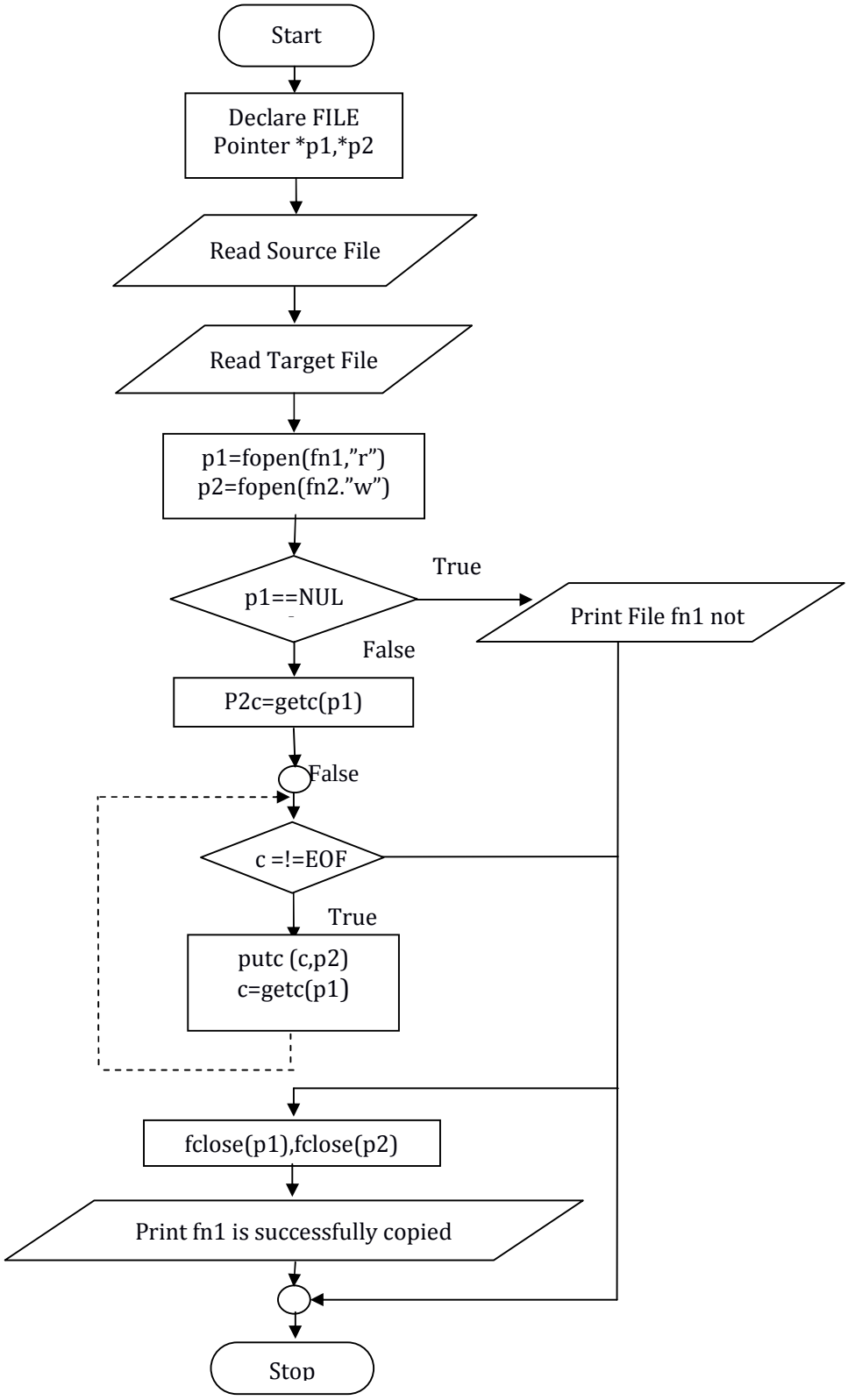
Rather than single characters, to deal with strings *fgets* and *fputs* are used in a similar manner.

Sample program title: COPY THE CONTENTS OF ONE FILE TO ANOTHER FILE

ALGORITHM:

Step 1- Open vi editor and start the program
Step 2- Declare two file Pointer variable p1 and *p2
Step 3- Read the source filename fn1
Step 4- Read the target filename fn2
Step 5- Assign p1=open the file fn1 in read mode
Step 6- Assign p2=open the file fn1 in write mode
Step 7- Test the condition if p1 is equal to NULL, true
    Print file fn1 not exist
Step 8- If the condition fails
    Read a character from File fn1 and assign to c
Step 9- While c is not equal to end of file and do
        Putc on File pointer *p2 position
        Read a character from File fn1 and assign to c
Step 10-Close the file pointer p1 and p2

FLOWCHART:


Start
Declare FILE Pointer *p1,*p2
Read Source File
Read Target File
p1=fopen(fn1,"r")
p2=fopen(fn2."w")
p1==NUL
True
Print File fn1 not
False
P2c=getc(p1)
False
c =!=EOF
True
putc (c,p2)
c=getc(p1)
fclose(p1),fclose(p2)
Print fn1 is successfully copied
Stop

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
FILE *p1,*p2;
int c;
char fn1[40],fn2[40];
printf("\n Enter the source file ");
scanf("%s",fn1);
printf("\n Enter the Target file");
scanf("%s",fn2);
p1=fopen(fn1,"r");
p2=fopen(fn2,"w");
if(p1==NULL)
        {
        printf("File %s  not exist",fn1);
        }
else
        {
        c=getc(p1);
        while(c!=EOF)
                {
                putc(c,p2);
                c=getc(p1);
                }
fclose(p1);
fclose(p2);
printf("file %s is  successfully copied to file %s",fn1,fn2);
}
}
```

INPUT	OUTPUT
[com35@localhost ~]$ cat f1 Varuvan vadivelan Institute of Technology Gundalpatti [com35@localhost ~]$ cc exf1.c [com35@localhost ~]$ ./a.out Enter the source file f1 Enter the Target file f2	file f1 is successfully copied to file f2 [com35@localhost ~]$ cat f2 Varuvan vadivelan Institute of Technology Gundalpatti

CONCLUSION:

Thus the program to copy the contents of one file to another file was successfully executed and output was obtained.

Practice Exercises:

1. Write a C program on UNIX to count the number of characters, words and lines in a File.
2. Write a C program to create a file with integer values and open the file in read mode to read the content of the file then sort the content of the file in ascending order
3. Write a C program to store a set of integer values. Open the file in read mode and count the number of single, two and three digit values in a file
4. Write a C program to create two files with a set of values. Merge the two file content to form a single file
5. Write a C program to create a file, which contains employee id, name, age etc and display the contents.
6. Write a C program to perform the following file operations.
   (i) Create a file named "sample.txt".
   (ii) Read characters from keyboard and write to the file "sample.txt".
   (iii) Read the contents of a file and display it in the screen
7. Write a c program to merge the contents of one text file into another text file.
8. Write a C program on UNIX to display the file contents using command line arguments.
9. Write a C Program to Capitalize First Letter of every Word in a File
10. Write a C Program to Reverse the Contents of a File and Print it

VIVA QUESTIONS:

1. Discuss how a file is opened and closed?
2. What is file pointer?
3. Write a c program to create a text file
4. What is file positioning?
5. Distinguish between append mode and write mode.
6. How the contents of a file are is erased?
7. What is the meant by 'a' in the following operation?
   fp = fopen("Random.txt", "a");
8. Which type of files can't be opened using fopen()?
9. Mention the modes of file operations.
10. What is fseek ?

MULTIPLE CHOICE QUESTIONS:

1. Which of the following fopen statements are illegal?
   a) fp = fopen("abc.txt", "r"); b) fp = fopen("/home/user1/abc.txt", "w");

c) fp = fopen(“abc”, “w”); d) None of the mentioned

2. What does the following segment of code do?

fprintf(fp, “Copying!”);

a) It writes “Copying!” into the file pointed by fp

b) It reads “Copying!” from the file and prints on display

c) It writes as well as reads “Copying!” to and from the file and prints it

d) None of the mentioned

3. FILE reserved word is

a) A structure tag declared in stdio.h

b) One of the basic data types in c

c) Pointer to the structure defined in stdio.h

d) It is a type name defined in stdio.h

4. The file /proc version shows the version of _____ used in the system.

a) kernel b) gcc c) both (a) and (b) d) none of the mentioned

5. Which among the following is odd one out?

a) printf b) fprintf c) putchar d) scanf

6. For a typical program, the input is taken using.

a) scanf b) Files c) Command-line d) None of the mentioned

7. Which is true?

a) The symbolic constant EOF is defined in b) The value is typically -1,

c) Both a & b d) Either a or b

8. What happens when we use

fprintf(stderr, “error: could not open filen”);

a) The diagnostic output is directly displayed in the output.

b) The diagnostic output is pipelined to the output file.

c) The line which caused error is compiled again.

d) The program is immediately aborted.

9. What is the output of this C code?

```
#include <stdio.h>
int main()
{
   FILE *fp = stdout;
   stderr = fp;
   fprintf(stderr, "%s", "hello");
}
```

a) Compilation error b) hello c) Undefined behavior d) Depends on the standard

10. Which of the following statements about stdout and stderr are true?

a) Same b) Both connected to screen always.

c) Both connected to screen by default. d) stdout is line buffered but stderr is unbuffered.