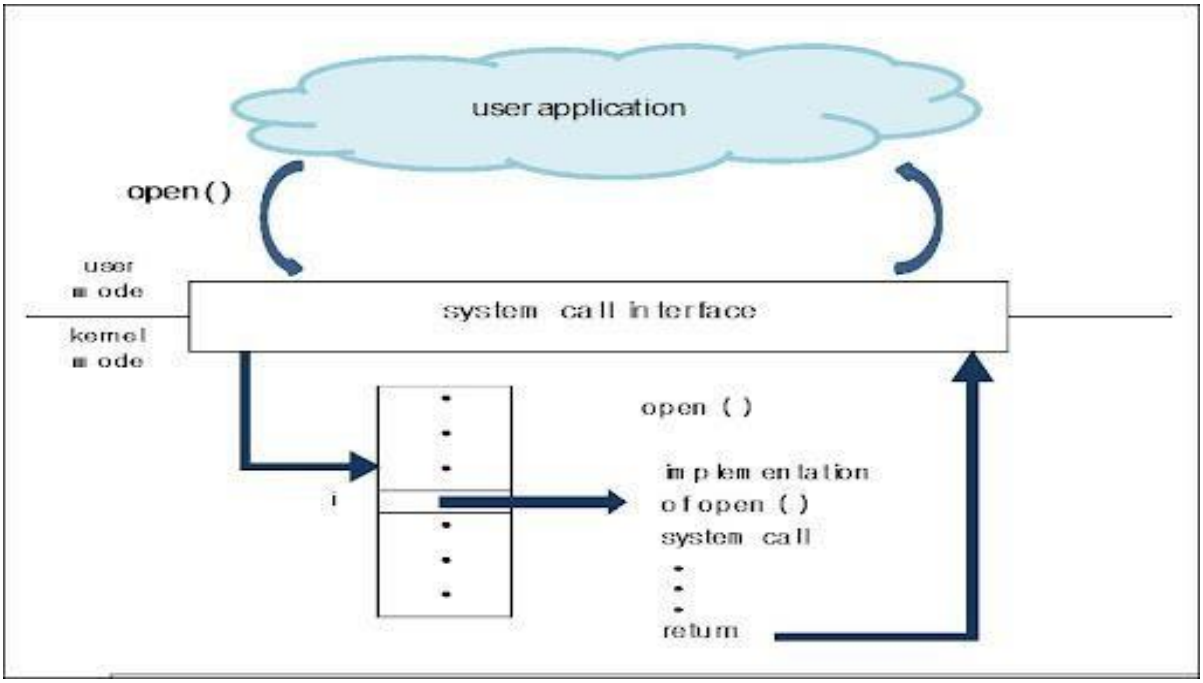
Operating System:

- A program that manages the computer hardware . It also provides a basics for application programs and acts as intermediary between the computer user and the application hardware.
- OS is a **resource allocator :** Manages all resources and decides between conflicting requests for efficient and fair resource use.
- OS is a **control program :** Controls execution of programs to prevent errors and Improper use of the computer.
- The one program running at all times on the computers(usually called the **kernel**)

❖ System Calls

- Programming interface to the services provided by the OS.
- Interface between the process and the operating system.
- Typically written in a high level language(C or C++)



Linux

- Linux is generic term referring to Unix- like computer operating system based on the Linux Kernel.
- The development of Linux is one of the most prominent examples of free and open source software collaboration typically and the underlying source code can be used, freely modified , and redistributed by anyone.
- Because Linux takes the UNIX system as its inspiration, Linux and UNIX programs are very similar. In fact, almost all programs written for UNIX can be compiled and run Linux. Also, many commercial application sold for commercial versions of UNIX can run unchanged in binary form on Linux systems.
- The problem with UNIX is that is has always been expensive and taken large computers to use. Some versions of UNIX have been available for personal computer-type hardware, but cost has been very prohibitive and the support by multiple vendors has been lacking. These problems are what let to the development of Linux.
- Linux was developed by "Linus Torvalds" at the University of Helsinki, with the help of UNIX programs from across the Internet.

Linux Architecture

The Linux architecture comprises two main sections: the **Kernel space** and **User space**

- **Kernel space**

  The Kernel space is where all of the system level processes happen. These processes are things affect the entire system and have to be very stable and well maintained. A program in kernel space can cause the system to crash. The main resident in kernel space is, of course, the kernel. The kernel is the piece of software manages memory allocation of processes and divides of the CPU's time appropriately. The kernel also contains the drivers for the hardware devices installed in the system. The kernel is the core of the Linux operating system.

- **User space**

  The User space manages the user processes run by people working on the system. User processes are things such as your e-mail client, web browser, or word processor. These processes work with the kernel to handle low level functions such as printing to the screen or talking tom storage hardware. But, since these functions are not handled in kernel space, a corrupted user application will not bring the entire system down.

Directory Structure

File system :

- All the files are grouped together in the directory structure. The file system is arranged in the hierarchical structure, like an inverted tree . The top of the hierarchy is traditionally called root (written as a slash/)
- Linux sorts directories Descending from the root directory according to their importance to the boot process.
- File systems from other hand drive partitions mount to directories beneath the root directory, providing access to a single directory structure.
- It'sThe File system hierarchy standard (FHS) Governs the unified file system for Linux by defining a standard set of directories, sub directories and files.
- Linux is case sensitive operating system

Directory	Description
/	The Root directory, all directories are below the/(root directory)
/ bin	Contains Binary commands available to all users.
/boot	Contains Kernel and boot loader files.
/dev	Contains device files.
/etc	Contains system configuration files.
/home	Contains by default the user home directories.
/lib	Contains shared programs libraries and kernel modules.
/root	Home directory further root user.
/media	Mount point for removable media.
/mnt	Mount point for mounting a file system temporarily.
/opt	Add on applications software packages.
/sbin	Contains system binary commands.
/proc	Contains information about system state and processes.
/srv	Contains the files for services like FTP and Web servers.
/tmp	Contains temporary files.
/usr	Contains system commands and utilities.
/var	Contains data files that are changed constantly.

Linux Distributions

Linux is actually just a kernel, So to create a complete Linux system you have to install the source code of kernel and many other freely distributed software programs.

Usually distributions are put on CD that contains the kernel and programming tools and utilities.

These distributions usually come with a set up program on CD to install a Linux system, they have the same kernel but with different interfaces.

- Some Linux distributions:
    1. Ubutu
    2. Slackware
    3. SuSE
    4. Debain
    5. RedHat
    6. Fedora
    7. Turbo Linux

- **Ubuntu**

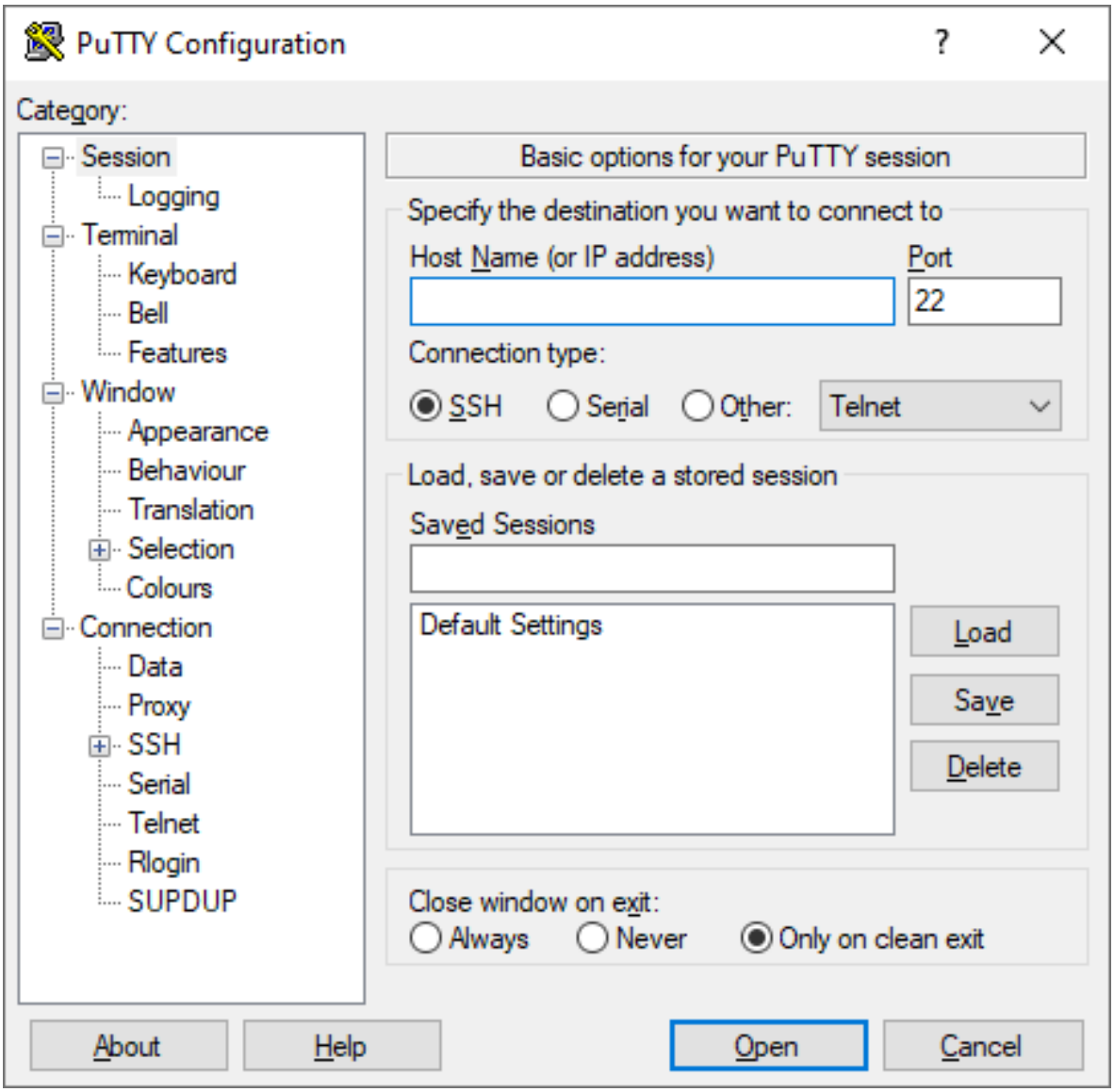
Canonical started Sending out free compact discs with Ubuntu Linux in 2004 and quickly became popular for home users(Many switching from Microsoft Windows)

Canonical wants Ubuntu to be an easy to use graphical Linux desktop without need to ever see a command line. Of course they also want to make a profit by spelling support for Ubuntu

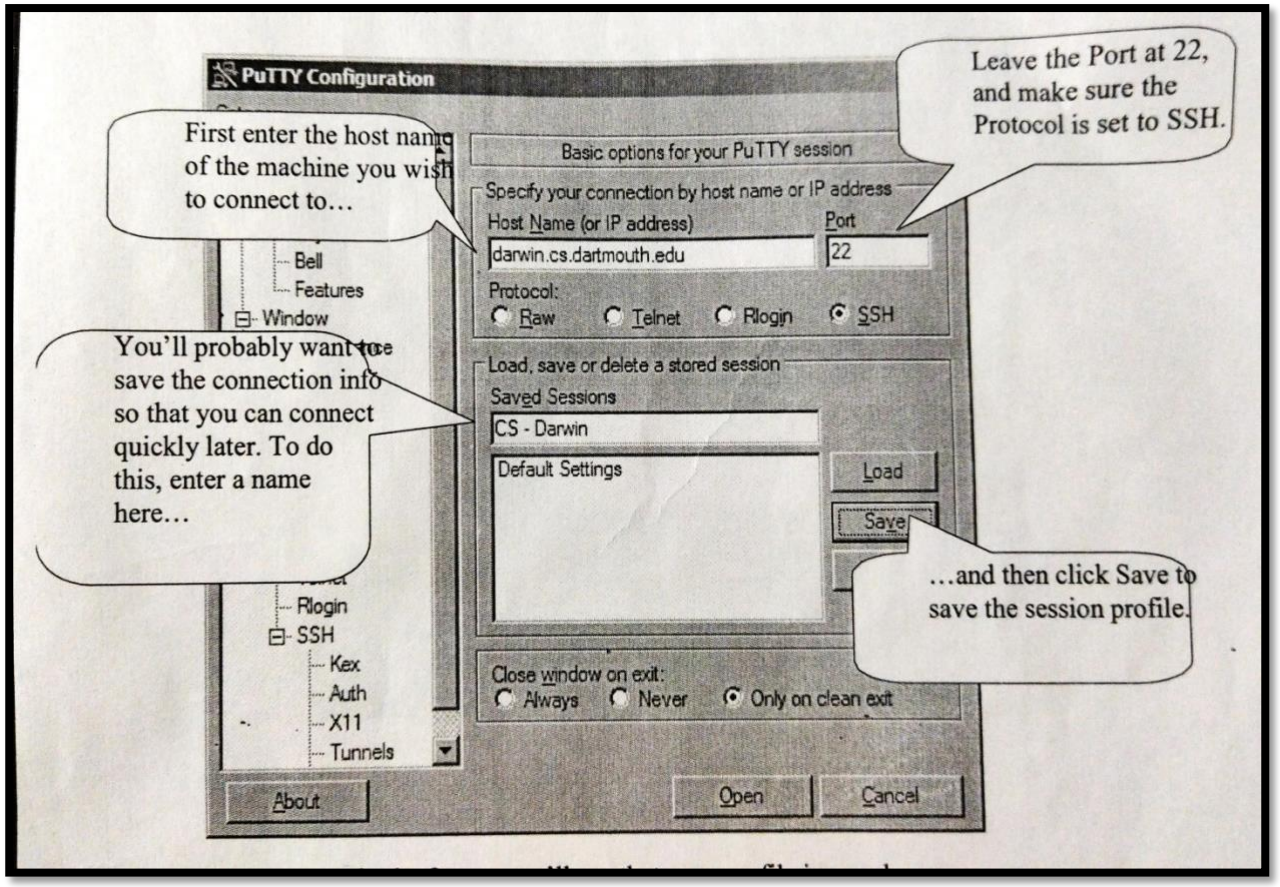
PuTTY

Start by downloading PuTTY (not a PuTTYTel, but PuTTY, as seen above.) You will want to save this file in an easy accessible location I personally recommended the desktop, so that you're just a double-click away.

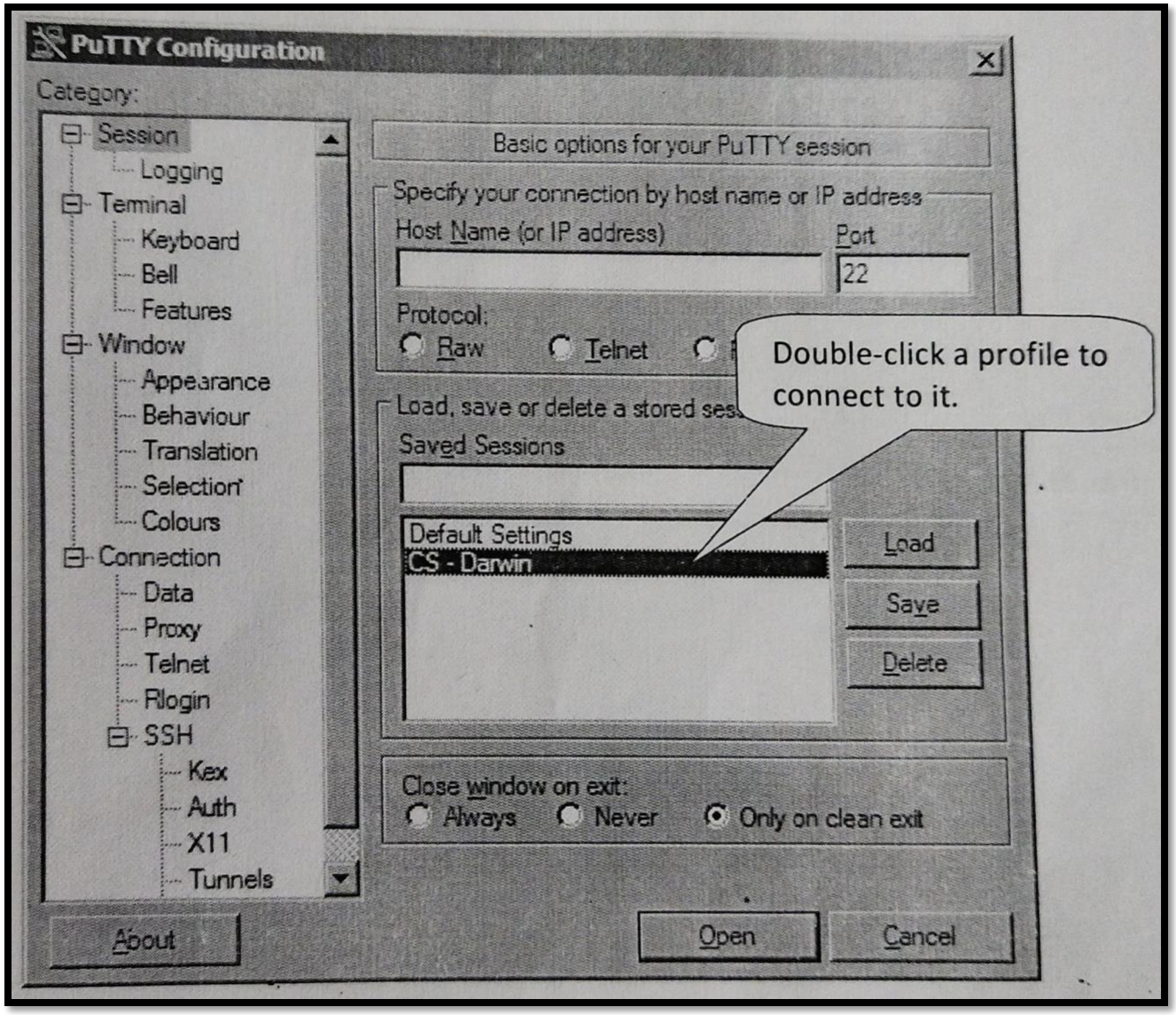
Now, let's begin! Double click the PuTTy icon, and the PuTTY configuration window will appear:



In this first portion of the tutorial, we run PuTTY as a simple SSH Client, without the use of X forwarding a graphical application (that Comes in the 2nd party of the tutorial) . For now, we'll just connect with a text consolm.

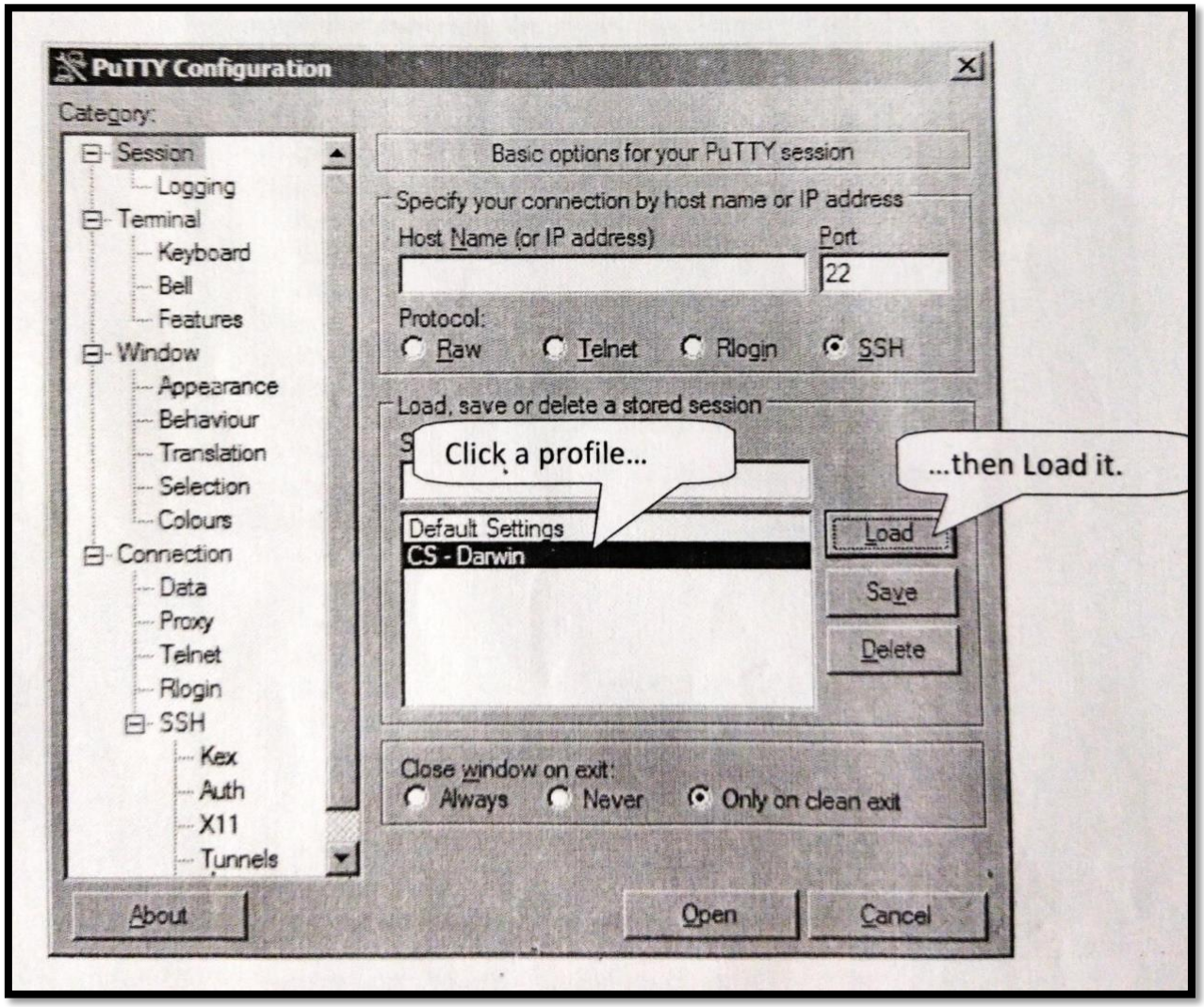


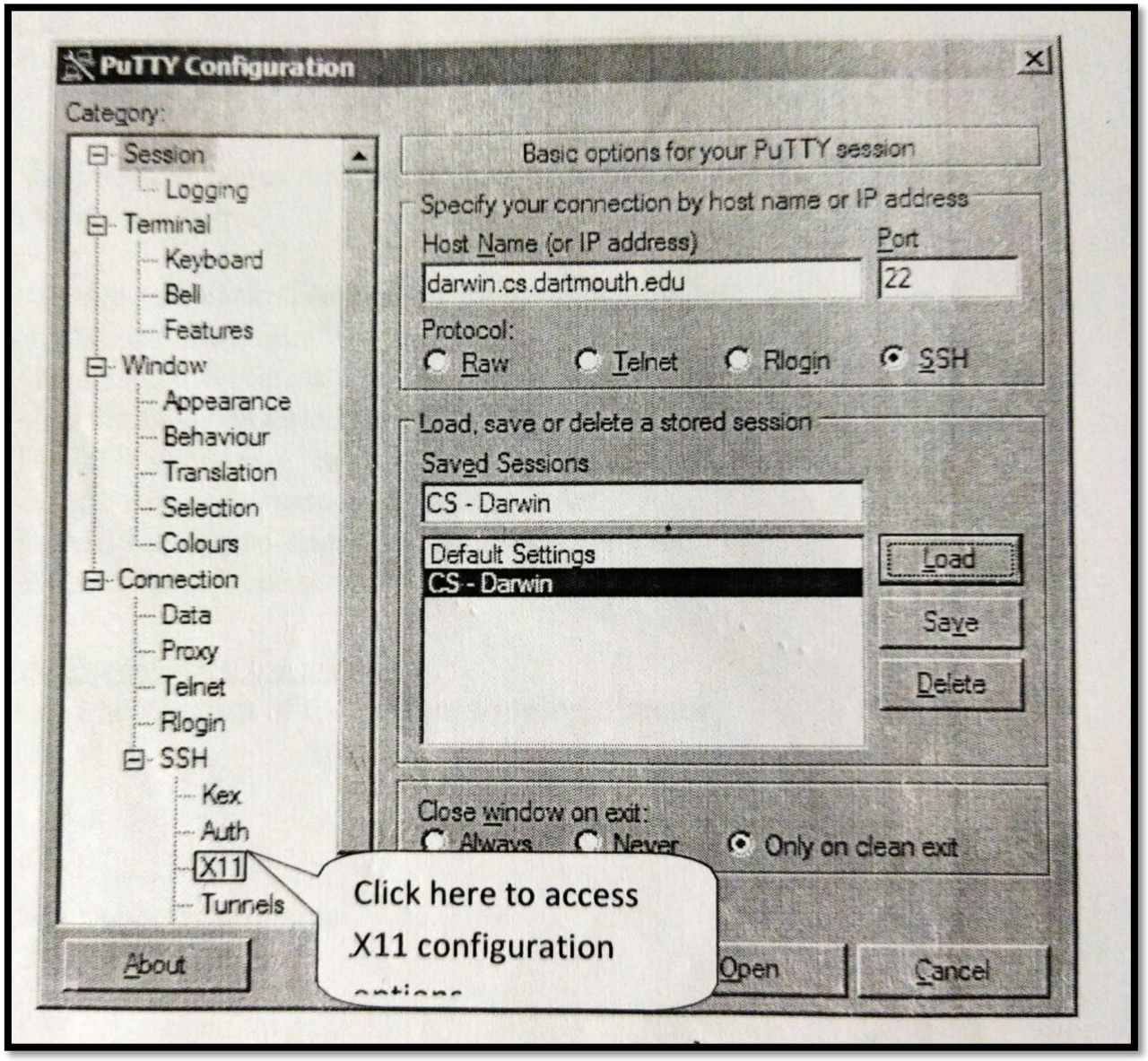
When you start PuTTY in the future, you'll see that your Profile is saved.



Now, let configure PuTTY. We want to enable X11 forwarding for our Profile, so that we can access graphical applications via SSH.

Start PuTTY. Select the profile you created earlier, and click “Load”.


PuTTY Configuration
Category:
Session
Logging
Terminal
Keyboard
Bell
Features
Window
Appearance
Behaviour
Translation
Selection
Colours
Connection
Data
Proxy
Telnet
Rlogin
SSH
Kex
Auth
X11
Tunnels
Basic options for your PuTTY session
Specify your connection by host name or IP address
Host Name (or IP address)
Port
22
Protocol:
Raw
Telnet
Rlogin
SSH
Load, save or delete a stored session
Click a profile...
...then Load it.
Default Settings
CS - Darwin
Load
Save
Delete
Close window on exit:
Always
Never
Only on clean exit
About
Open
Cancel



Now, under SSH in the tree in the left-hand side, click X11.

Experiment 1:

AIM: Usage of following commands Ls, pwd, tty,P cat, who, who am I, rm, mkdir , rmdir, touch, cd.

COMMANDS

LIST Command

It is used to list all the contents of current working directory

Syntax: $ls –opions <arguments>

If the command does not contain any argument means it is working in the current directory.

Options:

a- Used to list all the files in the current working directory.
c- List all the files column wise.
d- List all the directories.

m- list the files separated by commas.

p- list files include ‘’/’’ to all directories.

r- list the files in reverse alphabetical order.

f- list the files based on last modification

datex- list in column wise sorted order.

Present Working Directory Command:

To print the complete path of present working directory.

Syntax: $pwd

tty Command

It will display the terminal name.

Syntax: $tty

Who Command:

It is used to display who are the users connected to our computer currently.

Syntax: $who –options

Options:

H- Display the out with headers

b-Display the last booting date or time or when the when the system was system rebootedrecently.

Who am I Command:

Display the details of current working directory

Syntax: $who am I

Remove command:

rm command is used to remove the files permanently

Syntax: $rm <filename>

MKDIR command:

To create or make a new directory in a current directory.

Syntax: $mkdir <directoryname>

RMDIR command:

To remove a directory in the current directory & not the current directory itself.

Syntax: $rmdir<directory name>

Touch command:

Touch command is a standard command used in UNIX/Linux operating system which is used tocreate an empty file.

Syntax: $touch file_name

CD Command:

To change or move the directory to the mentioned directory.

Syntax: $CD<directory-name>.

Experiment 2:

AIM: Usage of following commands cal, cat, mv, cp, man, date

COMMANDS:

Calendar Command

This command is used to display the calendar of the year or particular month of the calendar year.

Syntax:

a. $cal<year>
b. $cal<moth><year>

Here the first syntax gives the entire calendar for given year& the second syntax gives the calendarof reserved moth of the year.

Cat Command:

Cat command is used to create file.

Syntax: $cat >filename

Cat command is used to display file.

Syntax : $cat filename

Cat command is used to copy content of one file to another file.

Syntax: $cat filename1>filename2

Cat command is used to append one file to another file.

Syntax: $cat filename1>>filename2

MOVE Command

To completely move the contents from source file to destination file and remove the source file.

Syntax: $cp<sourcefilename><destinationfile>

Man command:
It help us to know about a particular command and its option and working. It is like " help " command in windows.

Syntax : $man<command name>

Date Command:

This command is used to display current date and time.

Syntax: $date

Sat Apr 20 04:40:10 IST 2021

$date '+DATE:%d-%m-%y %n TIME: %H:%M:%S'

DATE:17-06-

2021

TIME:10:55:25

Experiment 3:

AIM: Usage of following commands chmod, grep, tput, bc

COMMANDS

CHMOD command

Chmod command can be used to assign or remove permissions.

Syntax : Chmod [reference][operator][mode] file

Reference	Class	Description
u	owner	file's owner
g	group	users who are members of the file's group
o	others	users who are neither the file's owner nor members of the file's group
a	all	All three of the above, same as ugo

Operator	Description
+	Adds the specified modes to the specified classes
-	Removes the specified modes from the specified classes
=	The modes specified are to be made the exact modes for the

specifiedclasses

GREP command

Grep stands for ‘’globally search a regular expression and print it”. This command searches the specified input fully for a match with supplied pattern and dispalys it.

grep [options] pattern [files]

-c : This prints only a count of the lines that match a pattern

-h : Display the matched lines, but do not display the filenames.

-i : Ignores, case for matching

-l : Displays list of a filenames only.

-n : Display the matched lines and their line numbers.

-v : This prints out all the lines that do not matches the pattern

Example: $grep -c "unix" geekfile.txt

HEAD: It is used to display the top ten lines of the file.

Syntax: $head <filename>

TAIL: this command is used to display last 10 lines of files

Sort: this command is used to sort the data in order.

Syntax: $sort<filename>

Page command: This command is used to display the file page by page a screenfull of information, after which the page command displays a prompt and passes for the users to strike the enter key tocontinue scrolling.

Syntax: $pg <filename>

Pipe command:

It is a mechanism by which the output of one command can be channelled into the input of anothercommand.

Syntax: $who|wc -l

TR:

The tr filter is used to translate one set of characters from the standard inputs to another.

Syntax: $put "[a-z] "[A-Z]"

Clear:

Clear command is used to clear the screen

Syntax: $clear

Binary calculator:

bc command is used for command line calculator. It is similar to basic calculator by using which wecan do basic mathematical calculations such as +,-,*,/,%.

Syntax: $bc operation

Λd

$

Experiment 4:

AIM: Usage of vi editor commands

VI editor commands:

The vi editor is a visual edit used to create and edit text,files,documents and programs. It display thecontent of files on the screen and allows a user to add, delete or change part of text.

There are three modes available in VI editors, they are

1. Command mode
2. Input (or) insert mode

Starting Vi:

The Vi editor is invoked by giving the following commands in unix prompt.

Syntax: $vi<filename>

$vi

This command would open a display screen with 25 lines and with tilt(~) symbol at the start of eachline.

Insert or input mode commands:

ESC a command

This command is used to move to the insert mode and start to append after the current

character.Syntax: <ESC> a

ESC A command

This command is used to append the file, but this command append at the end of the current

line.Syntax: <ESC> A

ESC i command:

This command is used to insert the text before the current cursor

position.**Syntax:** <ESC> i

ESC I command:

This command is used to insert at the beginning of the cuurent

line.**Syntax:** <ESC> I

ESC o command:

This command is used to insert the blank line below the current line & allowinsertion of the content.

Syntax: <ESC> o

ESC O command:

This command is used to insert blank line above and allow insertion of the

contents.**Syntax:** <ESC> O

ESC r command:

This command is used to replace a particular character with a given

character.**Syntax:** <ESC> r

Cursor movements in vi:

ESC h:

This command is used to move to the left of the text. It can be used to move character by characteror number of charecters.

Syntax: <ESC>h- to move one character to left

<ESC>nh-to move n charecters to left.

ESC l:

This command is used to move to the right of the cursor. It can be used to move character bycharacter or number of charecters.

Syntax: <ESC>l- to move one character to left

<ESC>nl-to move n charecters to left.

ESC j:

This command is used to move down a single line or number of

lines.**Syntax:** <ESC>j- single down movement

<ESC>nj-n times down movement

ESC k:

This command is used to move up a single line or number of

lines.**Syntax:** <ESC>k- single up movement

<ESC>nj-n times up movement

ESC +: This command is used to move to the beginning of the

next line.

Syntax: <ESC>+

ESC -: This command is used to move to the beginning of the previous

line. **Syntax**: <ESC>-

ESC 0: this command is used to bring the cursor to the beginning of the same current line.

Syntax: <ESC>0

ESC $: this command is used to bring the cursor to the end of the same current

line.**Syntax:** <ESC>$

ESC Λ:

This command is used to move to the first character of the first

lines.**Syntax:** <ESC>^

ESC b command:

This command is used to move back to the previous word or a number of words.

Syntax: <ESC>b <ESC>nb

ESC e command:

This command is used to move to end of the current

word**Syntax:** <ESC>e

ESC w:

This command is used to move to the beginning of the current

word.**Syntax:** <ESC>w

DELETING TEXT FROM Vi:

ESC x command:

To delete a character to the right of the current cursor

position.**Syntax:** <ESC>x

ESC X command:

To delete a character to the left of the current cursor

position.**Syntax**: <ESC>X

ESC dw command:

This command is used to delete a single word or number of words to the right of the current cursorposition.

Syntax: <ESC>dw <ESC>ndw

ESC db command : this command is used to delete a single word or number of words to the left ofthe current cursor position.

Syntax: <ESC>db <ESC>ndb

ESC dd command:

This command is used to delete current line or number of lines below the current

line.**Syntax:** <ESC>dd <ESC>ndd

ESC d$:

This command is used to delete the text from current cursor position to the last character of currentline.

Syntax: <ESC>d$

SAVING and QUITTING from vi

ESC w command:

To save the given text present in the file.

Syntax: <ESC>:w

ESC q! command:

To quit the given text with out

saving.**Syntax:** <ESC>:q!

ESC wq command:

This command quits the vi editor after saving the text in the mentioned file.

ESC q command:

This command would quit the window but it would ask for again to save the file.

Syntax: <ESC>:q

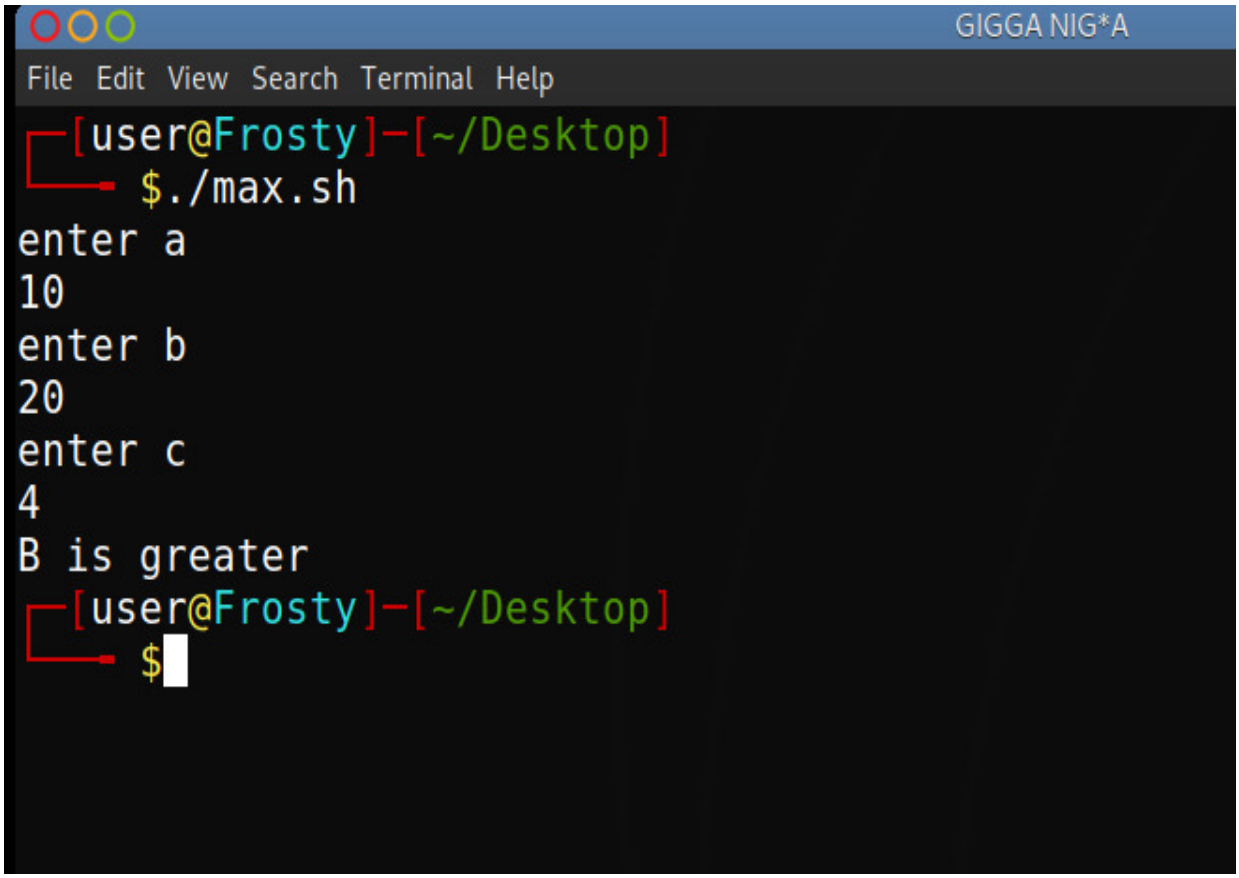
Experiment 5:

AIM: Shell script to find maximum of three numbers

Program:

```
echo "enter a"
read a
echo "enter b"
read b
echo "enter c"
read c
if [ $a -gt $b -a $a -gt $c ]; then
   echo "A is greater"
elif [ $b -gt $a -a $b -gt $c ]; then
   echo "B is greater"
else
   echo "C is greater"
fi
```

OUTPUT


GIGGA NIG*A
File Edit View Search Terminal Help
[user@Frosty]-[~/Desktop]
$./max.sh
enter a
10
enter b
20
enter c
4
B is greater
[user@Frosty]-[~/Desktop]
$

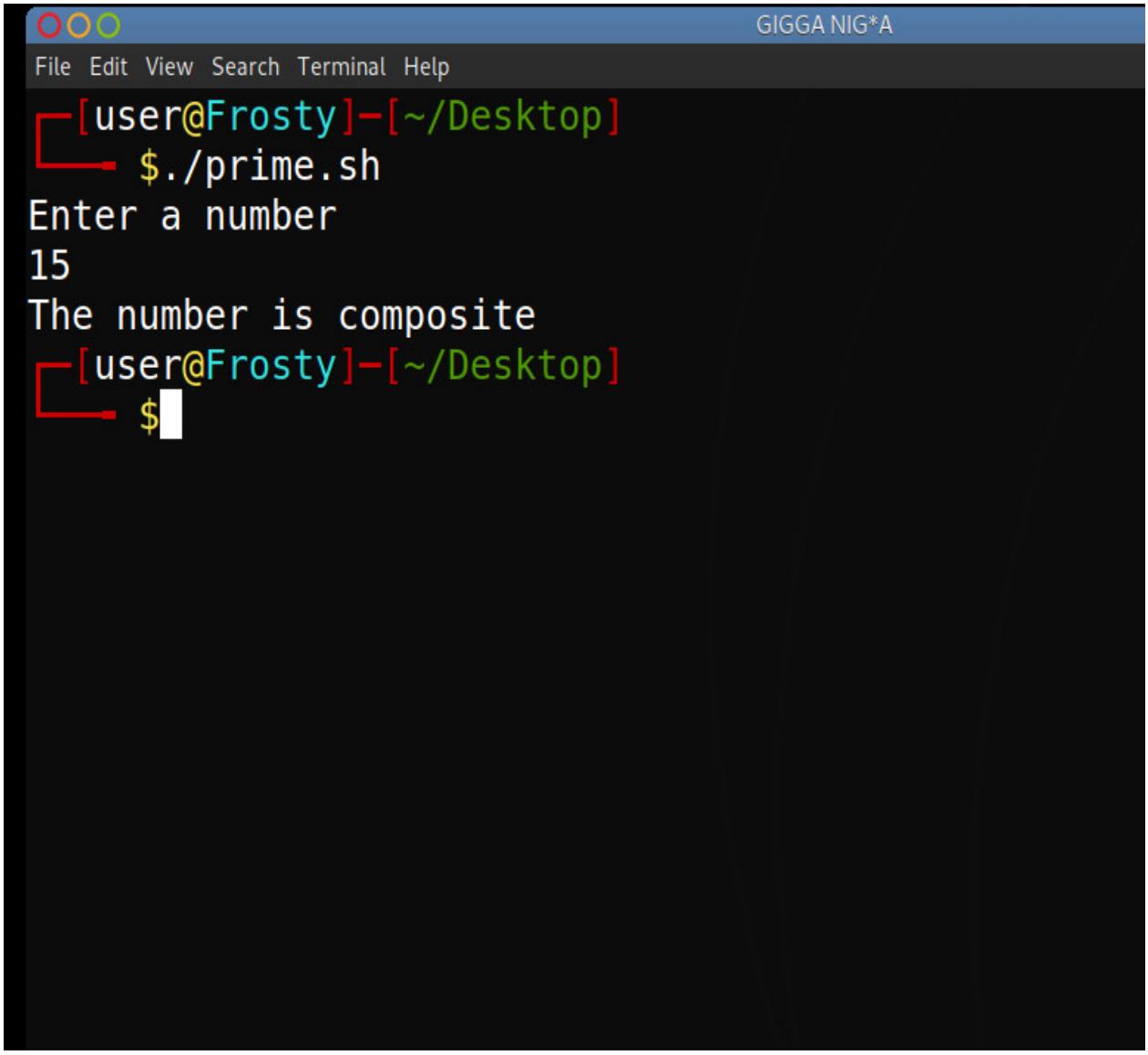
Experiment 6:

AIM: Shell scipt to find the given number is prime or not

PROGRAM

```
echo "Enter a number"
read num
i=2
f=0
while [ $i -le `expr $num / 2` ]
do
  if [ `expr $num % $i` -eq 0 ]
  then
    f=1
    break
  fi
  i=`expr $i + 1`
done
if [ $f -eq 1 ]
then
  echo "The number is composite"
else
  echo "The number is Prime"
fi
```

OUTPUT:


GIGGA NIG*A
File Edit View Search Terminal Help
[user@Frosty]-[~/Desktop]
$./prime.sh
Enter a number
15
The number is composite
[user@Frosty]-[~/Desktop]
$

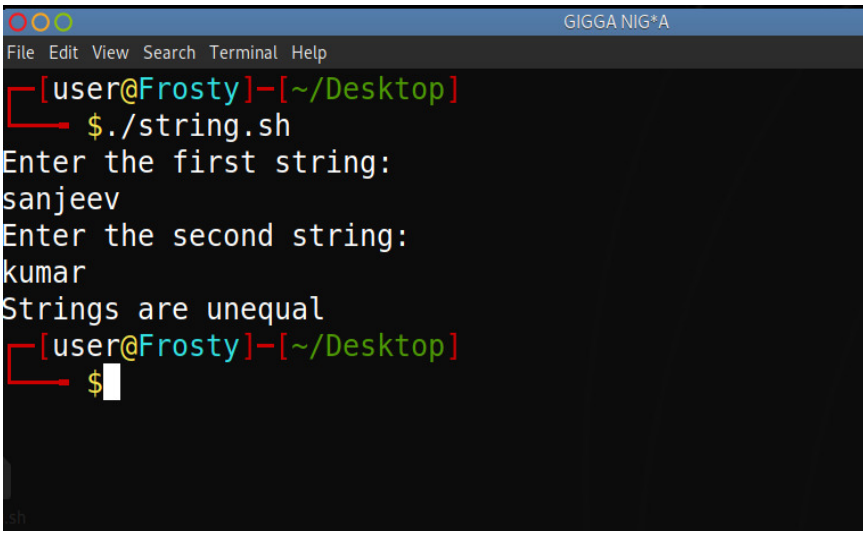
Experiment 7

AIM: Shell script to compare to strings

PROGRAM

```
#!/bin/bash

echo "Enter the first string:"
read str1

echo "Enter the second string:"
read str2

if [ "$str1" = "$str2" ]; then
    echo "Strings are equal"
else
    echo "Strings are unequal"
fi
```

OUTPUT



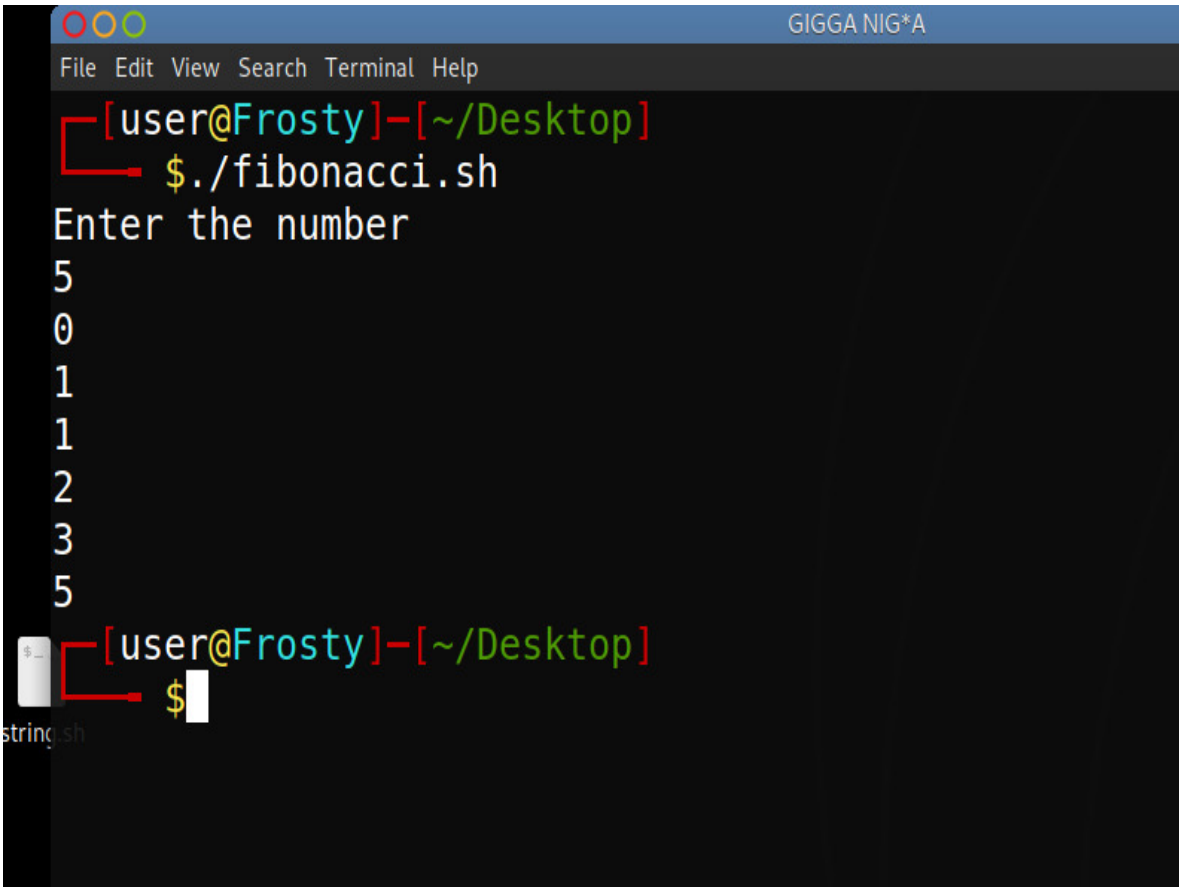
Experiment 8:

AIM: Write a shell script to display Fibonacci series

PROGRAM:

```
echo "Enter the number"
read n
a=-1
b=1
i=0
while [ $i -le $n ]
do
    t=$(expr $a + $b)
    echo $t
    a=$b
    b=$t
    i=$(expr $i + 1)
done
```

OUTPUT


GIGGA NIG*A
File Edit View Search Terminal Help
[user@Frosty]-[~/Desktop]
$./fibonacci.sh
Enter the number
5
0
1
1
2
3
5
[user@Frosty]-[~/Desktop]
$

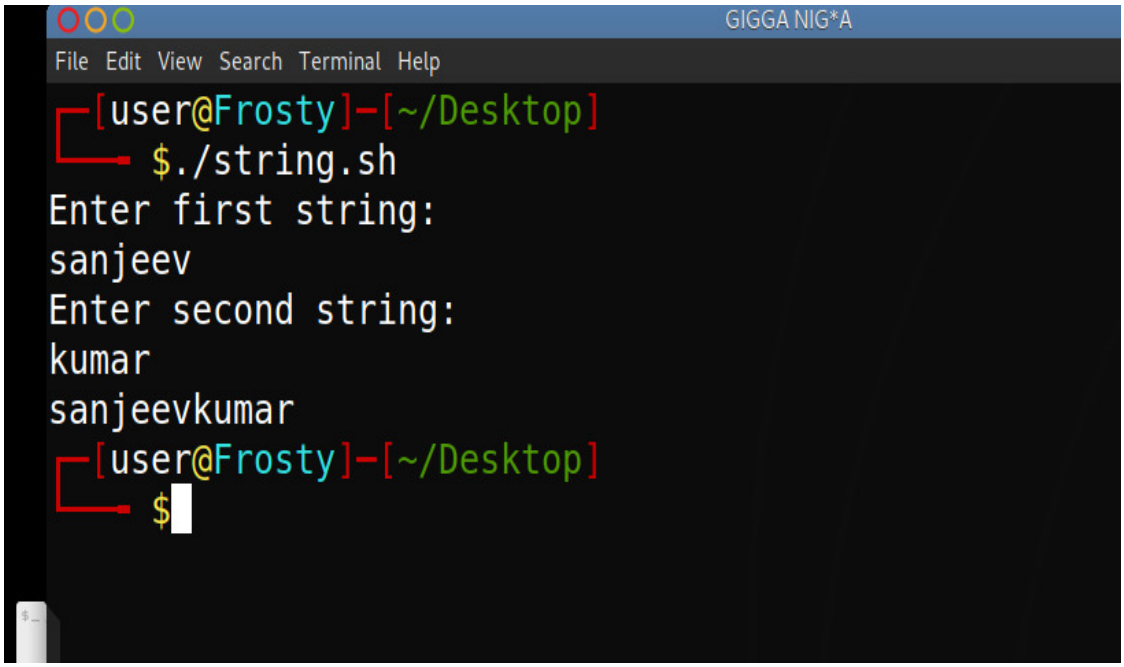
Experiment 9:

AIM: Write a shell script for concatenation of two strings.

PROGRAM:

```
echo "Enter first string:"
read str1
echo "Enter second string:"
read str2
str1="$str1$str2"
echo $str1
```

OUTPUT



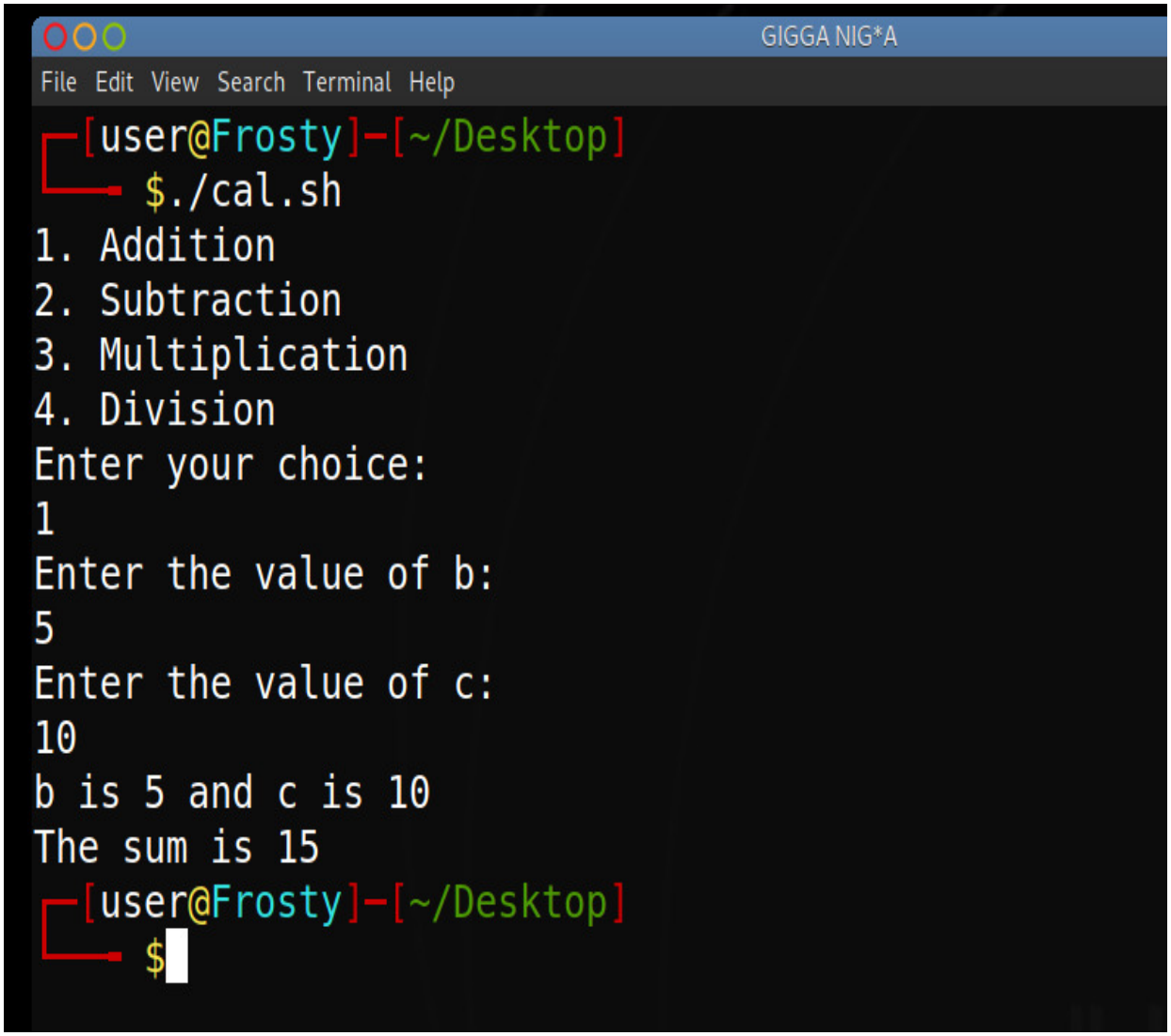
Experiment 10

AIM: Write a shell script to display arithmetic operations

PROGRAM

```
#!/bin/bash

echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
echo "Enter your choice: "
read a
echo "Enter the value of b: "
read b
echo "Enter the value of c: "
read c
echo "b is $b and c is $c"

case $a in
    1)
        d=$(expr $b + $c)
        echo "The sum is $d"
        ;;
    2)
        d=$(expr $b - $c)
        echo "The subtraction is $d"
        ;;
    3)
        d=$(expr $b \* $c)
        echo "The multiplication is $d"
        ;;
    4)
        d=$(expr $b / $c)
        echo "The division is $d"
        ;;
    *)
        echo "Invalid option"
        ;;
esac
```

OUTPUT


GIGGA NIG*A
File Edit View Search Terminal Help
[user@Frosty]-[~/Desktop]
$./cal.sh
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter your choice:
1
Enter the value of b:
5
Enter the value of c:
10
b is 5 and c is 10
The sum is 15
[user@Frosty]-[~/Desktop]
$

Experiment 11

AIM: Implementation of Fork() System call

Description: A fork system call creates a child process that is a clone of the parent. The child process may execute a different program in its context with a separate exec() system call. Parent may want to wait for children to finish by calling wait function. The wait() function suspends execution of its calling process until status information is available for a terminated child process, or a signal is received. The exit () function is used to terminate the calling process immediately.

Prpgram:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int p;
    int status;
    p = fork();
    switch(p) {
        case -1:
            perror("fork");
            exit(1);
        case 0:
            printf("Child process\n");
            break;
        default:
            wait(&status);
            printf("Parent process\n")
;
            break;
    }
    return 0;
}
```

OUTPUT:


23
input
Child process
Parent process
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 12

Aim: Simulate FCFS scheduling algorithm

Description

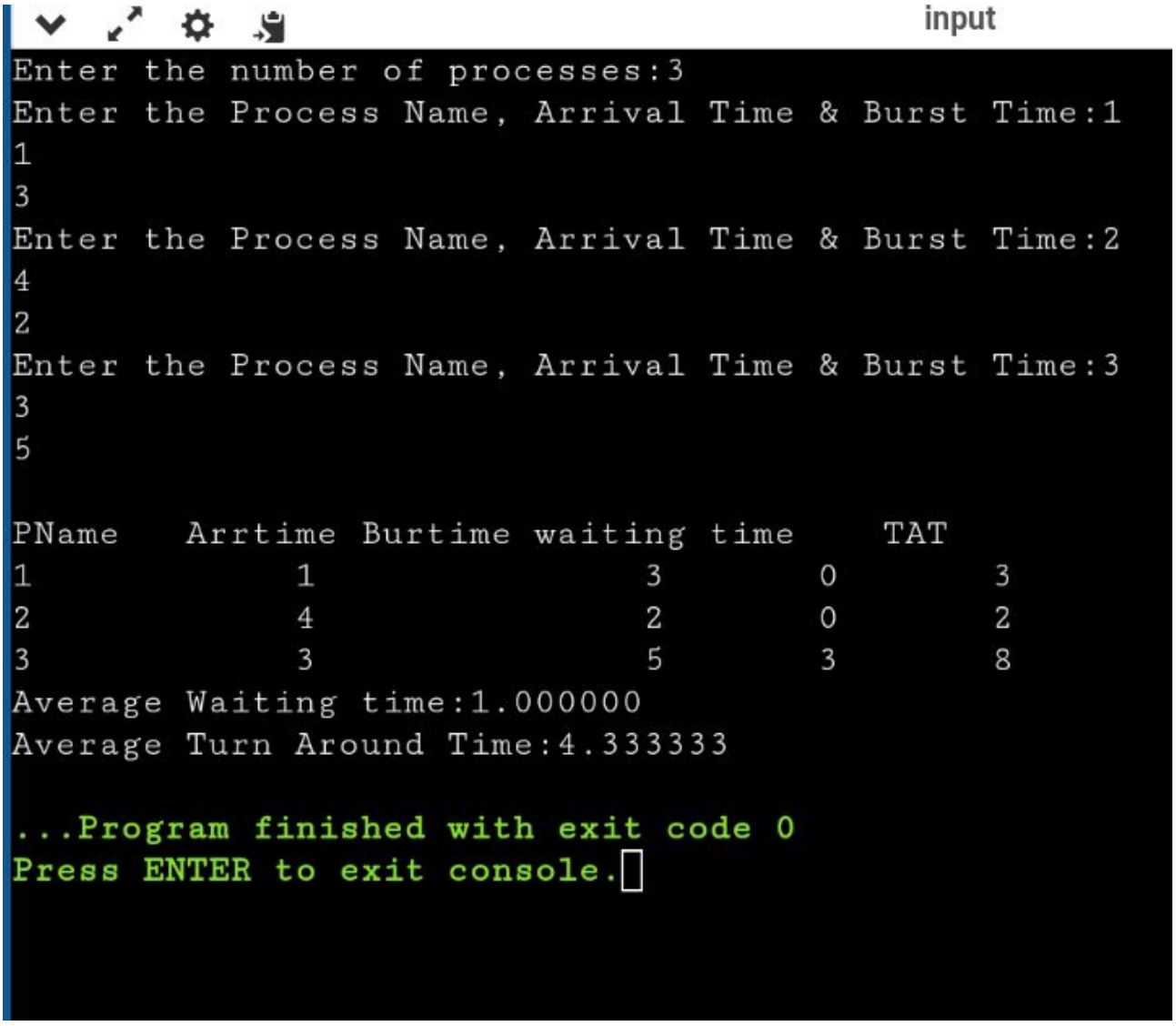
Simplest CPU-scheduling algorithm is the first come, first-served (FCFS) scheduling algorithm. The process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, it is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue.

Program:

```
#include<stdio.h>

void main()

{

int  pn[10];

int arr[10],bur[10],star[10],finish[10],tat[10],wt[10],i,n;

int totwt=0,tottat=0;


printf("Enter the number of processes:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter the Process Name, Arrival Time & Burst Time:");

scanf("%d %d%d",&pn[i],&arr[i],&bur[i]);

}

for(i=0;i<n;i++)

{

if(i==0)

{
```

```
star[i]=arr[i];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
else
{
star[i]=finish[i-1];
wt[i]=star[i]-arr[i];
finish[i]=star[i]+bur[i];
tat[i]=finish[i]-arr[i];
}
}
printf("\nPName\tArrtime\tBurtime\twaiting time\tTAT ");
for(i=0;i<n;i++)
{
printf("\n%d\t%6d\t\t%6d\t%6d\t%6d\t",pn[i],arr[i],bur[i],wt[i],tat[i]);
totwt+=wt[i];
tottat+=tat[i];
}

printf("\nAverage Waiting time:%f",(float)totwt/n);
printf("\nAverage Turn Around Time:%f",(float)tottat/n);
}
```

OUTPUT:


input
Enter the number of processes:3
Enter the Process Name, Arrival Time & Burst Time:1
1
3
Enter the Process Name, Arrival Time & Burst Time:2
4
2
Enter the Process Name, Arrival Time & Burst Time:3
3
5
PName Arrtime Burtime waiting time TAT
1 1 3 0 3
2 4 2 0 2
3 3 5 3 8
Average Waiting time:1.000000
Average Turn Around Time:4.333333
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 13:

Aim: Simulate SJF scheduling algorithm

Description:-

A different approach to CPU scheduling is the shortest-job first (SJF) scheduling algorithm. This algorithm associates with each process the length of the next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. if two processes have the same length next CPU burst, FCFS scheduling is used to break the tie.

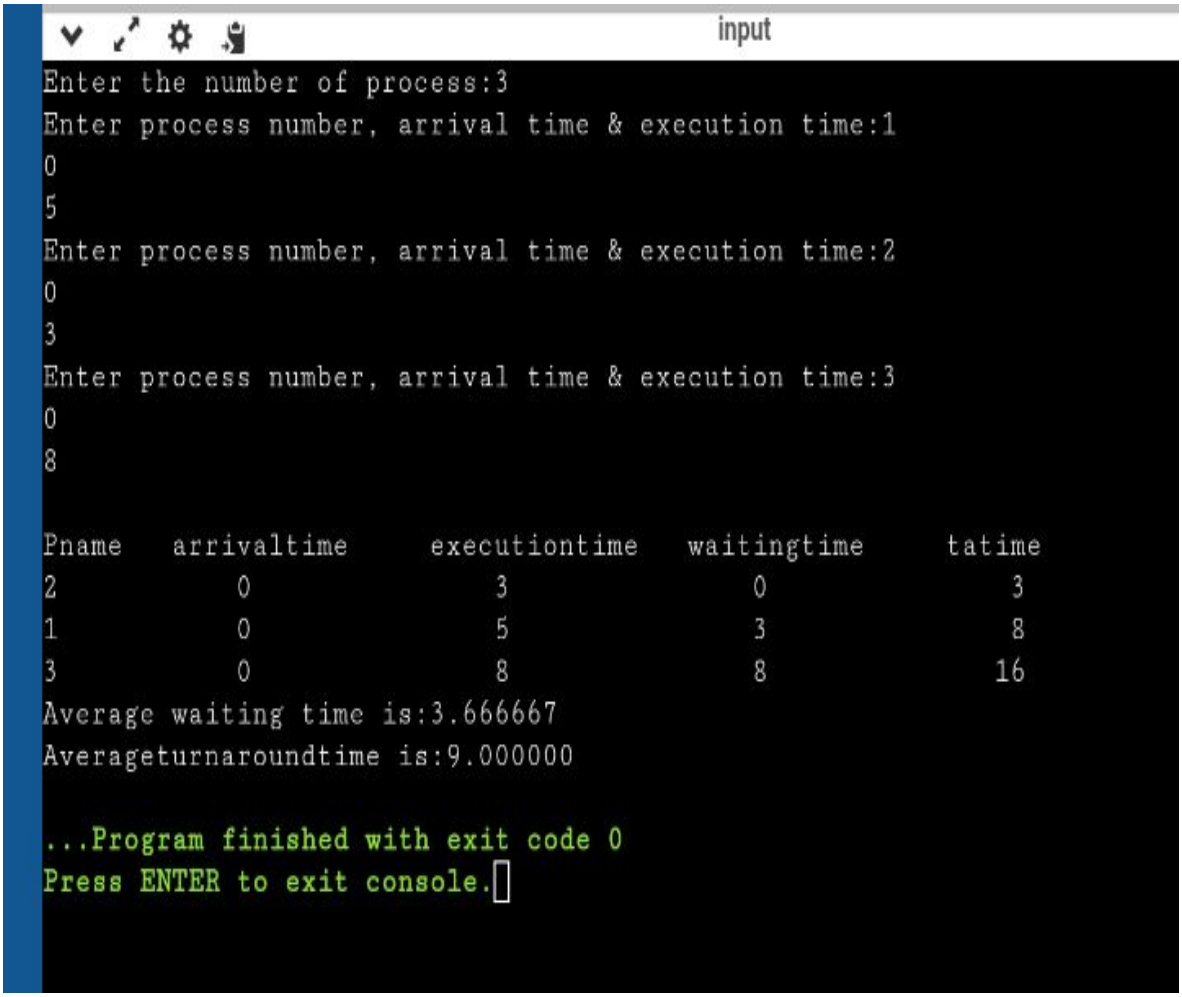
Program

```
#include<stdio.h>

#include<string.h>

void main()

{

int pn[10],et[20],at[10],n,i,j,temp,st[10],ft[10],wt[10],ta[10];

int totwt=0,totta=0;

float awt,ata;

printf("Enter the number of process:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter process number, arrival time & execution time:");

scanf("%d%d%d",&pn[i],&at[i],&et[i]);

}

for(i=0;i<n;i++)

for(j=0;j<n;j++)

{

if(et[i]<et[j])
```

```
{
temp=at[i];
at[i]=at[j];
at[j]=temp;
temp=et[i];
et[i]=et[j];
et[j]=temp;
temp=pn[i];
pn[i]=pn[j];
pn[j]=temp;
}
}
for(i=0;i<n;i++)
{
if(i==0)
{
st[i]=at[i];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
else
{
st[i]=ft[i-1];
wt[i]=st[i]-at[i];
```

```
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
}
for(i=0;i<n;i++)
{
totwt+=wt[i];
totta+=ta[i];
}

awt=(float)totwt/n;
ata=(float)totta/n;
printf("\nPname\tarrivaltime\texecutiontime\twaitingtime\ttatime");
for(i=0;i<n;i++)
printf("\n%d\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],wt[i],ta[i]);
printf("\nAverage waiting time is:%f",awt);
printf("\nAverageturnaroundtime is:%f",ata);
}
```

OUTPUT


input
Enter the number of process:3
Enter process number, arrival time & execution time:1
0
5
Enter process number, arrival time & execution time:2
0
3
Enter process number, arrival time & execution time:3
0
8
Pname arrivaltime executiontime waitingtime tatime
2 0 3 0 3
1 0 5 3 8
3 0 8 8 16
Average waiting time is:3.666667
Averageturnaroundtime is:9.000000
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 14

Aim: Simulate Round Robin scheduling algorithm

DESCRIPTION:-

The round-robin scheduling algorithm is designed especially for time-sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time called a time quantum is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

To implement RR scheduling we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum and dispatches the process.

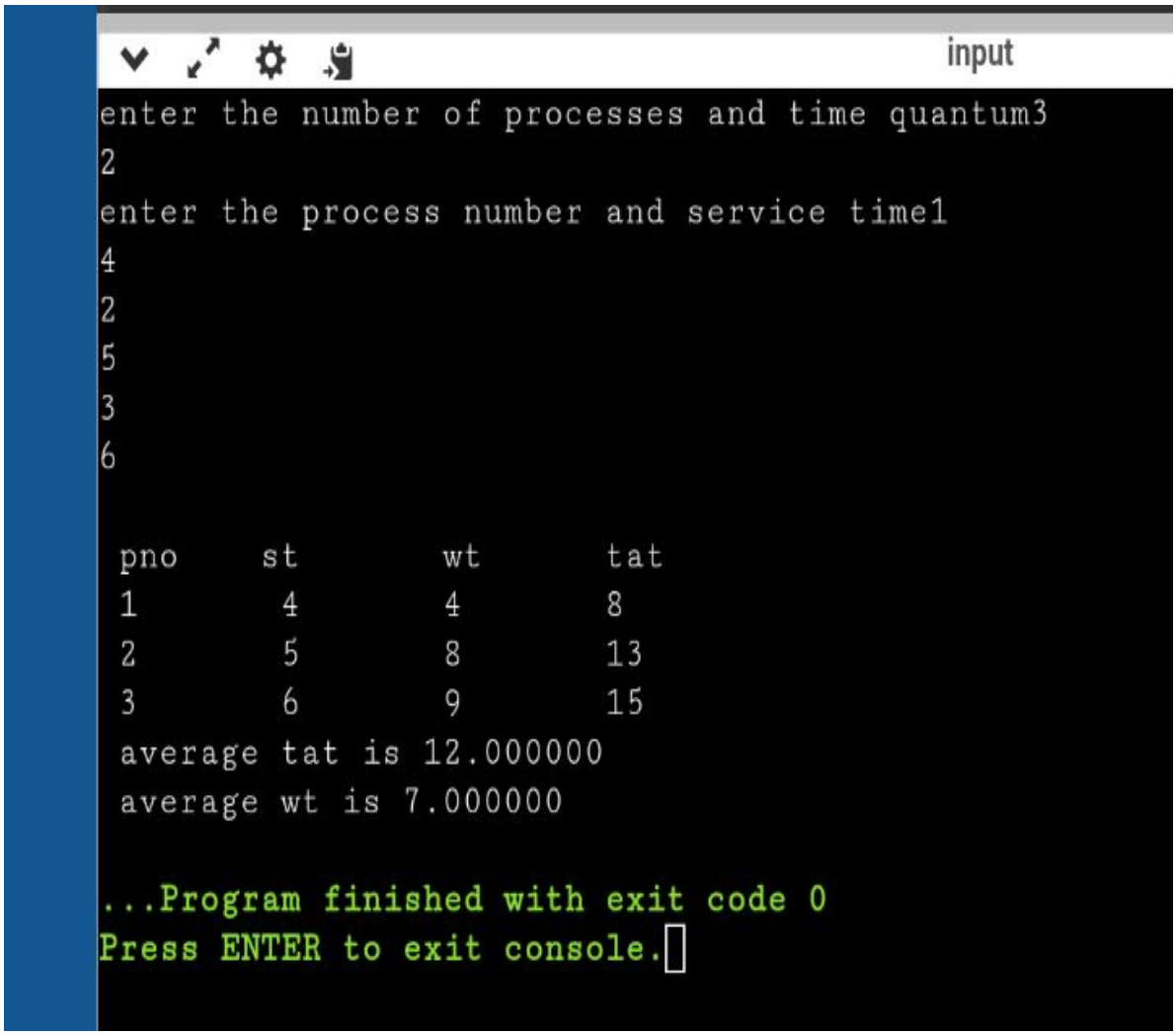
PROGRAM:

```
#include<stdio.h>

void main()

{

int i,j,n,tq,time=0,p[10],st[10],b[10],str[10],ft[10],tat[10],wt[10];

float avg1=0,avg2=0;

printf("enter the number of processes and time quantum");

scanf("%d%d",&n,&tq);

printf("enter the process number and service time");

for(i=0;i<n;i++)

{

scanf("%d%d",&p[i],&st[i]);

}


for(i=0;i<n;i++)
```

```
{
b[i]=st[i];
}

for(i=0;i<n;i++)
{
while(b[i]!=0)
{
for(j=0;j<n;j++)
{
if(b[j]==0)
{
continue;
}
if(b[j]>tq)
{
b[j]=b[j]-tq;
time=time+tq;
}
else
{
time=time+b[j];
b[j]=0;
ft[j]=time;
}
```

```
}
}
}
for(i=0;i<n;i++)
{
tat[i]=ft[i];
wt[i]=ft[i]-st[i];
avg1=avg1+tat[i];
avg2=avg2+wt[i];
}
printf("\n pno\tst\t    wt \t tat\t ");
for(i=0;i<n;i++)
{
printf("\n %d\t  %d\t  %d\t  %d\t ",p[i],st[i],wt[i],tat[i]);
}
printf("\n average tat is %f\n average wt is %f",(avg1/n),(avg2/n));
}
```

OUTPUT;


input
enter the number of processes and time quantum3
2
enter the process number and service time1
4
2
5
3
6
pno st wt tat
1 4 4 8
2 5 8 13
3 6 9 15
average tat is 12.000000
average wt is 7.000000
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 15

AIM: Simulate Priority scheduling scheduling algorithm

DESCRIPTION

Priority scheduling is a method of scheduling processes based on priority. In this method, the scheduler chooses the tasks to work as per the priority, which is different from other types of scheduling, for example, a simple round robin.

Priority scheduling involves priority assignment to every process, and processes with higher priorities are carried out first, whereas tasks with equal priorities are carried out on a first-come-first-served (fcfs) or round robin basis.

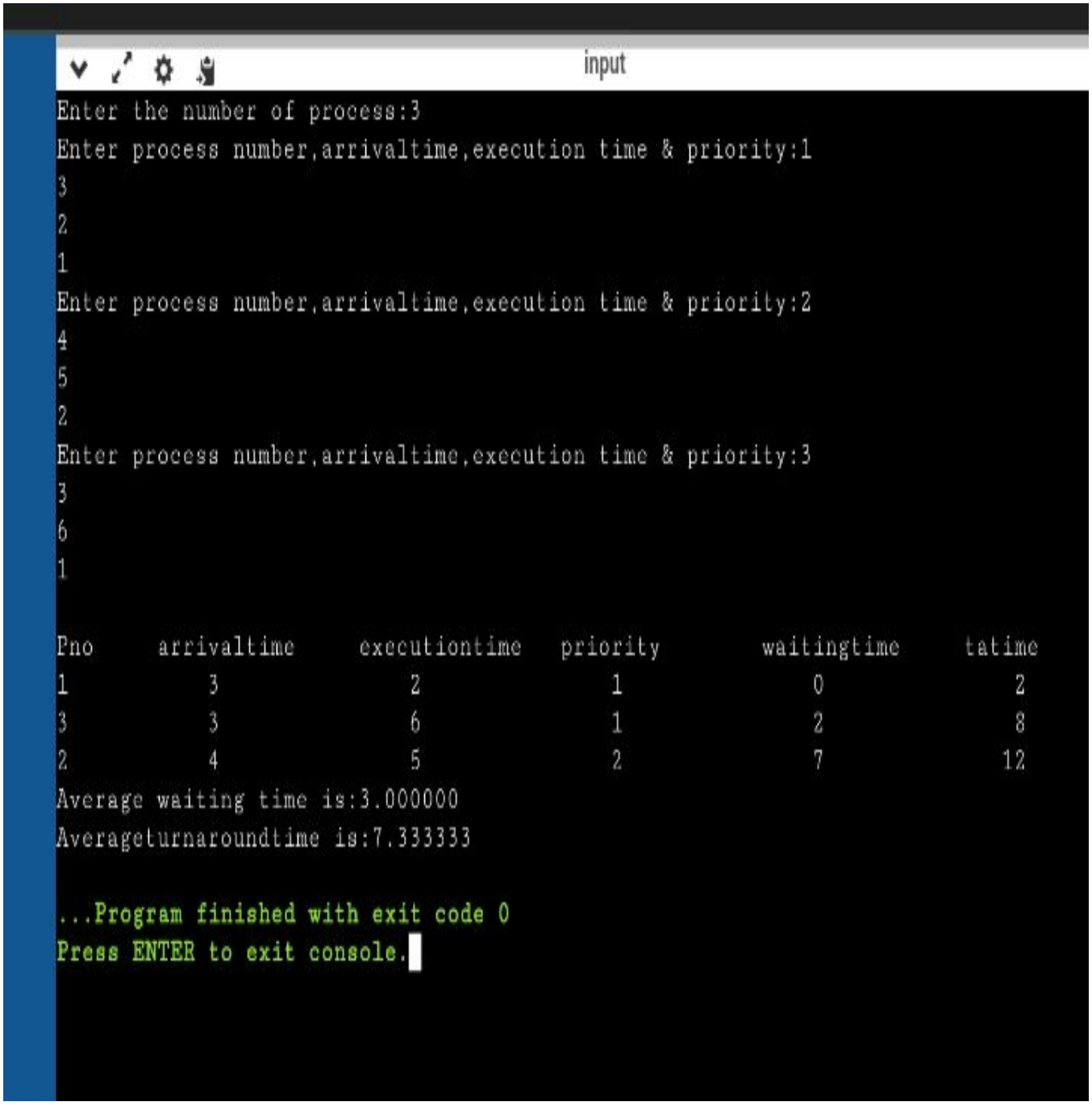
PROGRAM:

```
#include<stdio.h>

#include<string.h>

void main()

{

int pn[10],et[20],at[10],n,i,j,temp,p[10],st[10],ft[10],wt[10],ta[10];

int totwt=0,totta=0;

float awt,ata;


printf("Enter the number of process:");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter process number,arrivaltime,execution time & priority:");
```

```
scanf("%d%d%d%d",&pn[i],&at[i],&et[i],&p[i]);
}
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
if(p[i]<p[j])
{
temp=p[i];
p[i]=p[j];
p[j]=temp;
temp=at[i];
at[i]=at[j];
at[j]=temp;
temp=et[i];
et[i]=et[j];
et[j]=temp;
temp=pn[i];
pn[i]=pn[j];
pn[j]=temp;
}
}
for(i=0;i<n;i++)
{
if(i==0)
{
```

```
st[i]=at[i];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
else
{
st[i]=ft[i-1];
wt[i]=st[i]-at[i];
ft[i]=st[i]+et[i];
ta[i]=ft[i]-at[i];
}
totwt+=wt[i];
totta+=ta[i];
}
awt=(float)totwt/n;
ata=(float)totta/n;
printf("\nPno\tarrivaltime\texecutiontime\tpriority\twaitingtime\ttatime");
for(i=0;i<n;i++)
printf("\n%d\t%5d\t\t%5d\t\t%5d\t\t%5d\t\t%5d",pn[i],at[i],et[i],p[i],wt[i],ta[i]);
printf("\nAverage waiting time is:%f",awt);
printf("\nAverageturnaroundtime is:%f",ata);
}
```

OUTPUT


input
Enter the number of process:3
Enter process number,arrivaltime,execution time & priority:1
3
2
1
Enter process number,arrivaltime,execution time & priority:2
4
5
2
Enter process number,arrivaltime,execution time & priority:3
3
6
1
Pno arrivaltime executiontime priority waitingtime tatime
1 3 2 1 0 2
3 3 6 1 2 8
2 4 5 2 7 12
Average waiting time is:3.000000
Averageturnaroundtime is:7.333333
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 16

AIM: Simulate First in first out page replacement algorithm

DESCRIPTION:

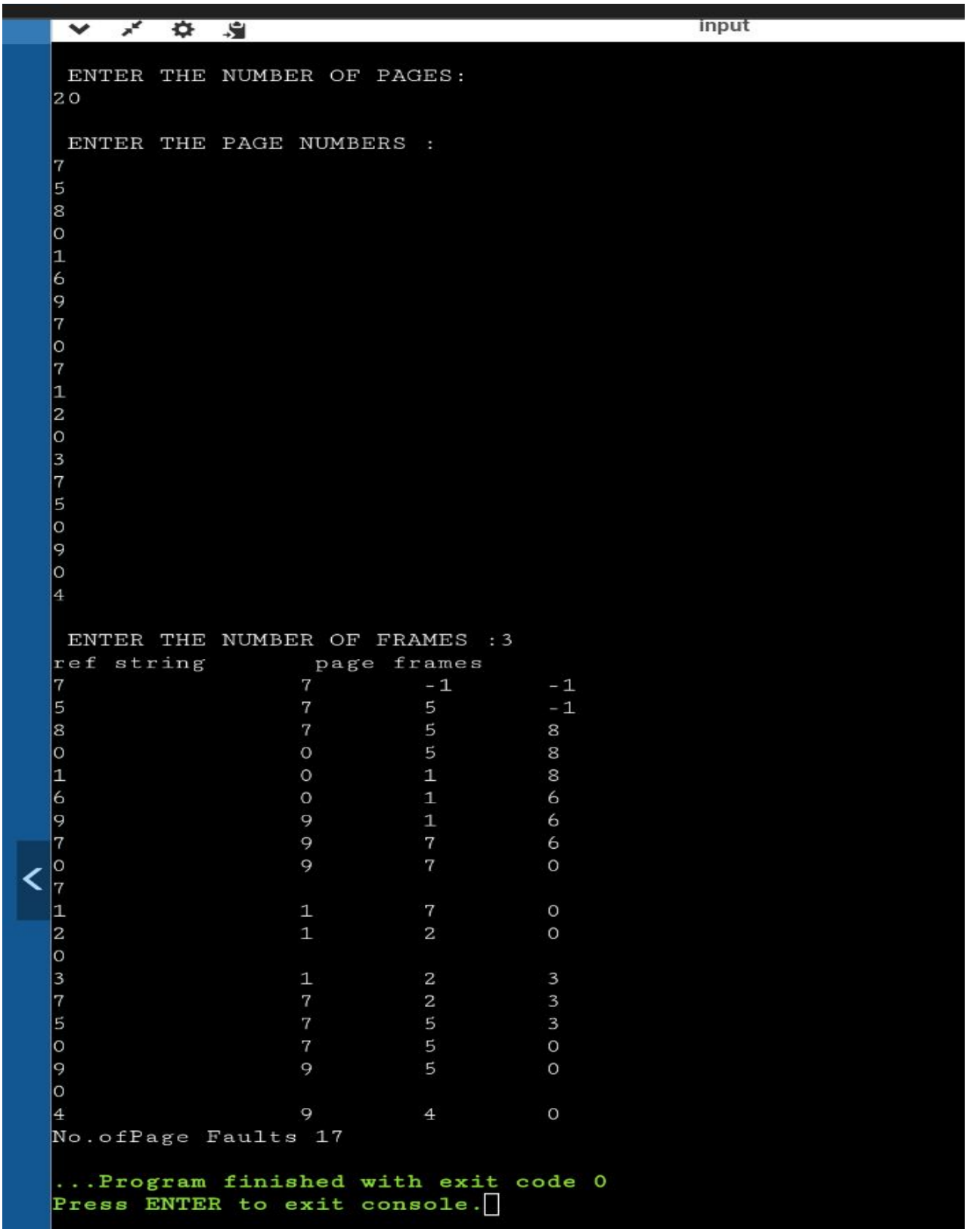
This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

PROGRAM

```
#include<stdio.h>

void main()

{

   int i,j,n,a[50],frame[10],no,k,avail,count=0;

printf("\n ENTER THE NUMBER OF PAGES:\n");

scanf("%d",&n);

printf("\n ENTER THE PAGE NUMBERS :\n");

for(i=1;i<=n;i++)

scanf("%d",&a[i]);

printf("\n ENTER THE NUMBER OF FRAMES :");

scanf("%d",&no);

for(i=0;i<no;i++)

frame[i]= -1;

 j=0;

        printf("ref string\t page frames\n");

for(i=1;i<=n;i++)

{

   printf("%d\t\t",a[i]);

   avail=0;
```

```
    for(k=0;k<no;k++)
     if(frame[k]==a[i])
            avail=1;
    if (avail==0)
    {
    frame[j]=a[i];
    j=(j+1)%no;
    count++;
    for(k=0;k<no;k++)
    printf("%d\t",frame[k]);
}
printf("\n");
}
printf("No.ofPage Faults %d",count);
}
```

OUTPUT:


input
ENTER THE NUMBER OF PAGES:
20
ENTER THE PAGE NUMBERS :
7
5
8
0
1
6
9
7
0
7
1
2
0
3
7
5
0
9
0
4
ENTER THE NUMBER OF FRAMES :3
ref string page frames
7 7 -1 -1
5 7 5 -1
8 7 5 8
0 0 5 8
1 0 1 8
6 0 1 6
9 9 1 6
7 9 7 6
0 9 7 0
7
1 1 7 0
2 1 2 0
0
3 1 2 3
7 7 2 3
5 7 5 3
0 7 5 0
9 9 5 0
0
4 9 4 0
No.ofPage Faults 17
...Program finished with exit code 0
Press ENTER to exit console.

Experiment 17

AIM: Simulate LRU page replacement algorithm

Description:

This algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.

Program:

```
#include<stdio.h>

#include<conio.h>

int fr[3];

void main()

{

void display();

int index,k,l,flag1=0,flag2=0,pf=0,frsize,i,j,fs[3],n,p[20];

printf("\nenter length of string\n");

scanf("%d",&n);

printf("\n  enter ref string\n");

for(i=0;i<n;i++)

scanf("%d",&p[i]);

printf("\n  enter frame size");

scanf("%d",&frsize);

for(i=0;i<frsize;i++)

{

fr[i]=-1;

}

for(j=0;j<n;j++)
```

```
{
flag1=0,flag2=0;
for(i=0;i<frsize;i++)
{
if(fr[i]==p[j])
{
flag1=1;
flag2=1;
break;
}
}
if(flag1==0)
{
for(i=0;i<frsize;i++)
{
if(fr[i]==-1)
{       pf++;
fr[i]=p[j];
flag2=1;
break;
}
}
}
if(flag2==0)
{
```

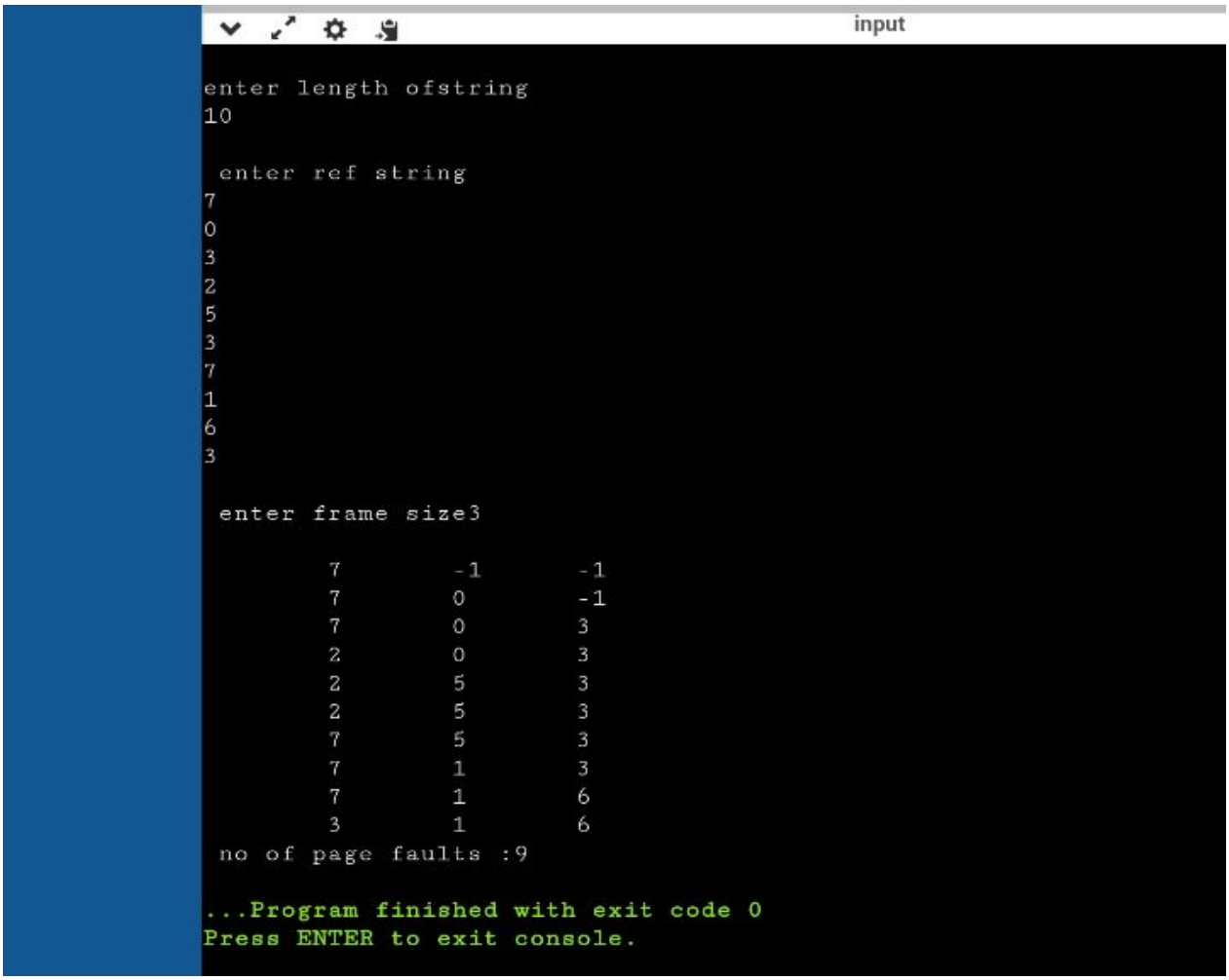
```
for(i=0;i<frsize;i++)
fs[i]=0;
for(k=j-1,l=1;l<=frsize-1;l++,k--)
{
for(i=0;i<frsize;i++)
{
if(fr[i]==p[k])
fs[i]=1;
}
}
for(i=0;i<frsize;i++)
{
if(fs[i]==0)
index=i;
}
fr[index]=p[j];
pf++;
}
display();
}
printf("\n no of page faults :%d",pf);

}
void display()
{
```

```
int i;
printf("\n");
for(i=0;i<3;i++)
printf("\t%d",fr[i]);
}
```

enter length of string

20

OUTPUT:



INDEX

S. No	EXPT. NO	DATE	Name of the Experiment	Page. No	Remarks