TABLE OF CONTENTS

Lab	Experiment
Lab 1	Commands: • pwd • cd • cd~ • cd .. • ls • ls -1 • ls –t • ls –s • ls –r • ls –ltr • ls *.txt • mkdir directory_name • rmdir directory _name • vi file_name.txt • esc + i • esc + : + w • esc + : + q • esc + : + wq • cat file_name.txt • head file_name.txt • head –n5 file_name.txt • tail file_name.txt • tail –n5 file_name.txt • less file_name.txt
Lab 2	Commands: • cp file_name.txt file_name2.txt • cp –i file_name.txt file_name2.txt • mv file_name2.txt path • mv file_name.txt file_name2.txt • rm file_name.txt • rm -r directory_name/ file_name • sh shell_script_name.sh • ps • chmod[user/group/others/all] +/- permission file_name • top • kill pid
Lab 3	Commands: • wc file_name

	• wc –l file_name • wc –w file_name • wc –c file_name • command1 | command2 • command1 > file_name.txt • command1 < file_name.txt • who • more • sort • grep “string” • alias • chown • curl • finger • history • ping • uname • uname –a • uname –s • uname –r • uname –v • whoami • echo “string”
Lab 4	Commands: • bc • expr operand1 operator operand2 • read variable • ls ? • ls [characters]* • command1; command2 • cal • command1 > file_name.txt • command1 >> file_name.txt • command1 < file_name.txt • command1 & • pstree
	Write a shell script to define a variable x with value 10 and print it on the screen.
	Write a shell script to define a variable xn with value Rani and print it on the screen.
	Write a shell script to print the sum of two numbers.

	Write a shell script to define two variables x=20, y=5 and then print division of x and y (x/y).
	Write a shell script to define two variables x=20, y=5, store the division of x and y (x/y) in z and print it.
	Rectify the error in the following script.
	Write a shell script for addition of two numbers.
	Write a shell script for calculating the area of circle. Radius is to be entered by user.
	Write a shell script for swapping two number using a third variable.
	Write a shell script for swapping two number without using a third variable.
Lab 5	Write a shell script to check whether a number is odd or even.
	Write a shell script to find the smaller of the 2 numbers.
	Write a shell script to find the greatest of the 3 numbers.
Lab 6	Write a program to implement First Come, First Serve CPU Scheduling Algorithm.
Lab 7	Write a shell script to find factorial of a number.
	Write a shell script to print a Fibonacci series and find the sum of its terms.
	Write a shell script to implement a simple calculator.
	Write a command to display content of a file from line 5 to 10.
	Write a command to delete line with a specified word.
	Write a command to display list of files of directory.
Lab 8	Write a program to implement Shortest Job First CPU Scheduling Algorithm.
	Write a program to implement Shortest Remaining Time First CPU Scheduling Algorithm.
Lab 9	Write a program to implement Round Robin CPU Scheduling Algorithm.
	Write a program to implement Priority Scheduling CPU Scheduling Algorithm.
Lab 10	Write a program to implement Banker’s Algorithm.
Lab 11	Write a program to implement FIFO Page Replacement Algorithm.

LAB-1

1.) Command: pwd
Description: To find the current path
Implementation:

```
drishti@LAPTOP-082UILMC:~$ pwd
/home/drishti
```

2.) Command: cd
Description: To change a specific directory
Implementation:

```
drishti@LAPTOP-082UILMC:~$ cd dir1
drishti@LAPTOP-082UILMC:~/dir1$ pwd
/home/drishti/dir1
```

3.) Command: cd ~
Description: Location to home directory
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cd ~
drishti@LAPTOP-082UILMC:~$ pwd
/home/drishti
```

4.) Command: cd ..
Description: Location of directory below the current one
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cd ..
drishti@LAPTOP-082UILMC:~$ pwd
/home/drishti
```

5.) Command: ls
Description: To list the files in the current directory
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls
Sample.txt  Sample2.txt
```

6.) Command: ls -l
Description: To list the files in the current directory and display information
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -l
total 8
-rw-r--r-- 1 drishti drishti 12 Mar 23 20:11 Sample.txt
-rw-r--r-- 1 drishti drishti 28 Mar 23 20:11 Sample2.txt
```

7.) Command: ls -t

Description: To list the files in the current directory sorted by modification time
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -t
Sample2.txt  Sample.txt
```

8.) Command: ls -s
Description: To list the files in the current directory sorted by size
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -s
total 8
4 Sample.txt  4 Sample2.txt
```

9.) Command: ls -r
Description: To list the files in the current directory in reverse order
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -r
Sample2.txt  Sample.txt
```

10.) Command: ls -ltr
Description: To list the files in the current directory and displays information, sorted by time in reverse order
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -ltr
total 8
-rw-r--r-- 1 drishti drishti 12 Mar 23 20:11 Sample.txt
-rw-r--r-- 1 drishti drishti 28 Mar 23 20:11 Sample2.txt
```

11.) Command: ls *.txt
Description: * is used as a wildcard to list all text files
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls S*.txt
Sample.txt  Sample2.txt
```

12.) Command: mkdir directory_name
Description: To create a new directory
Implementation:

```
drishti@LAPTOP-082UILMC:~$ mkdir dir1
drishti@LAPTOP-082UILMC:~$ pwd
/home/drishti
```

13.) Command: rmdir directory_name
Description: To remove an empty directory
Implementation:

```
drishti@LAPTOP-082UILMC:~$ rmdir dirr
```

14.) Command: vi file_name.txt
Description: To create new files
Implementation:

```
drishti@LAPTOP-082UILMC:~$ vi Sample.txt
drishti@LAPTOP-082UILMC:~$ vi Sample2.txt
```

15.) Command: esc + i
Description: To insert text in the file created by vi command

16.) Command: esc + : + w
Description: To save the file created by vi command

17.) Command: esc + : + q
Description: To quit the file created by vi command

18.) Command: esc + : + wq
Description: To save and quit the file created by vi command

19.) Command: cat file_name.txt
Description: It displays the entire file to standard output
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cat Sample2.txt
this
is
a sample
text
file
```

19.) Command: head file_name.txt
Description: It displays the first 10 lines of the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ head Sample.txt
hello
world
lorem
a
b
c
d
e
f
g
```

20.) Command: head -n5 file_name.txt
Description: It displays the first 5 lines of the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ head -n5 Sample.txt
hello
world
lorem
a
b
```

21.) Command: tail file_name.txt
Description: It displays the last 10 lines of the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ tail Sample.txt
q
r
s
t
u
v
w
x
y
z
```

22.) Command: tail -n5 file_name.txt
Description: It displays the last 5 lines of the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ tail -n5 Sample.txt
v
w
x
y
z
```

23.) Command: less file_name.txt
Description: It displays the file to standard output allowing forward/ backward movement within it
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ less Sample.txt
```

```
hello
world
lorem
a
b
c
d
e
f
g
h
i
j
k
l
m
n
o
p
q
r
s
t
u
v
w
x
y
z
Sample.txt (END)
```

LAB-2

1.) Command: cp file_name.txt file_name2.txt
Description: Copies the content of first file to another
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cp Sample.txt Sample_copy.txt
drishti@LAPTOP-082UILMC:~/dir1$ head -n5 Sample_copy.txt
hello
world
lorem
a
b
```

2.)Command: cp -i file_name.txt file_name2.txt
Description: Copies the content of first file to another and gives a prompst
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cp -i Sample1.txt Samplee1.txt
drishti@LAPTOP-082UILMC:~/dir1$ cat Samplee1.txt
Hello
World
lorem
a
b
c
d
e
f
g
h
i
j
```

3.) Command: mv file_name2.txt path
Description: Moves the file to given location
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ mv Sample_copy.txt /home/drishti
drishti@LAPTOP-082UILMC:~/dir1$ cd ..
drishti@LAPTOP-082UILMC:~$ ls
Sample_copy.txt  ex8.shh  lab4  prog1.sh   sample1.sh       sample2.sh
```

4.) Command: mv file_name.txt file_name2.txt
Description: Renames the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ mv Sample_copy.txt Sample_copy_renamed.txt
drishti@LAPTOP-082UILMC:~/dir1$ ls
Sample.txt  Sample2.txt  Sample_copy_renamed.txt
```

5.) Command: rm file_name.txt
Description: Removes the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ rm Sample_copy_renamed.txt
drishti@LAPTOP-082UILMC:~/dir1$ ls
Sample.txt  Sample2.txt
```

6.) Command: rm -r directory_name/ file_name
Description: Remove the files recursively
Implementation:

```
drishti@LAPTOP-082UILMC:~$ rm -r dir1
drishti@LAPTOP-082UILMC:~$ ls
ex8.shh  lab1  lab4  lab5  prog1.sh  sample.sh  sample1.sh  sample1.sh.save  sample2.sh
```

7.) Command: sh shell_script_name.sh
Description: To execute a shell script
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ vi sample.sh
drishti@LAPTOP-082UILMC:~/dir1$ cat sample.sh

echo "Hello"
drishti@LAPTOP-082UILMC:~/dir1$ sh sample.sh
Hello
```

8.) Command: ps
Description: To view all the running processes
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ps
  PID TTY          TIME CMD
   10 pts/0    00:00:00 bash
   99 pts/0    00:00:00 ps
```

9.) Command: chmod[user/group/others/all] +/- permission file_name
Description: To change the permission(read(r)/ write(w)/ execute(x)) of a file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ ls -l
total 12
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample1.txt
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample2.txt
-rw-r--r-- 1 drishti drishti 14 Mar 23 20:32 sample.sh
drishti@LAPTOP-082UILMC:~/dir1$ chmod a+x Sample1.txt
drishti@LAPTOP-082UILMC:~/dir1$ ls -l
total 12
-rwxr-xr-x 1 drishti drishti  7 Mar 23 20:34 Sample1.txt
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample2.txt
-rw-r--r-- 1 drishti drishti 14 Mar 23 20:32 sample.sh
drishti@LAPTOP-082UILMC:~/dir1$ chmod a-x Sample1.txt
drishti@LAPTOP-082UILMC:~/dir1$ ls -l
total 12
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample1.txt
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample2.txt
-rw-r--r-- 1 drishti drishti 14 Mar 23 20:32 sample.sh
drishti@LAPTOP-082UILMC:~/dir1$ chmod u+rwx Sample2.txt
drishti@LAPTOP-082UILMC:~/dir1$ ls -l
total 12
-rw-r--r-- 1 drishti drishti  7 Mar 23 20:34 Sample1.txt
-rwxr--r-- 1 drishti drishti  7 Mar 23 20:34 Sample2.txt
-rw-r--r-- 1 drishti drishti 14 Mar 23 20:32 sample.sh
```

10.) Command: top
Description: To view the CPU usage of all the processes
Implementation:

```
top - 20:38:13 up 32 min,  0 users,  load average: 0.00, 0.00, 0.00
Tasks:   5 total,   1 running,   4 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.0 us,  0.0 sy,  0.0 ni,100.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  12601.7 total,  12473.6 free,     90.0 used,     38.1 buff/cache
MiB Swap:   4096.0 total,   4096.0 free,      0.0 used.  12345.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    1 root      20   0     896    528    464 S   0.0   0.0   0:00.03 init
    8 root      20   0     896     84     20 S   0.0   0.0   0:00.00 init
    9 root      20   0     896     84     20 S   0.0   0.0   0:00.64 init
   10 drishti   20   0   10144   5040   3316 S   0.0   0.0   0:00.37 bash
  100 drishti   20   0   10884   3740   3184 R   0.0   0.0   0:00.00 top
```

11.) Command: kill pid
Description: Terminates the process

LAB-3

1.) Command: wc file_name
Description: Counts lines, words and characters in the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ cat Sample1.txt
Hello
World
lorem
a
b
c
d
e
f
g
h
i
j

drishti@LAPTOP-082UILMC:~/dir1$ wc Sample1.txt
14 13 39 Sample1.txt
```

2.) Command: wc -l file_name
Description: Counts number of lines in the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ wc -l Sample1.txt
14 Sample1.txt
```

3.) Command: wc -w file_name
Description: Counts number of words in the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ wc -w Sample1.txt
13 Sample1.txt
```

4.) Command: wc -c file_name
Description: Counts number of characters in the file
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ wc -c Sample1.txt
39 Sample1.txt
```

5.) Command: command1 | command2
Description: Pipe is used to give the output of first command as the input to the second
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ grep "Hello" Sample1.txt | wc -l
1
```

6.) Command: command1 > file_name.txt
Description: Output of command1 is written to the file (Redirection)
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ wc -l Sample1.txt > newSample.txt
drishti@LAPTOP-082UILMC:~/dir1$ cat newSample.txt
14 Sample1.txt
```

7.) Command: command1 < file_name.txt
Description: Input for command1 is taken from the file (Redirection)
Implementation:

```
drishti@LAPTOP-082UILMC:~$ wc -c < Sample.txt
12
```

8.) Command: who
Description: Displays the list of users
Implementation:

```
aib@pamoli1-virtual-machine:~/A2305220108$ man who
aib@pamoli1-virtual-machine:~/A2305220108$ aib@pamoli1-virtual-machine:~/A2305220108$
aib@pamoli1-virtual-machine:~/A2305220108$ who
aib      pts/1        2022-01-19 10:06 (106.201.210.143)
pamoli1  :0           2021-10-21 11:18 (:0)
aib      pts/2        2022-01-19 09:27 (49.36.185.223)
aib      pts/3        2022-01-19 09:35 (103.226.226.32)
aib      pts/4        2022-01-19 09:54 (106.215.70.186)
aib      pts/5        2022-01-19 09:39 (157.36.183.15)
aib      pts/7        2022-01-19 09:39 (49.36.215.185)
aib      pts/8        2022-01-19 09:39 (106.223.12.49)
aib      pts/9        2022-01-19 09:39 (49.36.218.191)
aib      pts/10       2022-01-19 09:39 (45.118.158.44)
aib      pts/11       2022-01-19 09:39 (47.9.82.100)
aib      pts/12       2022-01-19 09:39 (106.215.70.186)
aib      pts/13       2022-01-19 09:40 (103.39.116.209)
aib      pts/14       2022-01-19 09:46 (103.157.221.84)
aib      pts/15       2022-01-19 09:43 (106.201.210.143)
aib      pts/16       2022-01-19 09:42 (125.99.4.159)
aib      pts/17       2022-01-19 09:43 (45.249.84.21)
aib      pts/18       2022-01-19 09:44 (157.36.181.208)
aib      pts/19       2022-01-19 09:45 (103.226.226.32)
aib      pts/20       2022-01-19 09:47 (103.70.82.118)
aib      pts/21       2022-01-19 10:06 (112.196.174.100)
aib      pts/22       2022-01-19 09:50 (49.36.215.185)
aib      pts/23       2022-01-19 09:51 (125.99.4.159)
aib      pts/24       2022-01-19 09:54 (106.201.210.143)
aib      pts/25       2022-01-19 10:14 (125.99.4.159)
aib      pts/26       2022-01-19 09:52 (112.196.147.26)
aib      pts/27       2022-01-19 10:08 (49.36.218.191)
aib      pts/28       2022-01-19 09:54 (45.249.84.21)
aib      pts/29       2022-01-19 10:28 (103.39.116.209)
aib      pts/30       2022-01-19 10:07 (47.9.82.100)
aib      pts/31       2022-01-19 10:00 (223.233.64.39)
aib      pts/32       2022-01-19 10:12 (157.36.181.208)
aib      pts/33       2022-01-19 10:08 (103.157.221.84)
aib7     :1           2021-12-10 12:48 (:1)
aib      pts/34       2022-01-19 10:09 (49.36.215.185)
aib      pts/35       2022-01-19 10:15 (45.118.158.44)
aib      pts/36       2022-01-19 10:12 (106.201.210.143)
aib      pts/37       2022-01-19 10:17 (45.249.84.21)
aib      pts/39       2022-01-19 10:20 (49.36.218.191)
aib      pts/40       2022-01-19 10:21 (103.70.82.118)
aib      pts/41       2022-01-19 10:23 (112.196.147.26)
aib      pts/42       2022-01-19 10:24 (106.201.210.143)
```

9.) Command: more
Description: Displays the information one screen full page at a time
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ more Sample1.txt
Hello
World
lorem
a
b
c
d
e
f
g
h
i
j
```

11.) Command: sort
Description: Sorts the input alphabetically/ numerically
Implementation:

```
drishti@LAPTOP-082UILMC:~$ nano SortSample.txt
drishti@LAPTOP-082UILMC:~$ cat SortSample.txt
G
G
WD
F
H
H
JK
K
S
DD
F
FG
DF
2
4
6
```

```
drishti@LAPTOP-082UILMC:~$ sort SortSample.txt

2
4
6
DD
DF
F
F
FG
G
G
H
H
JK
K
S
WD
```

12.) Command: grep “string”
Description: To search for a string in a file/ dirctory
Implementation:

```
drishti@LAPTOP-082UILMC:~/dir1$ grep "World" Sample1.txt
World
```

13.) Command: alias
Description: Lets you give an alternative name to the Linux commands
Implementation:

```
aib@pamoli1-virtual-machine:~/A2305220108$ alias list=ls
aib@pamoli1-virtual-machine:~/A2305220108$ list
dirr  lab3.txt    new_dir          Sample2.txt  Samplee       Test.sh
fds   myfile.txt  Sample2copy.txt  Sample3.txt  Samplee1.txt  who.txt
aib@pamoli1-virtual-machine:~/A2305220108$ alias find=grep
aib@pamoli1-virtual-machine:~/A2305220108$ find "abc" lab3.txt
abc
abcd
```

14.) Command: chown
Description: Allows to change the owner and group owner of a file

15.) Command: curl
Description: Retrieves information and files from URL

16.) Command: finger
Description: Gives information about a user (time of user’s login, user’s home directory, user’s account full name etc.)

17.) Command: history
Description: Lists the commands previously used on the command line
Implementation:

```
drishti@LAPTOP-082UILMC:~$ history
    1  pwd
    2  x=10
    3  echo x
    4  echo$x
    5  x=10
    6  echo $x
    7  clear
    8  sh sample.sh
    9  mkdir lab4
   10  sh sample.sh
   11  vi prog1.sh
   12  sh prog1.sh
```

18.) Command: ping
Description: Verifies whether you have network connectivity with another network device

19.) Command: uname
Description: To obtain system information regarding Linux server
Implementation:

```
drishti@LAPTOP-082UILMC:~$ uname
Linux
```

20.) Command: uname -a
Description: To obtain system information regarding Linux server (all information)
Implementation:

```
drishti@LAPTOP-082UILMC:~$ uname -a
Linux LAPTOP-082UILMC 5.10.16.3-microsoft-standard-WSL2 #1 SMP Fri Apr 2 22:23:49 UTC 2021 x86_64 x86_64
 x86_64 GNU/Linux
```

21.) Command: uname -s
Description: To obtain kernel name
Implementation:

```
drishti@LAPTOP-082UILMC:~$ uname -s
Linux
```

22.) Command: uname -r
Description: To obtain kernel release
Implementation:

```
drishti@LAPTOP-082UILMC:~$ uname -r
5.10.16.3-microsoft-standard-WSL2
```

23.) Command: uname -v
Description: To obtain kernel release version
Implementation:

```
drishti@LAPTOP-082UILMC:~$ uname -v
#1 SMP Fri Apr 2 22:23:49 UTC 2021
```

24.) Command: whoami
Description: To find out who you are logged in as
Implementation:

```
drishti@LAPTOP-082UILMC:~$ whoami
drishti
```

25.) Command: echo “string”
Description: To print the given string
Implementation:

```
drishti@LAPTOP-082UILMC:~$ echo "Drishti"
Drishti
```

LAB-4

1.) Command: bc
Description: To work as a basic calculator
Implementation:

```
drishti@LAPTOP-082UILMC:~$ echo "12 + 5" | bc
17
```

2.) Command: expr operand1 operator operand2
Description: Used to perform arithmetic operations
Implementation:

```
drishti@LAPTOP-082UILMC:~$ expr 12 \* 5
60
```

3.) Command: read variable
Description: To get input from the user from keyboard and store the value in the variable
Implementation:

```
drishti@LAPTOP-082UILMC:~$ read num
4
drishti@LAPTOP-082UILMC:~$ echo $num
4
```

4.) Command: ls ?
Description: Used as wildcard which can be replaced by a single character to display the list of files
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls lab?
lab1:

lab4:
ex1.sh  ex10.sh  ex2.sh  ex3.sh  ex4.sh  ex5.sh  ex6.sh  ex7.sh  ex8.sh  ex9.sh  sampleshell.sh

lab5:
ex1.sh  ex1.sh.save

lab7:
Sample.txt  ex1.sh  ex2.sh  ex3.sh
```

5.) Command: ls [characters]*
Description: Used as wildcard which can be replaced by the given characters to display the list of files
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls [m-q]*
prog1.sh  program1  program1.c
```

6.) Command: command1; command2
Description: To run 2 commands with one command line
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls;cal
Sample.txt  ex8.shh  lab4  lab7      program1    sample.sh   sample1.sh.save
dir1        lab1     lab5  prog1.sh  program1.c  sample1.sh  sample2.sh
     April 2022
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

7.) Command: cal
Description: Displays the calender
Implementation:

```
drishti@LAPTOP-082UILMC:~$ cal
     April 2022
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

8.) Command: command1 > file_name.txt
Description: Output of command1 is written to the file (Redirection)
Implementation:

```
drishti@LAPTOP-082UILMC:~$ date > Sample.txt
drishti@LAPTOP-082UILMC:~$ cat Sample.txt
Sat Apr 16 01:00:21 IST 2022
```

9.) Command: command1 >> file_name.txt
Description: Output of command1 is appended at the end of the file (Redirection)
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls >> Sample.txt
drishti@LAPTOP-082UILMC:~$ head Sample.txt
4
Sample.txt
dir1
ex8.shh
lab1
lab4
lab5
lab7
prog1.sh
program1
```

10.) Command: command1 < file_name.txt
Description: Input for command1 is taken from the file (Redirection)
Implementation:

```
drishti@LAPTOP-082UILMC:~$ wc -c < Sample.txt
12
```

11.) Command: command1 &
Description: & at the end of the command tells the shell to run it in background and start the execution of the next command
Implementation:

```
drishti@LAPTOP-082UILMC:~$ ls &
[1] 120
drishti@LAPTOP-082UILMC:~$ Sample.txt  ex8.shh  lab4  lab7      program1    sample.sh   sample1.sh.save
dir1        lab1     lab5  prog1.sh  program1.c  sample1.sh  sample2.sh
```

12.) Command: pstree
Description: Displays a tree of processes
Implementation:

```
drishti@LAPTOP-082UILMC:~$ pstree
init─┬─init───init───bash───pstree
     └─{init}
```

PROGRAM-1

AIM

Write a shell script to define a variable x with value 10 and print it on the screen.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$  cat ex1.sh
# Define a variable x with value 10 and print it on screen
x=10
echo $x
drishti@LAPTOP-082UILMC:~/lab4$ sh ex1.sh
10
```

PROGRAM-2

AIM

Write a shell script to define a variable xn with value Rani and print it on the screen.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex2.sh
# Define variabel xn with Rani and print it on screen
xn=Rani
echo Value of xn is $xn
drishti@LAPTOP-082UILMC:~/lab4$ sh ex2.sh
Value of xn is Rani
```

PROGRAM-3

AIM

Write a shell script to print the sum of two numbers.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex3.sh
# Print sum of two numbers
echo `expr 6 + 3`
drishti@LAPTOP-082UILMC:~/lab4$ sh ex3.sh
9
```

PROGRAM-4

AIM

Write a shell script to define two variables x=20, y=5 and then print division of x and y (x/y).

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex4.sh
# Define two variables x=20, y=5 and then to print division of x and y
x=20
y=5
echo `expr $x / $y`
drishti@LAPTOP-082UILMC:~/lab4$ sh ex4.sh
4
```

PROGRAM-5

AIM

Write a shell script to define two variables x=20, y=5, store the division of x and y (x/y) in z and print it.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex5.sh
#Store divisioon of x and y to variable called z
x=20
y=5
z=`expr $x / $y`
echo $z
drishti@LAPTOP-082UILMC:~/lab4$ sh ex5.sh
4
```

PROGRAM-6

AIM

Rectify the error in the following script.

```
# Script to test MY knowledge about variables!
#
myname=Vivek
myos = TroubleOS
myno=5
echo "My name is $myname"
echo "My os is $myos"
echo "My number is myno, can you see this number"
```

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex6.sh
myname=Vivek
myos=TroubleOS
myno=5
echo My name is $myname
echo My os is $myos
echo My number is $myno
drishti@LAPTOP-082UILMC:~/lab4$ sh ex6.sh
My name is Vivek
My os is TroubleOS
My number is 5
```

PROGRAM-7

AIM

Write a shell script for addition of two numbers.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex7.sh
# Addition of two numbers
echo Enter two numbers
read num1
read num2
echo Sum is `expr $num1 + $num2`
drishti@LAPTOP-082UILMC:~/lab4$ sh ex7.sh
Enter two numbers
4
5
Sum is 9
```

PROGRAM-8

AIM

Write a shell script for calculating the area of circle. Radius is to be entered by user.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex8.sh
# Calculate area of circle
echo Enter radius
read rad
pi=3.14
area=$(echo "$pi * $rad * $rad"|bc)
echo Area of circle of radius $rad is $area
drishti@LAPTOP-082UILMC:~/lab4$ sh ex8.sh
Enter radius
7
Area of circle of radius 7 is 153.86
```

PROGRAM-9

AIM

Write a shell script for swapping two number using a third variable.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex9.sh
# Script for swapping two numbers using third variable
echo Enter two numbers
read num1
read num2
echo Numbers before swapping: $num1, $num2
temp=$num1
num1=$num2
num2=$temp
echo Numbers after swapping: $num1, $num2
drishti@LAPTOP-082UILMC:~/lab4$ sh ex9.sh
Enter two numbers
4
8
Numbers before swapping: 4, 8
Numbers after swapping: 8, 4
```

PROGRAM-10

AIM

Write a shell script for swapping two number without using a third variable.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab4$ cat ex10.sh
# Script fro swapping two numbers without using third variables
echo Enter two numbers
read num1
read num2
echo Numbers before swapping: $num1, $num2
num1=`expr $num1 + $num2`
num2=`expr $num1 - $num2`
num1=`expr $num1 - $num2`
echo Numbers after swapping: $num1, $num2
drishti@LAPTOP-082UILMC:~/lab4$ sh ex10.sh
Enter two numbers
4
8
Numbers before swapping: 4, 8
Numbers after swapping: 8, 4
```

LAB-5

PROGRAM-1

AIM

Write a shell script to check whether a number is odd or even.

SHELL SCRIPT

main.bash

```bash
# Program to check whether a number is odd or even

echo "Enter a number"
read n

if [ `expr $n % 2` -eq 0 ]
then
echo "$n is even"
else
echo "$n is Odd"
fi
```

OUTPUT

```
Enter a number
2
2 is even


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-2

AIM

Write a shell script to find the smaller of the 2 numbers.

SHELL SCRIPT

main.bash

```bash
# Program to check smaller of 2 numbers

echo "Enter the 2 number :"
read n1
read n2

if [ $n1 -gt $n2 ]
then
    echo "$n2 is smaller than $n1"
else
    echo "$n1 is smaller than $n2"
fi
```

OUTPUT

```
input
Enter the 2 number :
4
7
4 is smaller than 7


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-3

AIM

Write a shell script to find the greatest of the 3 numbers.

SHELL SCRIPT

main.bash

```bash
# Program to check largest of 3 numbers

echo "Enter 3 numbers :"
read n1
read n2
read n3

if [ $n1 -gt $n2 -a $n1 -gt $n3 ]
then
    echo -n "$n1 is largest"
elif [ $n2 -gt $n1 -a $n2 -gt $n3 ]
then
    echo -n "$n2 is largest "
elif [ $n3 -gt $n1 -a $n3 -gt $n2 ]
then
    echo -n "$n3 is largest "
fi
```

OUTPUT

```
input
Enter 3 numbers :
2
9
5
9 is largest

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB-6

PROGRAM-1

AIM

Write a program to implement First Come, First Serve CPU Scheduling Algorithm.

THEORY

First Come First Serve process scheduling algorithm executes the processes in the order of their arrival in the ready queue. The process which arrives first is executed first.

SOURCE CODE

```
arrival_time_list = []
burst_time_list = []
top = 0

n = int(input("Enter number of processes: "))
for i in range(n):
    pat = int(input("Enter P%d Arrival Time: " % (i)))
    arrival_time_list.append(pat)

    pbt = int(input("Enter P%d Burst Time: " % (i)))
    burst_time_list.append(pbt)

finish_time_list = []
turn_around_time_list = []
waiting_time_list = []

for i in range(n):
    if i == 0:
        top = top + burst_time_list[i]
        finish_time_list.append(top)
    elif i > 0:
        if top < arrival_time_list[i]:
            top = arrival_time_list[i] + burst_time_list[i]
            finish_time_list.append(top)
        else:
            top = top + burst_time_list[i]
            finish_time_list.append(top)

for i in range(n):
    tat = finish_time_list[i] - arrival_time_list[i]
    turn_around_time_list.append(tat)
    wt = turn_around_time_list[i]-burst_time_list[i]
    waiting_time_list.append(wt)

av_tat = 0.0
av_wt = 0.0
```

```
print("\tProcess\t", "\tArrival Time\t", "\t Burst Time\t", "\tFinish Time\t", "\tTurn Around Time\t",
"\tWaiting Time\t")

for i in range(n):
    print("\t" + str(i) + "\t\t\t" + str(arrival_time_list[i]) + "\t\t\t" + str(burst_time_list[i]) + "\t\t\t" +
str(finish_time_list[i]) +
     "\t\t\t"+ str(turn_around_time_list[i]) + "\t\t\t  " +str(waiting_time_list[i]))

    av_tat = av_tat + turn_around_time_list[i]
    av_wt = av_wt + waiting_time_list[i]

av_tat=float(av_tat)/n
av_wt = av_wt/n

print("Average Turn Around Time: %f" % av_tat)
print("Average Waiting Time: %f" % av_wt)
```

OUTPUT

```
Enter number of processes: 4
Enter P0 Arrival Time: 0
Enter P0 Burst Time: 10
Enter P1 Arrival Time: 1
Enter P1 Burst Time: 6
Enter P2 Arrival Time: 3
Enter P2 Burst Time: 2
Enter P3 Arrival Time: 5
Enter P3 Burst Time: 4
        Process         Arrival Time            Burst Time            Finish Time            Turn Around Time            Waiting Time
        0                       0                       10                      10                      10                       0
        1                       1                       6                       16                      15                       9
        2                       3                       2                       18                      15                       13
        3                       5                       4                       22                      17                       13
Average Turn Around Time: 14.250000
Average Waiting Time: 8.750000
```

LAB-7

PROGRAM-1

AIM

Write a shell script to find factorial of a number.

SHELL SCRIPT

main.bash

```bash
# Shell script to find factorial of a number
echo "Enter number"
read n
fact=1
for (( i=1; i<=n; i++ ))
do
    fact=$(( fact * i ))
done
echo "Factorial of $n is $fact"
```

OUTPUT

```
Enter number
5
Factorial of 5 is 120


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-2

AIM

Write a shell script to print a Fibonacci series and find the sum of its terms.

SHELL SCRIPT

main.bash

```bash
# Shell script to print fibonacci series and find the sum of its terms
echo "Enter number of terms"
read terms
n1=0
n2=1
sum=1
echo "Fibonacci Series: "
echo "0"
echo "1"
for (( i=2; i<terms; i++ ))
do
    n3=$(( n1 + n2 ))
    echo "$n3 "
    sum=$(( sum + n3 ))
    n1=$n2
    n2=$n3
    done
echo "Sum is $sum"
```

OUTPUT

```
Enter number of terms
5
Fibonacci Series:
0
1
1
2
3
Sum is 7


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-3

AIM

Write a shell script to implement a simple calculator.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab7$ cat ex1.sh
# Shell script to implement simple calculator
echo "Enter operand 1"
read op1
echo "Enter operand 2"
read op2
echo "Choose an operation"
echo "Enter add for Addition"
echo "Enter sub for Subtraction"
echo "Enter mul for Multiplication"
echo "Enter div for Division"
read choice
case $choice in
        "add") res=`echo $op1 + $op2 | bc`
        ;;
        "sub") res=`echo $op1 - $op2 | bc`
        ;;
        "mul") res=`echo $op1 \* $op2 | bc`
        ;;
        "div") res=`echo $op1 / $op2 | bc`
        ;;
        *) echo "Enter another choice"
        ;;
esac
echo "Answer = $res"
drishti@LAPTOP-082UILMC:~/lab7$ sh ex1.sh
Enter operand 1
5
Enter operand 2
4
Choose an operation
Enter add for Addition
Enter sub for Subtraction
Enter mul for Multiplication
Enter div for Division
mul
Answer = 20
```

PROGRAM-4

AIM

Write a command to display content of a file from line 5 to 10.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab7$ nano Sample.txt
drishti@LAPTOP-082UILMC:~/lab7$ sed -n '5,10p' Sample.txt
 a
 programmer
I have
learnt
Python
C
```

PROGRAM-5

AIM

Write a command to delete line with a specified word.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~/lab7$ grep -v "learnt" Sample.txt > Sample2.txt && mv Sample2.txt Sample.txt
drishti@LAPTOP-082UILMC:~/lab7$ cat Sample.txt
Hello
world
I
am
 a
 programmer
I have
Python
C
CPP
Java
```

PROGRAM-6

AIM

Write a command to display list of files of directory.

SHELL SCRIPT

```
drishti@LAPTOP-082UILMC:~$ ls
dir1  ex8.shh  lab1  lab4  lab5  lab7  prog1.sh  sample.sh  sample1.sh  sample1.sh.save  sample2.sh
```

LAB-8

PROGRAM-1

AIM
Write a program to implement Shortest Job First CPU Scheduling Algorithm.

THEORY
The processes having the least burst times are executed first. This is a non-preemptive scheduling algorithm.

LANGUAGE USED: C

SOURCE CODE

```
#include <stdio.h>

int main()
{
    float bt[1000]={0},at[1000]={0};
    float tat[1000]={0},wt[1000]={0};
    int n;

    printf("Enter the number of processes: ");
    scanf("%d",&n);

    for(int i=0;i<n;i++){
      printf("Enter the burst time %d: ",i+1);
      scanf("%f",&bt[i]);
      if(i>=1)
      bt[i]=bt[i-1]+bt[i];
    }

    for(int i=0;i<n;i++){
      printf("Enter the arrival time %d: ",i+1);
      scanf("%f",&at[i]);
    }

    for(int i=0;i<n;i++){
      tat[i]=bt[i]-at[i];
      tat[n]+=tat[i];
    }
```

```
    for(int i=0;i<n;i++){
      wt[i]=tat[i]-bt[i]+bt[i-1];
      wt[n]+=wt[i];
    }


    printf("Process no.\tBurst time\tArrival time\tTurn around time\t Waiting time\n");

    for(int i=0;i<n;i++){
      printf("%d\t\t %f\t\t %f\t\t%f\t\t%f\n",i+1,bt[i]-bt[i-1],at[i],tat[i],wt[i]);
    }

    printf("Average waiting time: %f\n",wt[n]/n);
    printf("Average turn around time: %f",tat[n]/n);

return 0;
}
```

OUTPUT

```
Enter the number of processes: 3
Enter the burst time 1: 24
Enter the burst time 2: 3
Enter the burst time 3: 4
Enter the arrival time 1: 0
Enter the arrival time 2: 0
Enter the arrival time 3: 0
Process no.     Burst time      Arrival time    Turn around time        Waiting time
1                24.000000               0.000000                24.000000               0.000000
2                3.000000                0.000000                27.000000               24.000000
3                4.000000                0.000000                31.000000               27.000000
Average waiting time: 17.000000
Average turn around time: 27.333334

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-2

AIM

Write a program to implement Shortest Remaining Time First CPU Scheduling Algorithm.

THEORY

After the arrival of each process an interrupt is sent to the CPU after which the time left for completion of each process of the ready queue and also the current ongoing process is compared and the process with the least time left is executed. This is a preemptive type of scheduling.

LANGUAGE USED: C

SOURCE CODE

```
#include<stdio.h>

int main()
{
int i,n,p[10]={1,2,3,4,5,6,7,8,9,10},min,k=1,btime=0,temp,j;
float bt[10],at[10],wt[10],tt[10],ta=0,sum=0;
float wavg=0,tavg=0,tsum=0,wsum=0;

printf("\nEnter the No. of processes :");
scanf("%d",&n);

for(i=0;i<n;i++)
{
printf("\tEnter the burst time of %d process :",i+1);
scanf(" %f",&bt[i]);
printf("\tEnter the arrival time of %d process :",i+1);
scanf(" %f",&at[i]);
}

for(i=0;i<n;i++)
  {
    for(j=0;j<n;j++)
     {
       if(at[i]<at[j])
        {
          temp=p[j];
          p[j]=p[i];
```

```
            p[i]=temp;
            temp=at[j];
            at[j]=at[i];
            at[i]=temp;
            temp=bt[j];
            bt[j]=bt[i];
            bt[i]=temp;
            }
        }
    }

for(j=0;j<n;j++)
  {
    btime=btime+bt[j];
    min=bt[k];
    for(i=k;i<n;i++)
     {
       if (btime>=at[i] && bt[i]<min)
        {
           temp=p[k];
           p[k]=p[i];
           p[i]=temp;
           temp=at[k];
           at[k]=at[i];
           at[i]=temp;
           temp=bt[k];
           bt[k]=bt[i];
           bt[i]=temp;
        }
     }
    k++;
  }

wt[0]=0;

for(i=1;i<n;i++)
  {
    sum=sum+bt[i-1];
    wt[i]=sum-at[i];
    wsum=wsum+wt[i];
  }
```

```
wavg=(wsum/n);

for(i=0;i<n;i++)
  {
    ta=ta+bt[i];
    tt[i]=ta-at[i];
    tsum=tsum+tt[i];
   }

tavg=(tsum/n);

printf("\nProcess\t Burst\t Arrival\t Waiting\t Turn-around" );
for(i=0;i<n;i++)
{
printf("\n p%d\t %f\t %f\t\t %f\t\t\t%f",p[i],bt[i],at[i],wt[i],tt[i]);
}

printf("\n\nAVERAGE WAITING TIME : %f",wavg);
printf("\nAVERAGE TURN AROUND TIME : %f",tavg);
return 0;
}
```

OUTPUT

```
Enter the No. of processes :4
        Enter the burst time of 1 process :7
        Enter the arrival time of 1 process :0
        Enter the burst time of 2 process :4
        Enter the arrival time of 2 process :2
        Enter the burst time of 3 process :1
        Enter the arrival time of 3 process :4
        Enter the burst time of 4 process :4
        Enter the arrival time of 4 process :5

Process  Burst   Arrival         Waiting         Turn-around
 p1      7.000000        0.000000                0.000000                        7.000000
 p3      1.000000        4.000000                3.000000                        4.000000
 p2      4.000000        2.000000                6.000000                        10.000000
 p4      4.000000        5.000000                7.000000                        11.000000

AVERAGE WAITING TIME : 4.000000
AVERAGE TURN AROUND TIME : 8.000000

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB-9

PROGRAM-1

AIM
Write a program to implement Round Robin CPU Scheduling Algorithm.

THEORY
Each process is executed for a certain fixed time quantum and the execution of the processes is carried out in a cyclic manner. No process remains untouched in this scheduling algorithm. This a preemptive scheduling algorithm.

LANGUAGE USED: C

SOURCE CODE

```
#include<stdio.h>
#include<conio.h>

void main()
{

   int i, NOP, sum=0,count=0, y, quant, wt=0, tat=0, at[10], bt[10], temp[10];
   float avg_wt, avg_tat;
   printf(" Total number of process in the system: ");
   scanf("%d", &NOP);
   y = NOP;

for(i=0; i<NOP; i++)
{
printf("\n Enter the Arrival and Burst time of the Process[%d]\n", i+1);
printf(" Arrival time is: \t");
scanf("%d", &at[i]);
printf(" \nBurst time is: \t");
scanf("%d", &bt[i]);
temp[i] = bt[i];
}

printf("Enter the Time Quantum for the process: \t");
scanf("%d", &quant);
```

```
printf("\n Process No \t\t Burst Time \t\t TAT \t\t Waiting Time ");
for(sum=0, i = 0; y!=0; )
{
if(temp[i] <= quant && temp[i] > 0)
{
   sum = sum + temp[i];
   temp[i] = 0;
   count=1;
   }
   else if(temp[i] > 0)
   {
      temp[i] = temp[i] - quant;
      sum = sum + quant;
   }
   if(temp[i]==0 && count==1)
   {
      y--;
      printf("\nProcess No[%d] \t %d\t\t\t %d\t\t %d", i+1, bt[i], sum-at[i], sum-at[i]-bt[i]);
      wt = wt+sum-at[i]-bt[i];
      tat = tat+sum-at[i];
      count =0;
   }
   if(i==NOP-1)
   {
      i=0;
   }
   else if(at[i+1]<=sum)
   {
      i++;
   }
   else
   {
      i=0;
   }
}

avg_wt = wt * 1.0/NOP;
avg_tat = tat * 1.0/NOP;
printf("\n Average Turn Around Time: \t%f", avg_wt);
printf("\n Average Waiting Time: \t%f", avg_tat);
getch();
```

}

OUTPUT

```
input
 Total number of process in the system: 3

 Enter the Arrival and Burst time of the Process[1]
 Arrival time is:       0

Burst time is:  3

 Enter the Arrival and Burst time of the Process[2]
 Arrival time is:       0

Burst time is:  4

 Enter the Arrival and Burst time of the Process[3]
 Arrival time is:       0

Burst time is:  3
Enter the Time Quantum for the process:        1

 Process No              Burst Time              TAT              Waiting Time
Process No[1]    3                      7              4
Process No[3]    3                      9              6
Process No[2]    4                      10             6
 Average Turn Around Time:      5.333333
 Average Waiting Time:  8.666667

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM-2

AIM

Write a program to implement Priority Scheduling CPU Scheduling Algorithm.

THEORY

The process with higher priority is executed first. This scheduling can be preemptive or non-preemptive.

LANGUAGE USED: C

SOURCE CODE

```
#include<stdio.h>
struct process
{
   int WT,AT,BT,TAT,PT;
};

struct process a[10];

int main()
{
   int n,temp[10],t,count=0,short_p;
   float total_WT=0,total_TAT=0,Avg_WT,Avg_TAT;
   printf("Enter the number of the process\n");
   scanf("%d",&n);
   printf("Enter the arrival time , burst time and priority of the process\n");
   printf("AT BT PT\n");
   for(int i=0;i<n;i++)
   {
      scanf("%d%d%d",&a[i].AT,&a[i].BT,&a[i].PT);

      temp[i]=a[i].BT;
   }

   a[9].PT=10000;

   for(t=0;count!=n;t++)
   {
      short_p=9;
      for(int i=0;i<n;i++)
```

```
        {
            if(a[short_p].PT>a[i].PT && a[i].AT<=t && a[i].BT>0)
            {
                short_p=i;
            }
        }

        a[short_p].BT=a[short_p].BT-1;

        if(a[short_p].BT==0)
        {

            count++;
            a[short_p].WT=t+1-a[short_p].AT-temp[short_p];
            a[short_p].TAT=t+1-a[short_p].AT;
            total_WT=total_WT+a[short_p].WT;
            total_TAT=total_TAT+a[short_p].TAT;

        }
    }
    Avg_WT=total_WT/n;
    Avg_TAT=total_TAT/n;

    printf("ID WT TAT\n");
    for(int i=0;i<n;i++)
    {
        printf("%d %d\t%d\n",i+1,a[i].WT,a[i].TAT);
    }
    printf("Avg waiting time of the process  is %f\n",Avg_WT);
    printf("Avg turn around time of the process is %f\n",Avg_TAT);
    return 0;
}
```

OUTPUT

```
Enter the number of the process
4
Enter the arrival time , burst time and priority of the process
AT BT PT
0 6 4
0 8 1
0 7 3
0 3 2
ID WT TAT
1 18    24
2 0     8
3 11    18
4 8     11
Avg waiting time of the process  is 9.250000
Avg turn around time of the process is 15.250000


...Program finished with exit code 0
Press ENTER to exit console.
```

LAB-10

PROGRAM-1

AIM
Write a program to implement Banker's Algorithm.

THEORY
Banker's algorithm is a deadlock avoidance algorithm for resources with more than one instance. It utilizes the static information about the allocation and availability of the resources along with the maximum need of all the processes. If at any instance, the need of the processes exceeds the availability of the resources, the execution of the process becomes stunted, there is absence of a safe sequence of execution of the processes and thus the state is said to be unsafe and deadlock is vulnerable to occur.

LANGUAGE USED: C

SOURCE CODE

```
#include <stdio.h>
int curr[10][10], maxclaim[10][10], avl[10];
int alloc[10] = {0};
int maxres[10], running[10], safe=0;
int count = 0, i, j, exec, r, p, k = 1;

int main()
{
   printf("\nEnter the number of processes: ");
   scanf("%d", &p);

   for (i = 0; i < p; i++) {
      running[i] = 1;
      count++;
   }

   printf("\nEnter the number of resources: ");
   scanf("%d", &r);

   for (i = 0; i < r; i++) {
      printf("\nEnter the resource for instance %d: ", k++);
      scanf("%d", &maxres[i]);
   }
```

```
printf("\nEnter maximum resource table:\n");
for (i = 0; i < p; i++) {
   for(j = 0; j < r; j++) {
      scanf("%d", &maxclaim[i][j]);
   }
}

printf("\nEnter allocated resource table:\n");
for (i = 0; i < p; i++) {
   for(j = 0; j < r; j++) {
      scanf("%d", &curr[i][j]);
   }
}

printf("\nThe resource of instances: ");
for (i = 0; i < r; i++) {
   printf("\t%d", maxres[i]);
}
printf("\nThe allocated resource table:\n");
for (i = 0; i < p; i++) {
   for (j = 0; j < r; j++) {
      printf("\t%d", curr[i][j]);
   }
   printf("\n");
}

printf("\nThe maximum resource table:\n");
for (i = 0; i < p; i++) {
   for (j = 0; j < r; j++) {
      printf("\t%d", maxclaim[i][j]);
   }
   printf("\n");
}

for (i = 0; i < p; i++) {
   for (j = 0; j < r; j++) {
      alloc[j] += curr[i][j];
   }
}

printf("\nAllocated resources:");
```

```
for (i = 0; i < r; i++) {
    printf("\t%d", alloc[i]);
}

for (i = 0; i < r; i++) {
    avl[i] = maxres[i] - alloc[i];
}

printf("\nAvailable resources:");
for (i = 0; i < r; i++) {
    printf("\t%d", avl[i]);
}
printf("\n");

while (count != 0) {
    safe = 0;
    for (i = 0; i < p; i++) {
        if (running[i]) {
            exec = 1;
            for (j = 0; j < r; j++) {
                if (maxclaim[i][j] - curr[i][j] > avl[j]) {
                    exec = 0;
                    break;
                }
            }
            if (exec) {
                printf("\nProcess%d is executing\n", i + 1);
                running[i] = 0;
                count--;
                safe = 1;

                for (j = 0; j < r; j++) {
                    avl[j] += curr[i][j];
                }
                break;
            }
        }
    }
    if (!safe) {
        printf("\nThe processes are in unsafe state.\n");
        break;
```

```
    } else {
      printf("\nThe process is in safe state");
      printf("\nSafe sequence is:");

      for (i = 0; i < r; i++) {
        printf("\t%d", avl[i]);
      }

      printf("\n");
    }
  }
}
```

OUTPUT

```
Enter the number of processes: 5

Enter the number of resources: 3

Enter the resource for instance 1: 10

Enter the resource for instance 2: 5

Enter the resource for instance 3: 7

Enter maximum resource table:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3

Enter allocated resource table:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

The resource of instances:      10      5       7
The allocated resource table:
        0       1       0
        2       0       0
        3       0       2
        2       1       1
        0       0       2
```

```
The maximum resource table:
        7       5       3
        3       2       2
        9       0       2
        2       2       2
        4       3       3

Allocated resources:    7       2       5
Available resources:    3       3       2

Process2 is executing

The process is in safe state
Safe sequence is:       5       3       2

Process4 is executing

The process is in safe state
Safe sequence is:       7       4       3

Process1 is executing

The process is in safe state
Safe sequence is:       7       5       3

Process3 is executing

The process is in safe state
Safe sequence is:       10      5       5

Process5 is executing

The process is in safe state
Safe sequence is:       10      5       7
```

LAB-11

PROGRAM-1

AIM
Write a program to implement FIFO Page Replacement Algorithm.

THEORY
The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

LANGUAGE USED: C

SOURCE CODE

```
#include <stdio.h>

int main()
{
int referenceString[100], pageFaults = 0, m, n, s, pages, frames;
printf("\nEnter the number of Pages:\t");
scanf("%d", &pages);
printf("\nEnter reference string values:\n");
for( m = 0; m < pages; m++)
{
   printf("Value No. [%d]:\t", m + 1);
   scanf("%d", &referenceString[m]);
}
printf("\n What are the total number of frames:\t");
{
   scanf("%d", &frames);
}
int temp[frames];
for(m = 0; m < frames; m++)
{
  temp[m] = -1;
}
for(m = 0; m < pages; m++)
{
  s = 0;
  for(n = 0; n < frames; n++)
```

```
  {
    if(referenceString[m] == temp[n])
      {
        s++;
        pageFaults--;
      }
  }
  pageFaults++;
  if((pageFaults <= frames) && (s == 0))
    {
      temp[m] = referenceString[m];
    }
  else if(s == 0)
    {
      temp[(pageFaults - 1) % frames] = referenceString[m];
    }
    printf("\n");
    for(n = 0; n < frames; n++)
     {
       printf("%d\t", temp[n]);
     }
}
printf("\nTotal Page Faults:\t%d\n", pageFaults);
return 0;
}
```

OUTPUT

```
Enter the number of Pages:      7

Enter reference string values:
Value No. [1]:  1
Value No. [2]:  3
Value No. [3]:  0
Value No. [4]:  3
Value No. [5]:  5
Value No. [6]:  6
Value No. [7]:  3

 What are the total number of frames:   3

1       -1      -1
1       3       -1
1       3       0
1       3       0
5       3       0
5       6       0
5       6       3
Total Page Faults:      6


...Program finished with exit code 0
Press ENTER to exit console.
```

OPERATING SYSTEM

OPEN ENDED EXPERIMENT

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY, UTTAR PRADESH

GROUP MEMBERS -

- Anupriya Chandrasekhar (A2305220110)
- Drishti Arora (A2305220108)
- Vinayak Banga (A2305220106)

OPEN ENDED EXPERIMENT

PROGRAM 1

AIM

Write a script which will shows all running process on your Linux system boots up.

THEORY

LINUX KERNEL BOOT

There are many processes are running in the background when we press the power button of the system . It is very important to learn the booting process to understand the working of any operating system and to solve the ***booting error***.

A process is associated with any program running on your system, and is used to manage and monitor a program's memory usage, processor time, and I/O resources.
The ps Linux command creates a snapshot of currently running processes. Unlike the other commands on this list, ps presents the output as a static list, not updated in real time.
The ps command uses the following syntax:
ps [options]
The *ps aux* command is a tool to monitor processes running on your Linux system.
Since the ps aux command displays an overview of all the processes that are running, it is a great tool to understand and troubleshoot the health and state of your Linux system.
To display all running processes on your Linux system, follow the below mentioned steps:

1. Create a shell script i.e process.sh in this case
2. Type the following command in the shell script

   ps aux
3. Save the script using the: wq command
4. Run the shell script using the bash command i.e bash process.sh in this case

OUTPUT:

```
aib1@pamoli1-virtual-machine:~$ vi process.sh
aib1@pamoli1-virtual-machine:~$ bash process.sh
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 226328  9040 ?        Ss    2021   7:04 /sbin/init splash
root         2  0.0  0.0      0     0 ?        S     2021   0:00 [kthreadd]
root         4  0.0  0.0      0     0 ?        I<    2021   0:00 [kworker/0:0H]
root         6  0.0  0.0      0     0 ?        I<    2021   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0     0 ?        S     2021   6:31 [ksoftirqd/0]
root         8  0.0  0.0      0     0 ?        I     2021  67:03 [rcu_sched]
root         9  0.0  0.0      0     0 ?        I     2021   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S     2021   0:00 [migration/0]
root        11  0.0  0.0      0     0 ?        S     2021   0:52 [watchdog/0]
root        12  0.0  0.0      0     0 ?        S     2021   0:00 [cpuhp/0]
root        13  0.0  0.0      0     0 ?        S     2021   0:00 [kdevtmpfs]
root        14  0.0  0.0      0     0 ?        I<    2021   0:00 [netns]
root        15  0.0  0.0      0     0 ?        S     2021   0:00 [rcu_tasks_kthre]
root        16  0.0  0.0      0     0 ?        S     2021   0:00 [kauditd]
root        17  0.0  0.0      0     0 ?        S     2021   0:41 [khungtaskd]
root        18  0.0  0.0      0     0 ?        S     2021   0:00 [oom_reaper]
root        19  0.0  0.0      0     0 ?        I<    2021   0:00 [writeback]
root        20  0.0  0.0      0     0 ?        S     2021   0:00 [kcompactd0]
root        21  0.0  0.0      0     0 ?        SN    2021   0:00 [ksmd]
root        22  0.0  0.0      0     0 ?        SN    2021   0:49 [khugepaged]
root        23  0.0  0.0      0     0 ?        I<    2021   0:00 [crypto]
root        24  0.0  0.0      0     0 ?        I<    2021   0:00 [kintegrityd]
root        25  0.0  0.0      0     0 ?        I<    2021   0:00 [kblockd]
root        26  0.0  0.0      0     0 ?        I<    2021   0:00 [ata_sff]
root        27  0.0  0.0      0     0 ?        I<    2021   0:00 [md]
root        28  0.0  0.0      0     0 ?        I<    2021   0:00 [edac-poller]
root        29  0.0  0.0      0     0 ?        I<    2021   0:00 [devfreq_wq]
root        30  0.0  0.0      0     0 ?        I<    2021   0:00 [watchdogd]
root        34  0.0  0.0      0     0 ?        S     2021   0:15 [kswapd0]
root        35  0.0  0.0      0     0 ?        I<    2021   0:00 [kworker/u3:0]
root        36  0.0  0.0      0     0 ?        S     2021   0:00 [ecryptfs-kthrea]
root        78  0.0  0.0      0     0 ?        I<    2021   0:00 [kthrotld]
root        79  0.0  0.0      0     0 ?        I<    2021   0:00 [acpi_thermal_pm]
root        80  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_0]
root        81  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_0]
root        82  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_1]
root        83  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_1]
root        86  0.0  0.0      0     0 ?        I     2021   0:00 [kworker/0:2]
root        90  0.0  0.0      0     0 ?        I<    2021   0:00 [ipv6_addrconf]
root        99  0.0  0.0      0     0 ?        I<    2021   0:00 [kstrp]
root       116  0.0  0.0      0     0 ?        I<    2021   0:00 [charger_manager]
```

```
root       163  0.0  0.0      0     0 ?        I<    2021   0:00 [mpt_poll_0]
root       164  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_2]
root       165  0.0  0.0      0     0 ?        I<    2021   0:00 [mpt/0]
root       166  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_2]
root       167  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_3]
root       168  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_3]
root       169  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_4]
root       170  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_4]
root       171  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_5]
root       172  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_5]
root       173  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_6]
root       174  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_6]
root       175  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_7]
root       176  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_7]
root       177  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_8]
root       178  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_8]
root       179  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_9]
root       180  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_9]
root       181  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_10]
root       182  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_10]
root       183  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_11]
root       184  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_11]
root       185  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_12]
root       186  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_12]
root       187  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_13]
root       188  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_13]
root       189  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_14]
root       190  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_14]
root       191  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_15]
root       192  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_15]
root       193  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_16]
root       194  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_16]
root       195  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_17]
root       196  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_17]
root       197  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_18]
root       198  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_18]
root       199  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_19]
root       200  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_19]
root       201  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_20]
root       202  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_20]
root       203  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_21]
root       204  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_21]
root       205  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_22]
root       206  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_22]
```

```
root       207  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_23]
root       208  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_23]
root       209  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_24]
root       210  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_24]
root       211  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_25]
root       212  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_25]
root       213  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_26]
root       214  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_26]
root       215  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_27]
root       216  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_27]
root       217  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_28]
root       218  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_28]
root       219  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_29]
root       220  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_29]
root       221  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_30]
root       222  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_30]
root       223  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_31]
root       224  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_31]
root       252  0.0  0.0      0     0 ?        S     2021   0:00 [scsi_eh_32]
root       253  0.0  0.0      0     0 ?        I<    2021   0:00 [scsi_tmf_32]
root       255  0.0  0.0      0     0 ?        I<    2021   0:00 [ttm_swap]
root       256  0.0  0.0      0     0 ?        S     2021   0:00 [irq/16-vmwgfx]
root       260  0.0  0.0      0     0 ?        I<    2021   1:00 [kworker/0:1H]
aib1       310  0.0  0.0 110084  5468 ?        S    16:12   0:00 sshd: aib1@pts/11
aib1       311  0.0  0.0  22396  4940 pts/11   Ss+  16:12   0:00 -bash
root       352  0.0  0.0      0     0 ?        S     2021   1:01 [jbd2/sda1-8]
root       353  0.0  0.0      0     0 ?        I<    2021   0:00 [ext4-rsv-conver]
root       391  0.0  1.9 402472 162328 ?       S<s   2021   5:37 /lib/systemd/systemd-journald
root       405  0.0  0.0  36100  5644 ?        Ss    2021   0:18 /lib/systemd/systemd-udevd
root       413  0.0  0.0      0     0 ?        S<    2021   0:00 [loop1]
root       421  0.0  0.0      0     0 ?        S<    2021   0:00 [loop2]
root       439  0.0  0.0      0     0 ?        S<    2021   0:00 [loop4]
root       447  0.0  0.0      0     0 ?        S<    2021   0:00 [loop9]
root       455  0.0  0.0      0     0 ?        S<    2021   0:00 [loop12]
root       457  0.0  0.0      0     0 ?        S<    2021   0:00 [loop14]
root       468  0.0  0.0      0     0 ?        S<    2021   0:00 [loop17]
root       469  0.0  0.0      0     0 ?        S<    2021   0:00 [loop18]
root       478  0.0  0.0      0     0 ?        S<    2021   0:00 [loop22]
root       494  0.0  0.0 107788  7212 ?        Ss   16:18   0:00 sshd: aib1 [priv]
aib1       560  0.0  0.0 110084  5444 ?        S    16:18   0:00 sshd: aib1@pts/15
aib1       561  0.0  0.0  22396  4768 pts/15   Ss+  16:18   0:00 -bash
systemd+   601  0.0  0.0 146136  2312 ?        Ssl   2021   0:26 /lib/systemd/systemd-timesyncd
systemd+   602  0.0  0.0  70924  5148 ?        Ss    2021   1:27 /lib/systemd/systemd-resolved
aib1       684  0.0  0.0  32444  4108 pts/15   T    16:22   0:00 vi charan1
```

```
aib1        790  0.0  0.0  22396  4728 pts/16   Ss+  16:28   0:00 -bash
root        794  0.0  0.0  45232  2136 ?        Ss    2021   1:05 /sbin/wpa_suppl
root        796  0.0  0.0 427260  4544 ?        Ssl   2021   0:23 /usr/sbin/Modem
root        798  0.0  0.1 487280 11520 ?        Ssl   2021  28:21 /usr/sbin/Netwo
root        799  0.0  0.0  71028  5712 ?        Ss    2021   1:40 /lib/systemd/sy
syslog      808  0.0  0.0 263036  4092 ?        Ssl   2021   0:55 /usr/sbin/rsysl
avahi       814  0.0  0.0  47256  2816 ?        Ss    2021 170:12 avahi-daemon: r
root        815  0.0  0.2 303548 20532 ?        Ssl   2021  94:05 /usr/lib/accoun
root        816  0.0  0.1 503720  8580 ?        Ssl   2021   0:32 /usr/lib/udisks
root        828  0.0  0.0   4552   700 ?        Ss    2021   0:00 /usr/sbin/acpid
root        829  0.0  0.0  31320  2764 ?        Ss    2021   0:32 /usr/sbin/cron
avahi       860  0.0  0.0  47076    40 ?        S     2021   0:00 avahi-daemon: c
root        916  0.0  0.0 301332  6044 ?        Ssl   2021   0:00 /usr/sbin/gdm3
root        918  0.0  0.0  72304  4204 ?        Ss    2021   0:30 /usr/sbin/sshd
root        923  0.0  0.2 255848 23240 ?        Sl    2021  12:24 gdm-session-wor
root        936  0.0  0.0 107788  7100 ?        Ss   16:35   0:00 sshd: aib1 [pri
whoopsie    946  0.0  0.1 538060  9844 ?        Ssl   2021   1:03 /usr/bin/whoops
kernoops    965  0.0  0.0  56944    76 ?        Ss    2021   4:02 /usr/sbin/kerne
kernoops    967  0.0  0.0  56944    80 ?        Ss    2021   4:02 /usr/sbin/kerne
aib1       1007  0.0  0.0 110084  5384 ?        S    16:35   0:00 sshd: aib1@pts/
aib1       1008  0.0  0.0  22396  4860 pts/17   Ss+  16:35   0:00 -bash
gdm        1053  0.0  0.0  77328  7136 ?        Ss    2021   0:16 /lib/systemd/sy
gdm        1057  0.0  0.0 114076   288 ?        S     2021   0:00 (sd-pam)
gdm        1078  0.0  0.0 190692  3700 tty1     Ssl+  2021   0:00 /usr/lib/gdm3/g
gdm        1080  0.0  0.0  50348  4044 ?        Ss    2021   0:00 /usr/bin/dbus-d
gdm        1082  0.0  0.1 551884  9332 tty1     Sl+   2021   0:15 /usr/lib/gnome-
gdm        1088  0.0  3.6 3115788 295200 tty1   Sl+   2021 129:21 /usr/bin/gnome-
root       1119  0.0  0.0 306880  5660 ?        Ssl   2021   0:08 /usr/lib/upower
gdm        1123  0.0  0.0 219496  6668 tty1     S+    2021   0:09 /usr/bin/Xwayla
root       1124  0.0  0.0 107788  7212 ?        Ss   16:36   0:00 sshd: aib1 [pri
gdm        1126  0.0  0.0 349408  4212 ?        Ssl   2021   0:00 /usr/lib/at-spi
gdm        1131  0.0  0.0  49928  3124 ?        S     2021   0:00 /usr/bin/dbus-d
gdm        1133  0.0  0.0 220768  4940 ?        Sl    2021   0:00 /usr/lib/at-spi
gdm        1136  0.0  0.0 1236696 6188 ?        Ssl   2021   0:07 /usr/bin/pulsea
rtkit      1138  0.0  0.0 183508  2712 ?        SNsl  2021   2:14 /usr/lib/rtkit/
gdm        1149  0.0  0.0 354260  5356 tty1     Sl    2021   0:00 ibus-daemon --x
gdm        1152  0.0  0.0 273640  3824 tty1     Sl    2021   0:00 /usr/lib/ibus/i
gdm        1155  0.0  0.2 481964 17684 tty1     Sl    2021   0:00 /usr/lib/ibus/i
gdm        1159  0.0  0.0 271584  3708 ?        Sl    2021   0:00 /usr/lib/ibus/i
root       1171  0.0  0.0 289700  5084 ?        Ssl   2021   0:19 /usr/lib/x86_64
root       1173  0.0  1.3 515864 106680 ?       Ssl   2021  14:56 /usr/lib/packag
gdm        1174  0.0  0.2 487476 17088 tty1     Sl+   2021   0:00 /usr/lib/gnome-
gdm        1182  0.0  0.0 271052  4152 tty1     Sl+   2021   0:00 /usr/lib/gnome-
gdm        1184  0.0  0.1 337156 15412 tty1     Sl+   2021   0:00 /usr/lib/gnome-
```

```
root      4512  0.0  0.0      0     0 ?        S<    2021   0:00 [loop10]
root      5279  0.0  0.0      0     0 ?        S<   Jan12   0:00 [loop16]
umesh     5954  0.0  0.0   4636   836 tty4     S+   Apr08   0:00 /bin/sh -c /usr
umesh     5955  0.0  0.6 482472 51700 tty4     Sl+  Apr08   0:01 /usr/bin/python
aib7      9469  0.0  0.0   4628   848 tty3     S+   Jan21   0:00 /bin/sh -c /usr
aib7      9470  0.0  0.5 482052 41788 tty3     Sl+  Jan21   0:01 /usr/bin/python
root      9653  0.0  0.0      0     0 ?        S<    2021   0:00 [loop0]
root      9691  0.0  0.0      0     0 ?        I<    2021   0:00 [xfsalloc]
root      9694  0.0  0.0      0     0 ?        I<    2021   0:00 [xfs_mru_cache]
root      9703  0.0  0.0      0     0 ?        S     2021   0:00 [jfsIO]
root      9704  0.0  0.0      0     0 ?        S     2021   0:00 [jfsCommit]
root      9705  0.0  0.0      0     0 ?        S     2021   0:00 [jfsSync]
pamoli1   9727  0.0  1.5 1382348 122944 ?      Sl   Apr12   5:28 gnome-control-c
umesh     9749  0.0  0.0   4628   776 tty4     S+   Feb25   0:00 /bin/sh -c /usr
umesh     9750  0.0  0.5 481956 41576 tty4     Sl+  Feb25   0:01 /usr/bin/python
root      9894  0.0  0.2 409708 24440 ?        Sl    2021  12:31 gdm-session-wor
pamoli1   9898  0.0  0.0  77416  7460 ?        Ss    2021   0:17 /lib/systemd/sy
pamoli1   9899  0.0  0.0 196004   292 ?        S     2021   0:00 (sd-pam)
pamoli1   9912  0.0  0.0 281276  5020 ?        Sl    2021   0:00 /usr/bin/gnome-
pamoli1   9916  0.0  0.0 205028  4724 tty2     Ssl+  2021   0:00 /usr/lib/gdm3/g
pamoli1   9918  0.0  0.9 431280 81628 tty2     Sl+   2021   1:50 /usr/lib/xorg/X
pamoli1   9921  0.0  0.0  51228  5484 ?        Ss    2021   0:05 /usr/bin/dbus-d
pamoli1   9925  0.0  0.1 625872 11108 tty2     Sl+   2021   0:16 /usr/lib/gnome-
root      9936  0.0  0.0      0     0 ?        S<    2021   0:00 [loop3]
pamoli1  10018  0.0  0.0  11308   320 ?        Ss    2021   1:02 /usr/bin/ssh-ag
pamoli1  10020  0.0  0.0 349292  4948 ?        Ssl   2021   0:00 /usr/lib/at-spi
pamoli1  10025  0.0  0.0  50060  3840 ?        S     2021   0:00 /usr/bin/dbus-d
pamoli1  10028  0.0  0.0 220784  5320 ?        Sl    2021   0:00 /usr/lib/at-spi
pamoli1  10046  0.1  5.1 3082840 418892 tty2   Sl+   2021 271:10 /usr/bin/gnome-
pamoli1  10052  0.0  0.0 284964  5612 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
pamoli1  10057  0.0  0.0 416116  4232 ?        Sl    2021   0:00 /usr/lib/gvfs/g
pamoli1  10068  0.0  0.1 1253780 8652 ?        S<l   2021   0:07 /usr/bin/pulsea
pamoli1  10077  0.0  0.0 354520  6260 tty2     Sl    2021   0:02 ibus-daemon --x
pamoli1  10081  0.0  0.0 273640  4804 tty2     Sl    2021   0:00 /usr/lib/ibus/i
pamoli1  10083  0.0  0.2 337384 17736 tty2     Sl    2021   0:00 /usr/lib/ibus/i
pamoli1  10085  0.0  0.0 271584  4192 ?        Sl    2021   0:00 /usr/lib/ibus/i
pamoli1  10096  0.0  0.1 689788 11812 ?        Sl    2021   0:00 /usr/lib/gnome-
pamoli1  10102  0.0  0.2 971840 16732 ?        Ssl   2021   0:01 /usr/lib/evolut
pamoli1  10112  0.0  0.0 188012  4404 ?        Sl    2021   0:00 /usr/lib/dconf/
pamoli1  10117  0.0  0.2 783844 16564 ?        Sl    2021   0:36 /usr/lib/gnome-
pamoli1  10118  0.0  0.0 299560  7424 ?        Ssl   2021   1:20 /usr/lib/gvfs/g
pamoli1  10124  0.0  0.0 268732  3880 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
pamoli1  10128  0.0  0.0 266940  4416 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
pamoli1  10137  0.0  0.0 296256  5448 ?        Sl    2021   0:05 /usr/lib/gnome-
```

```
aib7     28041  0.0  0.0 280988  5608 ?        Sl    2021   0:00 /usr/bin/gnome-
aib7     28050  0.0  0.0 284864  5328 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
aib7     28055  0.0  0.0 416116  4056 ?        Sl    2021   0:00 /usr/lib/gvfs/g
aib7     28069  0.1  5.2 3046292 426392 tty3   Sl+   2021 201:07 /usr/bin/gnome-
aib7     28079  0.0  0.0 1187964 7788 ?        S<l   2021   0:07 /usr/bin/pulsea
aib7     28088  0.0  0.0 354376  6096 tty3     Sl    2021   0:00 ibus-daemon --x
aib7     28092  0.0  0.1 689656 11104 ?        Sl    2021   0:00 /usr/lib/gnome-
aib7     28096  0.0  0.2 971980 16524 ?        Ssl   2021   0:00 /usr/lib/evolut
aib7     28102  0.0  0.0 187920  4368 ?        Sl    2021   0:00 /usr/lib/dconf/
aib7     28108  0.0  0.2 857568 16480 ?        Sl    2021   0:29 /usr/lib/gnome-
aib7     28115  0.0  0.0 273632  4972 tty3     Sl    2021   0:00 /usr/lib/ibus/i
aib7     28118  0.0  0.2 337376 17684 tty3     Sl    2021   0:00 /usr/lib/ibus/i
aib7     28121  0.0  0.0 271452  4152 ?        Sl    2021   0:00 /usr/lib/ibus/i
aib7     28131  0.0  0.0 369988  5320 ?        Sl    2021   0:05 /usr/lib/gnome-
aib7     28147  0.0  0.0 447048  7404 ?        Ssl   2021   1:07 /usr/lib/gvfs/g
aib7     28153  0.0  0.0 268732  3844 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
aib7     28157  0.0  0.0 266940  4420 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
aib7     28161  0.0  0.0 371724  6200 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
aib7     28166  0.0  0.0 281524  3976 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
aib7     28170  0.0  0.2 510632 18888 tty3     Sl+   2021   0:21 /usr/lib/gnome-
aib7     28171  0.0  0.1 342224  8676 tty3     Sl+   2021   0:01 /usr/lib/gnome-
aib7     28173  0.0  0.0 416240  4536 tty3     Sl+   2021   0:22 /usr/lib/gnome-
aib7     28174  0.0  0.0 268628  3464 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28176  0.0  0.0 445728  7416 tty3     Sl+   2021   0:43 /usr/lib/gnome-
aib7     28181  0.0  0.0 370952  3796 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28183  0.0  0.0 327860  5820 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28185  0.0  0.2 487316 18428 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28187  0.0  0.2 421884 18112 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28198  0.0  0.0 271040  4516 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28202  0.0  0.2 336996 17440 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28205  0.6  0.6 689288 54472 tty3     Sl+   2021 1154:55 /usr/lib/gnome
aib7     28207  0.0  0.1 462628 10572 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28210  0.0  0.0 357336  5372 tty3     Sl+   2021   4:43 /usr/lib/gnome-
aib7     28212  0.0  0.5 500380 40884 tty3     Sl+   2021  21:12 /usr/lib/gnome-
aib7     28214  0.0  0.2 788364 18432 tty3     Sl+   2021   0:08 /usr/lib/gnome-
aib7     28215  0.0  0.0 271044  4388 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28231  0.0  0.1 501664 10676 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28270  0.0  9.6 1498772 787244 tty3   SLl+  2021  11:45 /usr/bin/gnome-
aib7     28271  0.0  0.9 1310120 78024 tty3    Sl+   2021   3:58 /usr/lib/gnome-
aib7     28272  0.0  0.0 271936  5080 tty3     Sl+   2021   0:00 /usr/lib/gnome-
aib7     28273  0.0  0.4 772908 38072 tty3     Sl+   2021   0:01 nautilus-deskto
aib7     28289  0.0  0.0 361136  5780 ?        Sl    2021   4:10 /usr/lib/gvfs/g
aib7     28313  0.0  0.6 885564 52996 ?        Ssl   2021   0:00 /usr/lib/evolut
aib7     28322  0.0  0.5 937240 48608 ?        Sl    2021   0:30 /usr/lib/evolut
```

```
umesh    30117  0.0  0.0  51296  5312 ?        Ss    2021   0:02 /usr/bin/dbus-d
umesh    30121  0.0  0.1 625876 12304 tty4     Sl+   2021   0:15 /usr/lib/gnome-
aib1     30123  0.0  0.0  32332  4020 pts/3    S+   15:31   0:00 vi main
root     30207  0.0  0.0 107788  7104 ?        Ss   15:39   0:00 sshd: aib1 [pri
umesh    30215  0.0  0.0  11308   316 ?        Ss    2021   0:39 /usr/bin/ssh-ag
umesh    30217  0.0  0.0 349292  5720 ?        Ssl   2021   0:00 /usr/lib/at-spi
umesh    30222  0.0  0.0  50060  3876 ?        S     2021   0:00 /usr/bin/dbus-d
umesh    30225  0.0  0.0 220784  6128 ?        Sl    2021   0:00 /usr/lib/at-spi
umesh    30239  0.0  0.0 284864  6100 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
umesh    30244  0.0  0.0 416116  5812 ?        Sl    2021   0:00 /usr/lib/gvfs/g
umesh    30259  0.1  4.8 3028780 395576 tty4   Sl+   2021 184:42 /usr/bin/gnome-
umesh    30268  0.0  0.1 1187980 8992 ?        S<l   2021   0:07 /usr/bin/pulsea
umesh    30278  0.0  0.0 427996  6772 tty4     Sl    2021   0:00 ibus-daemon --x
umesh    30282  0.0  0.0 273640  5620 tty4     Sl    2021   0:00 /usr/lib/ibus/i
umesh    30286  0.0  0.2 337376 18448 tty4     Sl    2021   0:00 /usr/lib/ibus/i
umesh    30289  0.0  0.0 271584  5236 ?        Sl    2021   0:00 /usr/lib/ibus/i
umesh    30298  0.0  0.1 689660 11960 ?        Sl    2021   0:00 /usr/lib/gnome-
umesh    30302  0.0  0.2 971904 17776 ?        Ssl   2021   0:00 /usr/lib/evolut
umesh    30311  0.0  0.2 783844 16788 ?        Sl    2021   0:29 /usr/lib/gnome-
umesh    30320  0.0  0.0 369988  6384 ?        Sl    2021   0:04 /usr/lib/gnome-
umesh    30330  0.0  0.1 447056  8512 ?        Ssl   2021   1:07 /usr/lib/gvfs/g
umesh    30336  0.0  0.0 268732  4684 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
umesh    30340  0.0  0.0 266940  5332 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
umesh    30344  0.0  0.0 371724  6988 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
umesh    30349  0.0  0.0 281524  4948 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
umesh    30353  0.0  0.2 510632 19488 tty4     Sl+   2021   0:21 /usr/lib/gnome-
umesh    30354  0.0  0.1 342228  9476 tty4     Sl+   2021   0:01 /usr/lib/gnome-
umesh    30356  0.0  0.0 416240  5324 tty4     Sl+   2021   0:21 /usr/lib/gnome-
umesh    30359  0.0  0.0 268628  4424 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30364  0.0  0.1 445728  8984 tty4     Sl+   2021   0:42 /usr/lib/gnome-
umesh    30367  0.0  0.0 370956  4660 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30372  0.0  0.0 327868  6664 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30376  0.0  0.2 487320 19168 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30378  0.0  0.2 421864 18976 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30381  0.0  0.0   4636   780 tty4     S+   Mar25   0:00 /bin/sh -c /usr
umesh    30382  0.0  0.5 482424 47220 tty4     Sl+  Mar25   0:01 /usr/bin/python
aib1     30384  0.0  0.0 110084  5308 ?        S    15:39   0:00 sshd: aib1@pts/
umesh    30385  0.0  0.0 271044  5304 tty4     Sl+   2021   0:00 /usr/lib/gnome-
aib1     30386  0.0  0.0  22396  4840 pts/4    Ss+  15:39   0:00 -bash
umesh    30391  0.0  0.2 336996 18112 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30396  0.5  0.6 687964 53812 tty4     Sl+   2021 1063:14 /usr/lib/gnome
umesh    30398  0.0  0.1 462632 11208 tty4     Sl+   2021   0:00 /usr/lib/gnome-
umesh    30399  0.0  0.0 357336  6188 tty4     Sl+   2021   4:32 /usr/lib/gnome-
umesh    30401  0.0  0.2 491376 18596 tty4     Sl+   2021   0:00 /usr/lib/gnome-
```

```
umesh    30511  0.0  0.6 932692 49404 ?        Sl    2021   0:29 /usr/lib/evolut
umesh    30513  0.0  0.0 187908  4984 ?        Sl    2021   0:00 /usr/lib/dconf/
umesh    30534  0.0  0.1 725604 13536 ?        Ssl   2021   0:00 /usr/lib/evolut
umesh    30543  0.0  0.1 879036 13884 ?        Sl    2021   0:29 /usr/lib/evolut
umesh    30557  0.0  0.0 197784  5808 tty4     Sl    2021   0:00 /usr/lib/ibus/i
root     30586  0.0  0.0 107788  7200 ?        Ss   15:41   0:00 sshd: aib1 [pri
aib1     30652  0.0  0.0 110084  5560 ?        S    15:41   0:00 sshd: aib1@pts/
aib1     30653  0.0  0.0  22396  4872 pts/5    Ss+  15:41   0:00 -bash
root     30683  0.0  0.0      0     0 ?        S<   Apr07   0:00 [loop6]
umesh    30685  0.0  0.2 661816 22556 tty4     Sl+   2021   3:46 update-notifier
umesh    30758  0.0  0.0 197368  5356 ?        Ssl   2021   0:00 /usr/lib/gvfs/g
root     30780  0.0  0.0 107788  7184 ?        Ss   15:59   0:00 sshd: aib1 [pri
umesh    30834  0.0  0.2 871064 18196 tty4     Sl+   2021   0:28 /usr/lib/deja-d
aib1     30872  0.0  0.0 110084  5700 ?        S    16:00   0:00 sshd: aib1@pts/
aib1     30873  0.0  0.0  22528  5144 pts/7    Ss   16:00   0:00 -bash
root     31000  0.0  0.4 765428 35920 ?        Ssl  Apr07   1:52 /usr/lib/snapd/
root     31004  0.0  0.0      0     0 ?        S<    2021   0:00 [loop21]
root     31140  0.0  0.0 107788  7188 ?        Ss   16:01   0:00 sshd: aib1 [pri
aib1     31208  0.0  0.0 110084  5632 ?        S    16:01   0:00 sshd: aib1@pts/
aib1     31209  0.0  0.0  22396  4800 pts/9    Ss+  16:01   0:00 -bash
root     31405  0.0  0.0 107788  7228 ?        Ss   16:02   0:00 sshd: aib1 [pri
aib1     31473  0.0  0.0 110216  5616 ?        S    16:02   0:00 sshd: aib1@pts/
aib1     31474  0.0  0.0  22544  4772 pts/8    Ss+  16:02   0:00 -bash
aib1     31837  0.0  0.0  32504  4060 pts/7    S+   16:04   0:00 vi array
root     31852  0.0  0.0 107788  7160 ?        Ss   16:06   0:00 sshd: aib1 [pri
aib1     31918  0.0  0.0 110084  5604 ?        S    16:06   0:00 sshd: aib1@pts/
aib1     31923  0.0  0.0  22396  4896 pts/10   Ss+  16:06   0:00 -bash
root     32192  0.0  0.0 107788  7204 ?        Ss   16:07   0:00 sshd: aib1 [pri
aib7     32194  0.0  0.0   4636   764 tty3     S+   Mar18   0:00 /bin/sh -c /usr
aib7     32195  0.0  0.5 482396 47088 tty3     Sl+  Mar18   0:01 /usr/bin/python
root     32196  0.0  0.0 107788  7192 ?        Ss   16:07   0:00 sshd: aib1 [pri
aib1     32317  0.0  0.0 110084  5724 ?        S    16:07   0:00 sshd: aib1@pts/
aib1     32331  0.0  0.0 110084  5708 ?        S    16:07   0:00 sshd: aib1@pts/
aib1     32332  0.0  0.0  22396  4660 pts/12   Ss   16:07   0:00 -bash
aib1     32335  0.0  0.0  22396  4836 pts/13   Ss+  16:07   0:00 -bash
aib1     32429  0.0  0.0  32332  4100 pts/12   S+   16:07   0:00 vi abcd
root     32435  0.0  0.0 107788  7184 ?        Ss   16:09   0:00 sshd: aib1 [pri
aib1     32506  0.0  0.0 110084  5628 ?        S    16:09   0:00 sshd: aib1@pts/
aib1     32507  0.0  0.0  22396  4840 pts/14   Ss   16:09   0:00 -bash
aib1     32602  0.0  0.0  32332  4084 pts/14   S+   16:09   0:00 vi arrays
root     32619  0.0  0.0      0     0 ?        I    16:12   0:00 [kworker/u2:0]
root     32712  0.0  0.0 107788  7264 ?        Ss   16:12   0:00 sshd: aib1 [pri
aib1@pamoli1-virtual-machine:~$ vi exp1.sh
```

PROGRAM 2

AIM

WAP to generate maximum number of child process in your system and with the help of program explain what are Zombie processes.

THEORY:

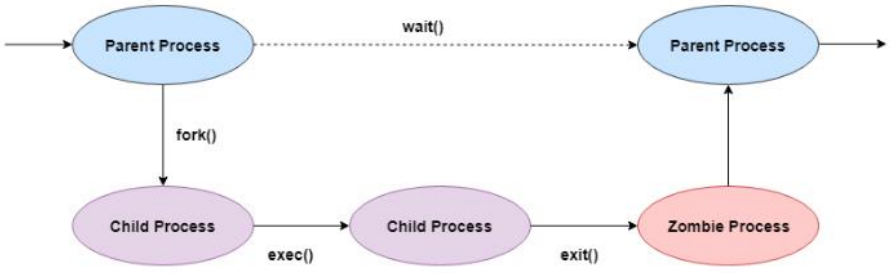
ZOMBIE PROCESS

A zombie process is a process whose execution is completed but it still has an entry in the process table. Zombie processes usually occur for child processes, as the parent process still needs to read its child's exit status. Once this is done using the wait system call, the zombie process is eliminated from the process table. This is known as reaping the zombie process.
In the C and C++ programming languages, **unistd.h** is the name of the header file that provides access to the POSIX operating system API. On Unix-like systems, the interface defined by unistd.h is typically made up largely of system call wrapper functions such as fork, pipe and I/O primitives.

DANGERS OF ZOMBIE PROCESSES

Zombie processes don't use any system resources but they do retain their process ID. If there are a lot of zombie processes, then all the available process ID's are monopolized by them. This prevents other processes from running as there are no process ID's available.

The presence of zombie processes also indicates an operating system bug if their parent processes are not running anymore. This is not a serious problem if there are a few zombie processes but under heavier loads, this can create issues for the system.



Zombie Process in Linux

IMPORTANT FUNCTIONS:

fork(): It is the primary method of process creation on Unix-like operating systems. This function creates a new copy called the *child* out of the original process, that is called the *parent*. When the parent process closes or crashes for some reason, it also kills the child process.

wait() : A call to wait() blocks the calling process until one of its child processes exits or a signal is received. After child process terminates, parent continues its execution after wait system call instruction.

SOURCE CODE

```
#include<iostream>
#include<unistd.h>
using namespace std;

int main(){
   int num=0;
   while(fork() > 0){
num++;
cout<<num<<" ";
cout<<getpid()<<" ";
cout<<getppid()<<" ";
cout<<endl;
   }
return 0;
}
```

OUTPUT

```
1 306 305
2 306 305
3 306 305
4 306 305
5 306 305
6 306 305
7 306 305
8 306 305
9 306 305
10 306 305
11 306 305
12 306 305
13 306 305
14 306 305
15 306 305
16 306 305
17 306 305
18 306 305
19 306 305
20 306 305
21 306 305
22 306 305
23 306 305
24 306 305
25 306 305
26 306 305
27 306 305
28 306 305
29 306 305
30 306 305
31 306 305
32 306 305
33 306 305
34 306 305
35 306 305
36 306 305
37 306 305
38 306 305
39 306 305
40 306 305
41 306 305
42 306 305
43 306 305
44 306 305
45 306 305
```

```
45 306 305
46 306 305
47 306 305
48 306 305
49 306 305
50 306 305
51 306 305
52 306 305
53 306 305
54 306 305
55 306 305
56 306 305
57 306 305
58 306 305
59 306 305
60 306 305
61 306 305
62 306 305
63 306 305
64 306 305
65 306 305
66 306 305
67 306 305
68 306 305
69 306 305
70 306 305
71 306 305
72 306 305
73 306 305
74 306 305
75 306 305
76 306 305
77 306 305
78 306 305
79 306 305
80 306 305
81 306 305
82 306 305
83 306 305
84 306 305
85 306 305
86 306 305
87 306 305
88 306 305
89 306 305
```

```
53 306 305
54 306 305
55 306 305
56 306 305
57 306 305
58 306 305
59 306 305
60 306 305
61 306 305
62 306 305
63 306 305
64 306 305
65 306 305
66 306 305
67 306 305
68 306 305
69 306 305
70 306 305
71 306 305
72 306 305
73 306 305
74 306 305
75 306 305
76 306 305
77 306 305
78 306 305
79 306 305
80 306 305
81 306 305
82 306 305
83 306 305
84 306 305
85 306 305
86 306 305
87 306 305
88 306 305
89 306 305
90 306 305
91 306 305
92 306 305
93 306 305


...Program finished with exit code 0
Press ENTER to exit console.
```