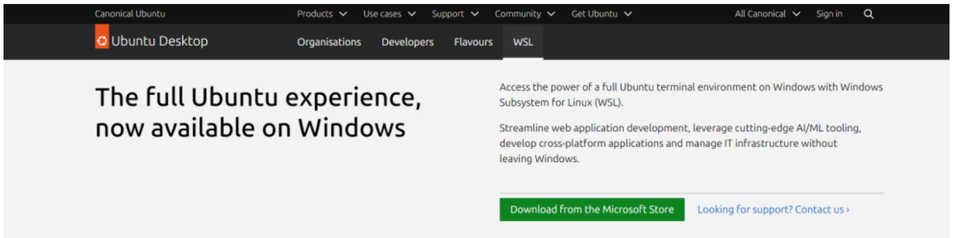
Experiment 1

Aim: Installing Ubuntu on windows and, compile and run first C program using gcc.

Theory:

1. Visit https://ubuntu.com/wsl

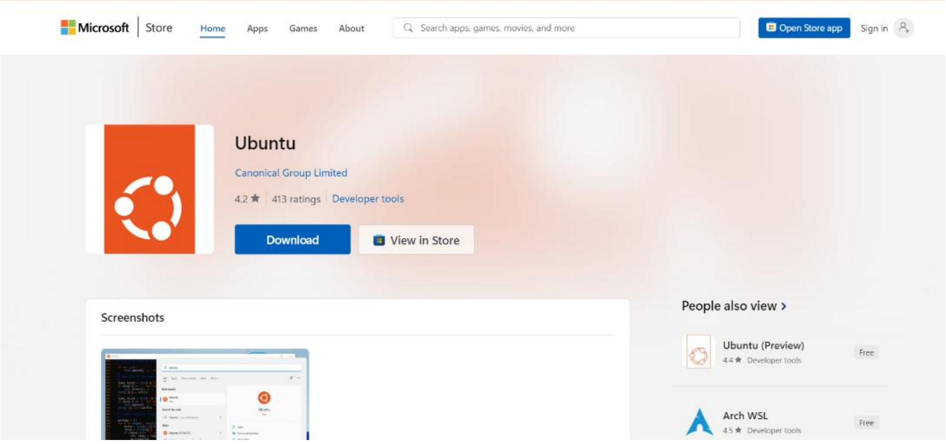


The full Ubuntu experience, now available on Windows

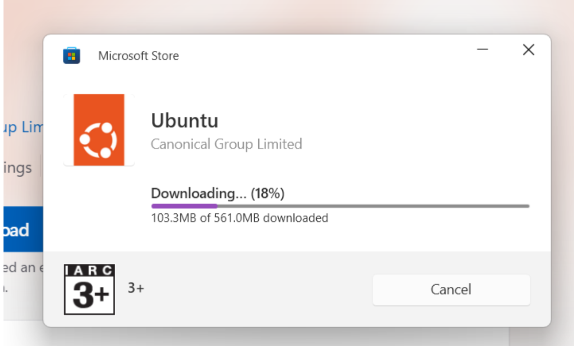
Access the power of a full Ubuntu terminal environment on Windows with Windows Subsystem for Linux (WSL).

Streamline web application development, leverage cutting-edge AI/ML tooling, develop cross-platform applications and manage IT infrastructure without leaving Windows.

2. Click on download.



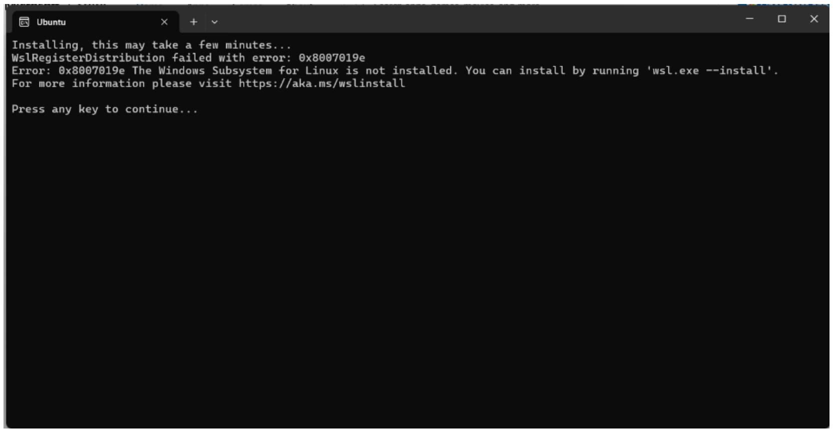
3. Download Started



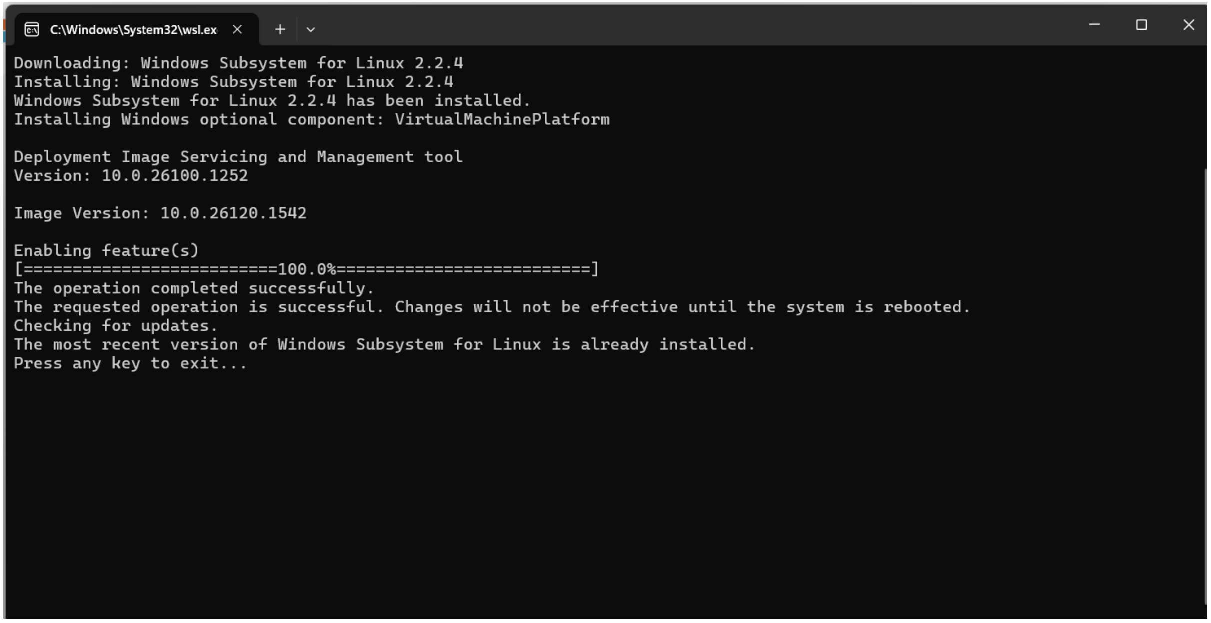
Downloading... (18%)

103.3MB of 561.0MB downloaded

4. Press any button to continue.

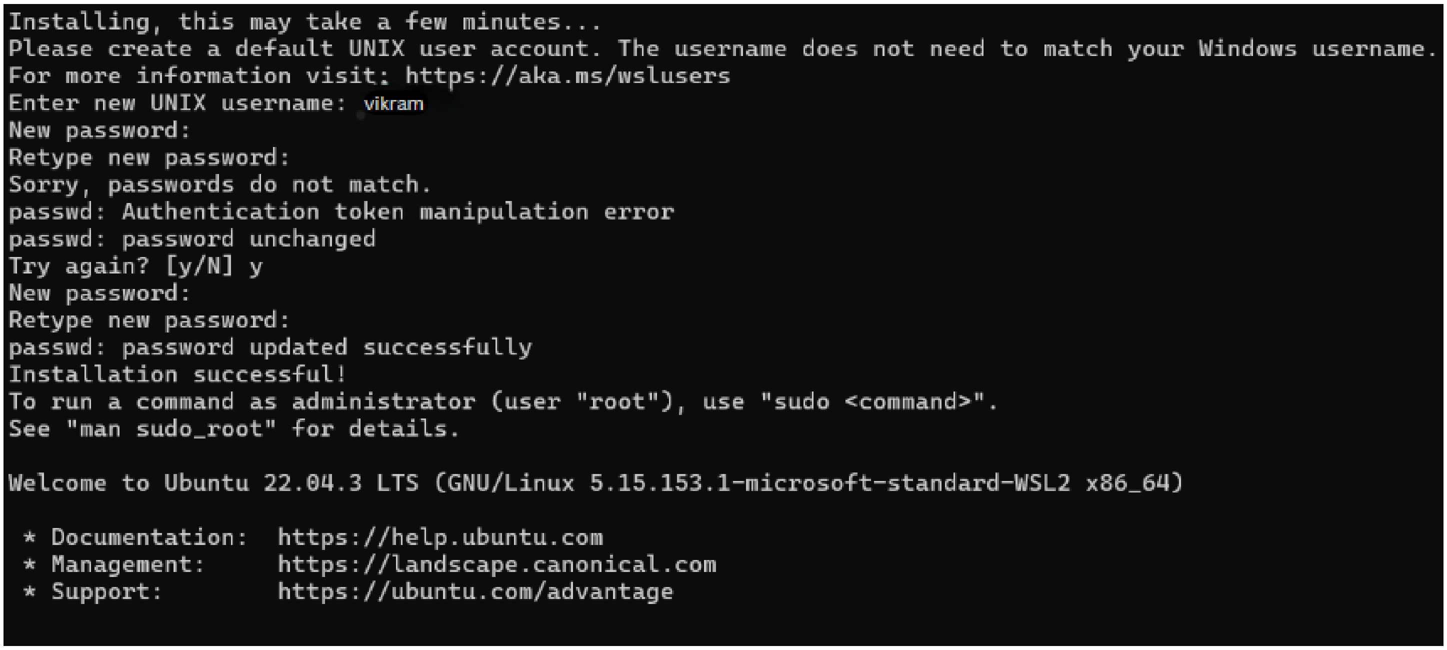


```
Installing, this may take a few minutes...
WslRegisterDistribution failed with error: 0x8007019e
Error: 0x8007019e The Windows Subsystem for Linux is not installed. You can install by running 'wsl.exe --install'.
For more information please visit https://aka.ms/wslinstall

Press any key to continue...
```



```
Downloading: Windows Subsystem for Linux 2.2.4
Installing: Windows Subsystem for Linux 2.2.4
Windows Subsystem for Linux 2.2.4 has been installed.
Installing Windows optional component: VirtualMachinePlatform

Deployment Image Servicing and Management tool
Version: 10.0.26100.1252

Image Version: 10.0.26120.1542

Enabling feature(s)
[==========================100.0%==========================]
The operation completed successfully.
The requested operation is successful. Changes will not be effective until the system is rebooted.
Checking for updates.
The most recent version of Windows Subsystem for Linux is already installed.
Press any key to exit...
```

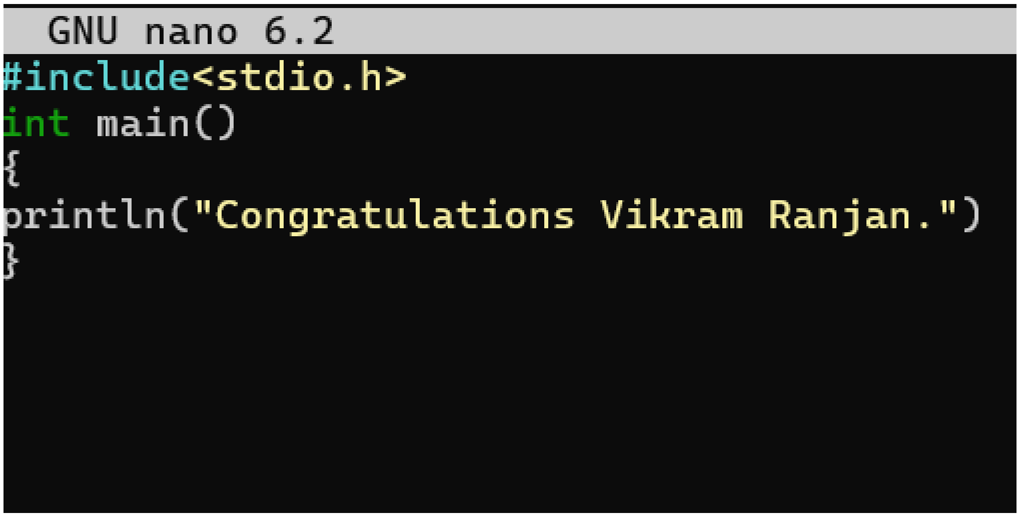
5. Enter Username and Password



```
Installing, this may take a few minutes...
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username: vikram
New password:
Retype new password:
Sorry, passwords do not match.
passwd: Authentication token manipulation error
passwd: password unchanged
Try again? [y/N] y
New password:
Retype new password:
passwd: password updated successfully
Installation successful!
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.153.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage
```

6. type “nano vikram.c”.

7. write c program to congratulate.



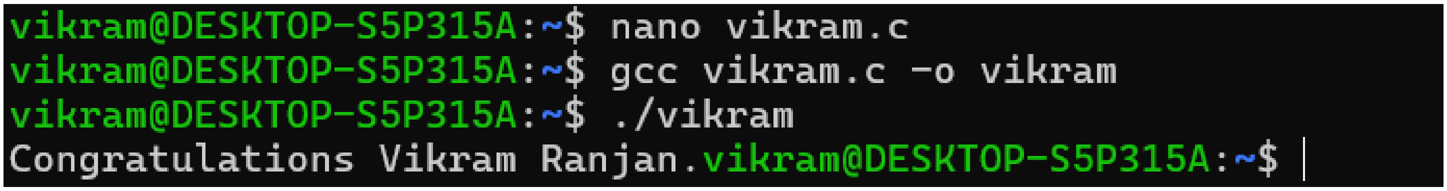
```
  GNU nano 6.2
#include<stdio.h>
int main()
{
println("Congratulations Vikram Ranjan.")
}
```

8. press “ctrl+O” to save program.

9. press “ctrl+x” to exit editor.

10. Type gcc “filename.c” –o “new file name” to compile program.

11. Type ./” filename ” to run program.



```
vikram@DESKTOP-S5P315A:~$ nano vikram.c
vikram@DESKTOP-S5P315A:~$ gcc vikram.c -o vikram
vikram@DESKTOP-S5P315A:~$ ./vikram
Congratulations Vikram Ranjan.vikram@DESKTOP-S5P315A:~$
```

Learning Outcome:

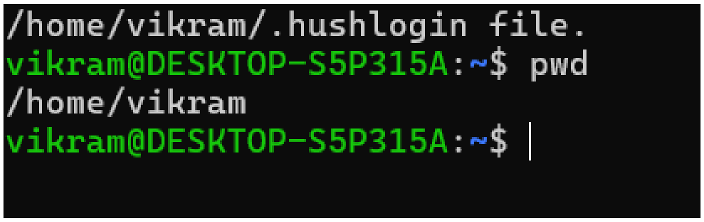
Experiment 2

Aim: Open Ubuntu terminal and write the command for following operations and share the output screen.

Theory:

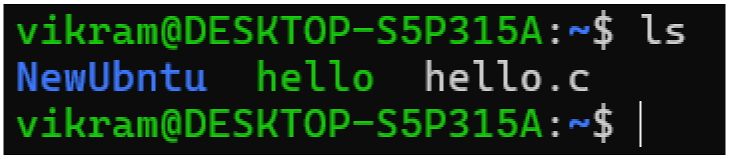
Commands:

1. Command to know your current working directory: pwd



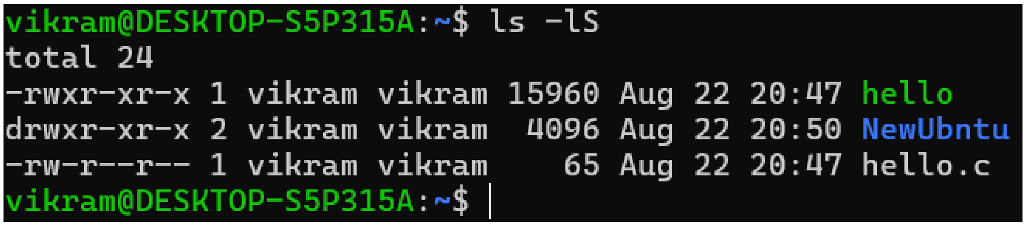
```
/home/vikram/.hushlogin file.
vikram@DESKTOP-S5P315A:~$ pwd
/home/vikram
vikram@DESKTOP-S5P315A:~$
```

2. List all the files in the current directory: ls



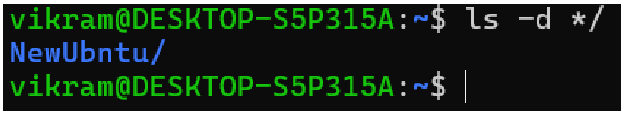
```
vikram@DESKTOP-S5P315A:~$ ls
NewUbntu  hello  hello.c
vikram@DESKTOP-S5P315A:~$
```

3. List all the files in the order of their file size: ls -ls



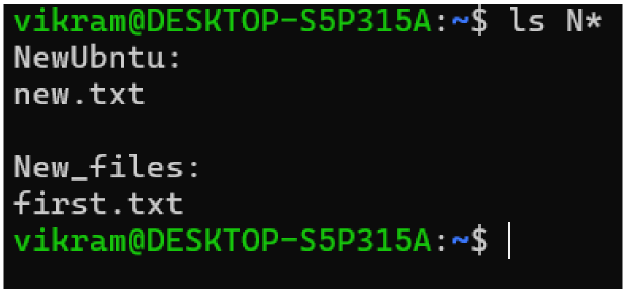
```
vikram@DESKTOP-S5P315A:~$ ls -lS
total 24
-rwxr-xr-x 1 vikram vikram 15960 Aug 22 20:47 hello
drwxr-xr-x 2 vikram vikram  4096 Aug 22 20:50 NewUbntu
-rw-r--r-- 1 vikram vikram    65 Aug 22 20:47 hello.c
vikram@DESKTOP-S5P315A:~$
```

4. List only directories in the current folder: ls -d */



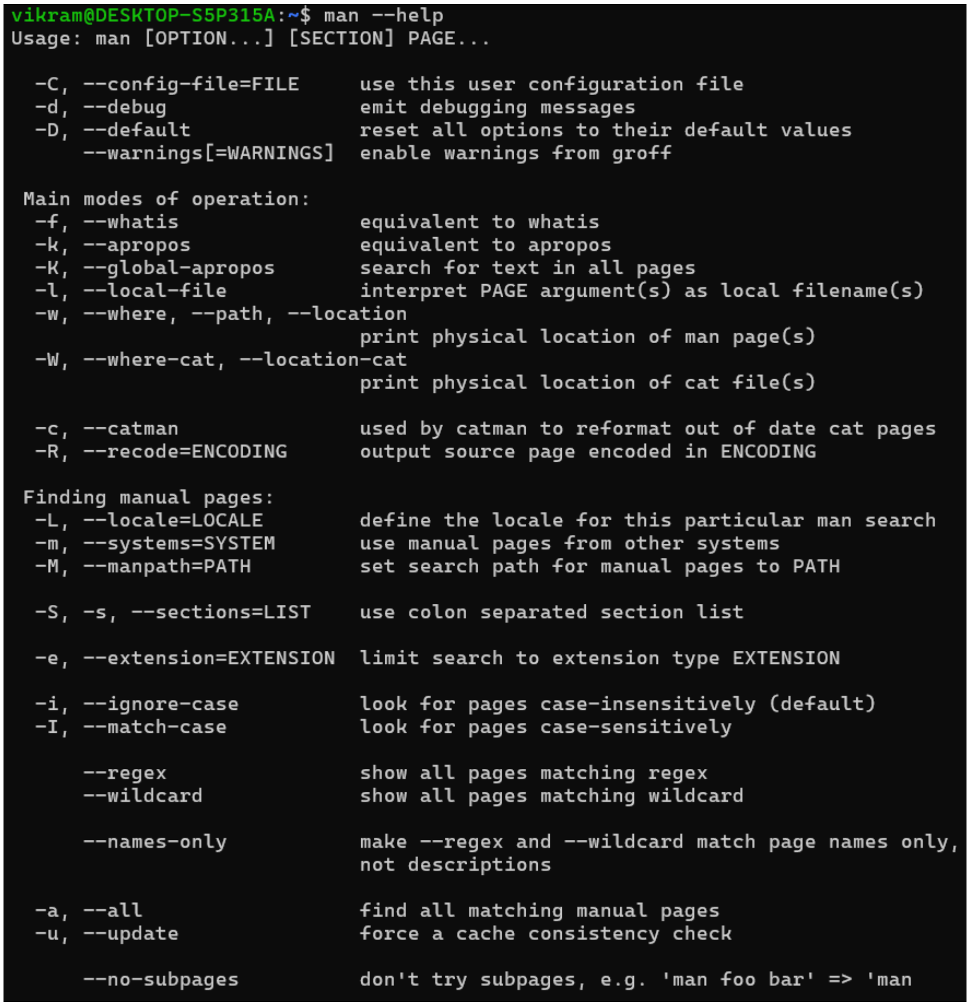
```
vikram@DESKTOP-S5P315A:~$ ls -d */
NewUbntu/
vikram@DESKTOP-S5P315A:~$
```

5. List only files starting with “N” alphabet: ls N*.

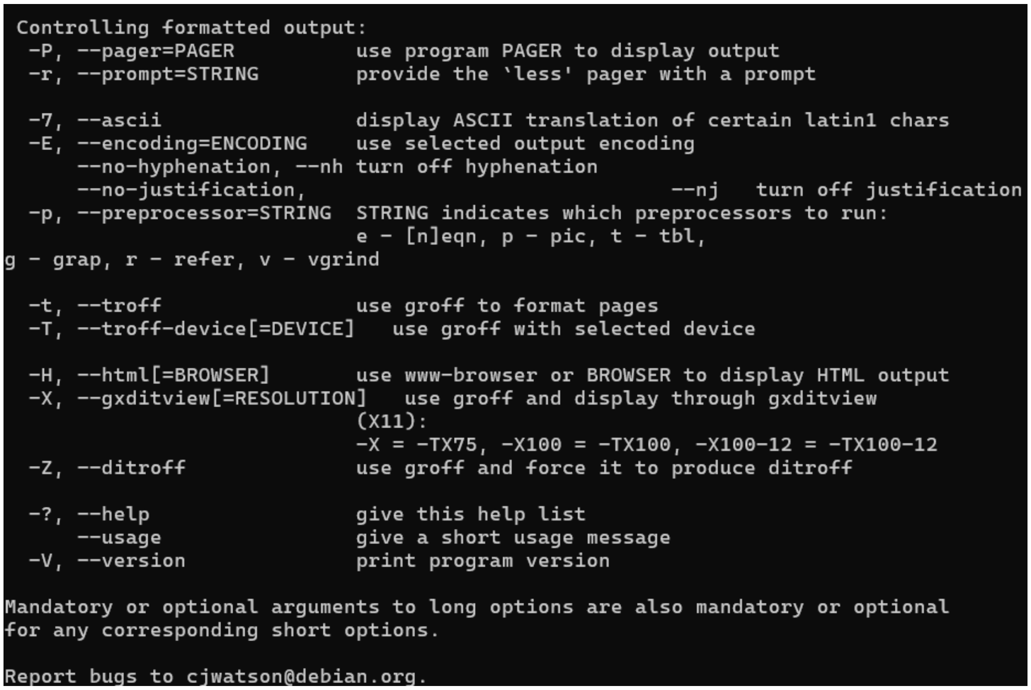


```
vikram@DESKTOP-S5P315A:~$ ls N*
NewUbntu:
new.txt

New_files:
first.txt
vikram@DESKTOP-S5P315A:~$
```

6. Using help command find the help on man command: man –help

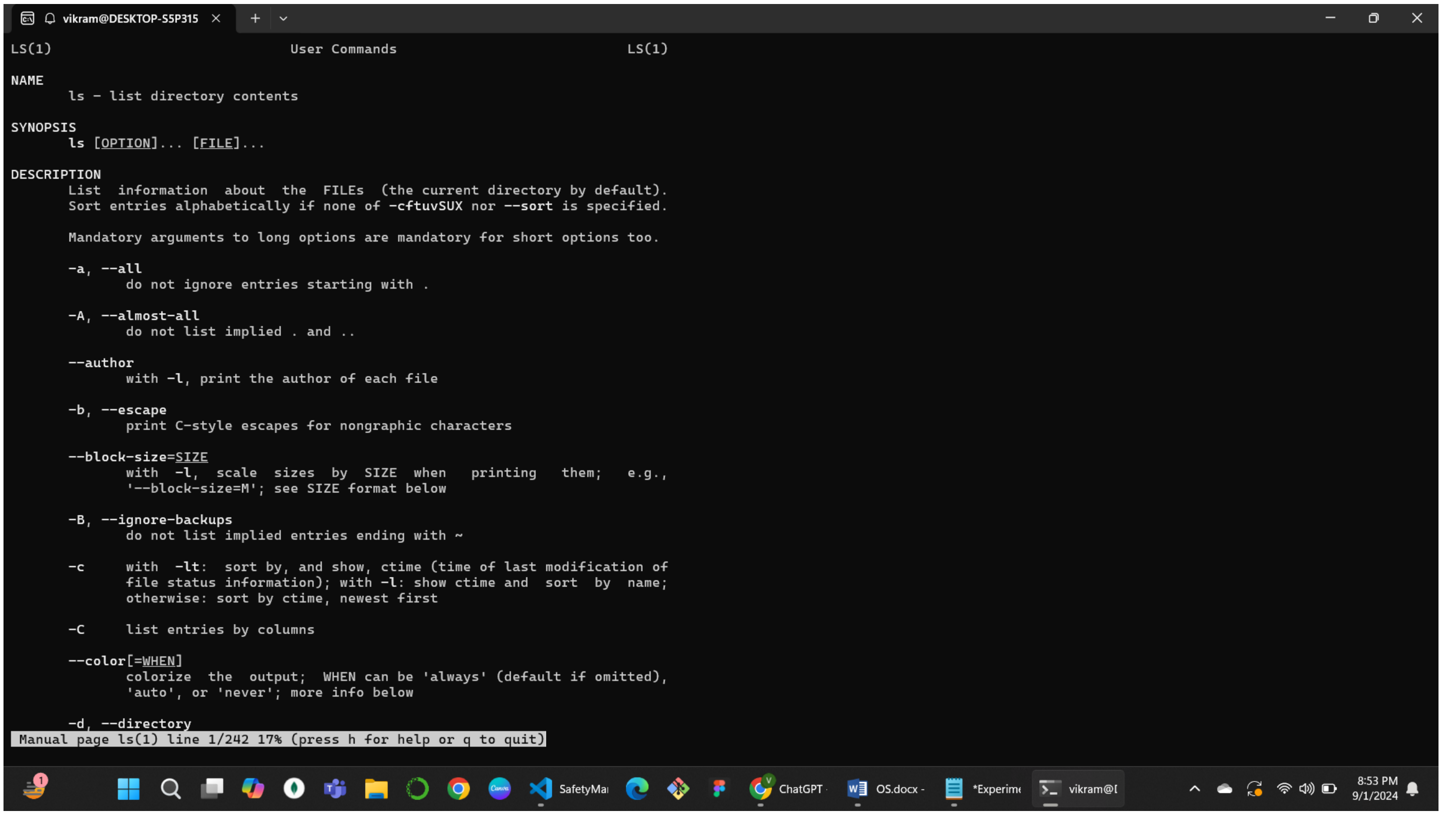


```
vikram@DESKTOP-S5P315A:~$ man --help
Usage: man [OPTION...] [SECTION] PAGE...

  -C, --config-file=FILE     use this user configuration file
  -d, --debug                emit debugging messages
  -D, --default              reset all options to their default values
      --warnings[=WARNINGS]  enable warnings from groff

 Main modes of operation:
  -f, --whatis               equivalent to whatis
  -k, --apropos              equivalent to apropos
  -K, --global-apropos       search for text in all pages
  -l, --local-file           interpret PAGE argument(s) as local filename(s)
  -w, --where, --path, --location
                             print physical location of man page(s)
  -W, --where-cat, --location-cat
                             print physical location of cat file(s)

  -c, --catman               used by catman to reformat out of date cat pages
  -R, --recode=ENCODING      output source page encoded in ENCODING

 Finding manual pages:
  -L, --locale=LOCALE        define the locale for this particular man search
  -m, --systems=SYSTEM       use manual pages from other systems
  -M, --manpath=PATH         set search path for manual pages to PATH

  -S, -s, --sections=LIST    use colon separated section list

  -e, --extension=EXTENSION  limit search to extension type EXTENSION

  -i, --ignore-case          look for pages case-insensitively (default)
  -I, --match-case           look for pages case-sensitively

      --regex                show all pages matching regex
      --wildcard             show all pages matching wildcard

      --names-only           make --regex and --wildcard match page names only,
                             not descriptions

  -a, --all                  find all matching manual pages
  -u, --update               force a cache consistency check

      --no-subpages          don't try subpages, e.g. 'man foo bar' => 'man
```



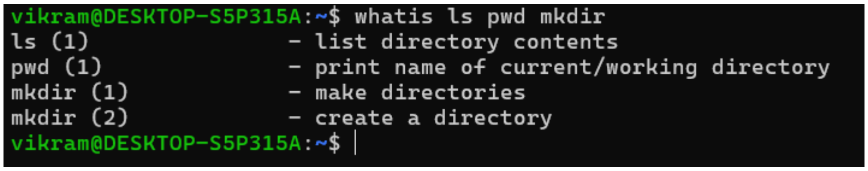
```
 Controlling formatted output:
  -P, --pager=PAGER          use program PAGER to display output
  -r, --prompt=STRING        provide the `less' pager with a prompt

  -7, --ascii                display ASCII translation of certain latin1 chars
  -E, --encoding=ENCODING    use selected output encoding
      --no-hyphenation, --nh turn off hyphenation
      --no-justification,                        --nj   turn off justification
  -p, --preprocessor=STRING  STRING indicates which preprocessors to run:
                             e - [n]eqn, p - pic, t - tbl,
g - grap, r - refer, v - vgrind

  -t, --troff                use groff to format pages
  -T, --troff-device[=DEVICE]   use groff with selected device

  -H, --html[=BROWSER]       use www-browser or BROWSER to display HTML output
  -X, --gxditview[=RESOLUTION]   use groff and display through gxditview
                             (X11):
                             -X = -TX75, -X100 = -TX100, -X100-12 = -TX100-12
  -Z, --ditroff              use groff and force it to produce ditroff

  -?, --help                 give this help list
      --usage                give a short usage message
  -V, --version              print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.

Report bugs to cjwatson@debian.org.
```

7. Display the content of manual pages on ls command: man ls



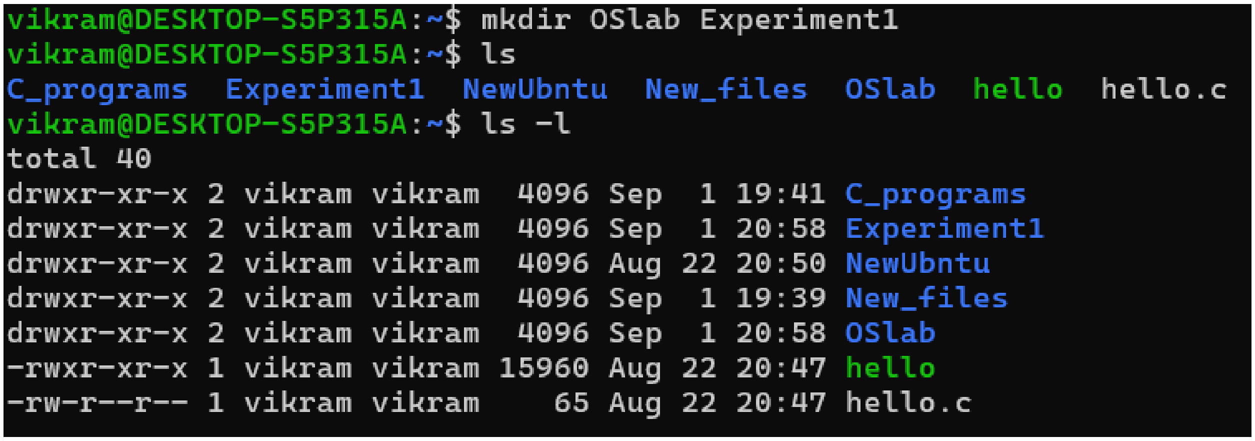
```
LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs  (the current directory by default).
       Sort entries alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with  -l,  scale  sizes  by  SIZE  when   printing   them;   e.g.,
              '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with  -lt:  sort by, and show, ctime (time of last modification of
              file status information); with -l: show ctime and  sort  by  name;
              otherwise: sort by ctime, newest first

       -C     list entries by columns

       --color[=WHEN]
              colorize  the  output;  WHEN can be 'always' (default if omitted),
              'auto', or 'never'; more info below

       -d, --directory
 Manual page ls(1) line 1/242 17% (press h for help or q to quit)
```

8. Demonstrate the usage of “whatis” command: whatis ls pwd mkdir.



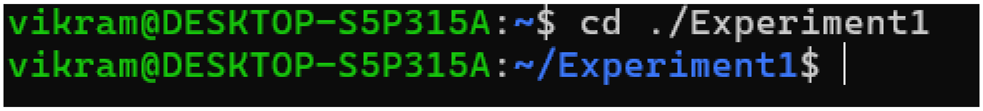
```
vikram@DESKTOP-S5P315A:~$ whatis ls pwd mkdir
ls (1)               - list directory contents
pwd (1)              - print name of current/working directory
mkdir (1)            - make directories
mkdir (2)            - create a directory
vikram@DESKTOP-S5P315A:~$ 
```

9. Make directory named "OSLab/Experiment1". OSLab and Experiment1 both: mkdir OSlab Experiment1



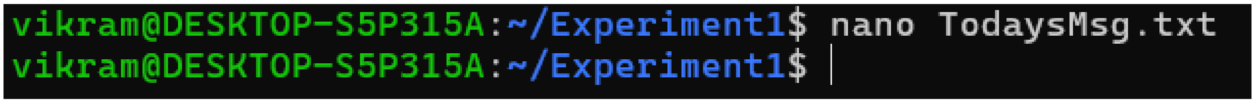
```
vikram@DESKTOP-S5P315A:~$ mkdir OSlab Experiment1
vikram@DESKTOP-S5P315A:~$ ls
C_programs  Experiment1  NewUbntu  New_files  OSlab  hello  hello.c
vikram@DESKTOP-S5P315A:~$ ls -l
total 40
drwxr-xr-x 2 vikram vikram  4096 Sep  1 19:41 C_programs
drwxr-xr-x 2 vikram vikram  4096 Sep  1 20:58 Experiment1
drwxr-xr-x 2 vikram vikram  4096 Aug 22 20:50 NewUbntu
drwxr-xr-x 2 vikram vikram  4096 Sep  1 19:39 New_files
drwxr-xr-x 2 vikram vikram  4096 Sep  1 20:58 OSlab
-rwxr-xr-x 1 vikram vikram 15960 Aug 22 20:47 hello
-rw-r--r-- 1 vikram vikram    65 Aug 22 20:47 hello.c
```

10. Write command to reach to "Experiment1" Directory: cd Experiment1.

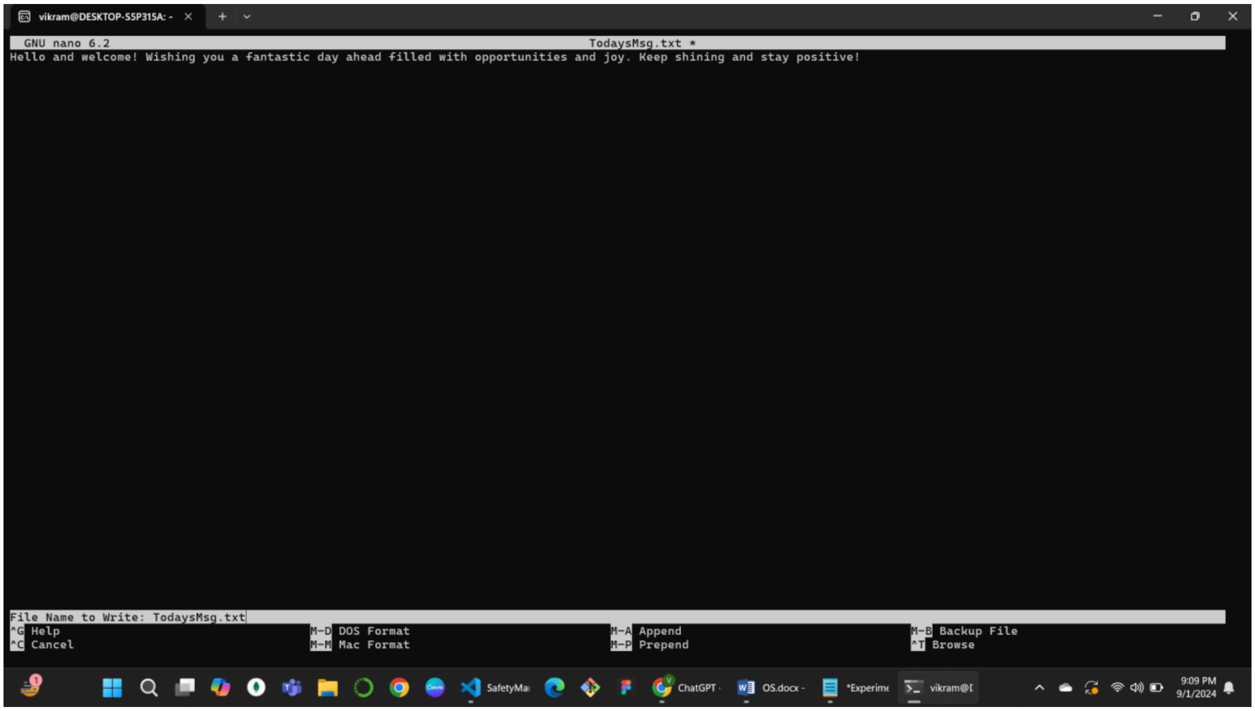


```
vikram@DESKTOP-S5P315A:~$ cd ./Experiment1
vikram@DESKTOP-S5P315A:~/Experiment1$ 
```

11. Create a txt file named "TodaysMsg.txt" and write a greeting message in it.

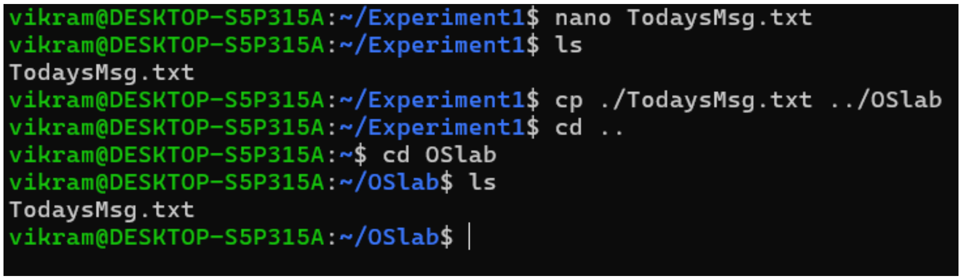


```
vikram@DESKTOP-S5P315A:~/Experiment1$ nano TodaysMsg.txt
vikram@DESKTOP-S5P315A:~/Experiment1$ 
```



```
  GNU nano 6.2                                    TodaysMsg.txt *
Hello and welcome! Wishing you a fantastic day ahead filled with opportunities and joy. Keep shining and stay positive!

File Name to Write: TodaysMsg.txt
^G Help                  M-D DOS Format           M-A Append               M-B Backup File
^C Cancel                M-M Mac Format           M-P Prepend              ^T Browse
```

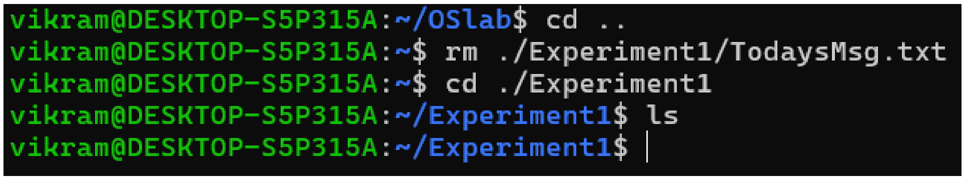
12. Copy this file "TodaysMsg.txt" to OSLab directory:cp ./TodaysMsg.txt ../OSlab



```
vikram@DESKTOP-S5P315A:~/Experiment1$ nano TodaysMsg.txt
vikram@DESKTOP-S5P315A:~/Experiment1$ ls
TodaysMsg.txt
vikram@DESKTOP-S5P315A:~/Experiment1$ cp ./TodaysMsg.txt ../OSlab
vikram@DESKTOP-S5P315A:~/Experiment1$ cd ..
vikram@DESKTOP-S5P315A:~$ cd OSlab
vikram@DESKTOP-S5P315A:~/OSlab$ ls
TodaysMsg.txt
vikram@DESKTOP-S5P315A:~/OSlab$ 
```

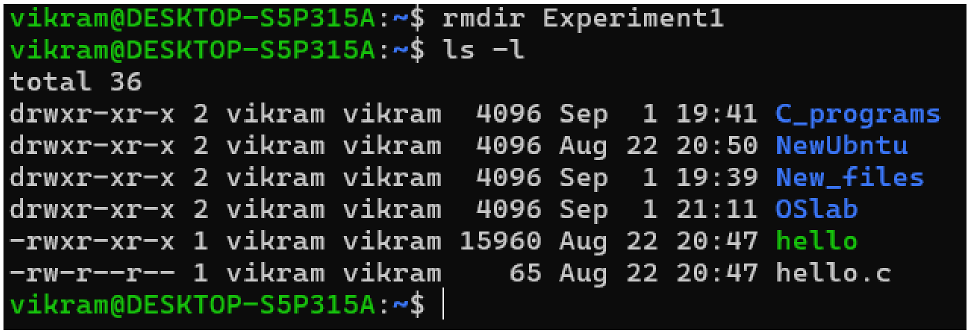
13. Delete the file “TodaysMsg.txt” from the Experiment1 Folder:

rm ./Experiment1/TodaysMsg.txt



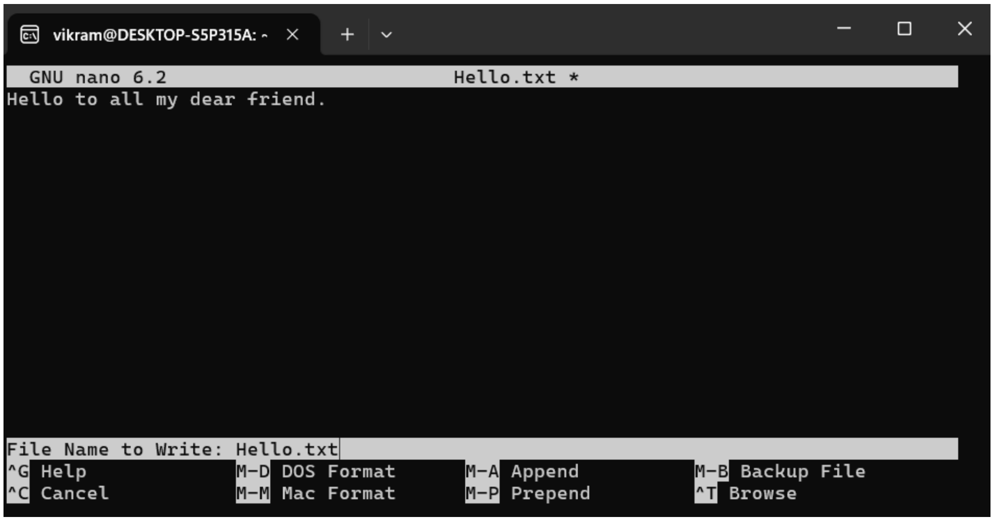
```
vikram@DESKTOP-S5P315A:~/OSlab$ cd ..
vikram@DESKTOP-S5P315A:~$ rm ./Experiment1/TodaysMsg.txt
vikram@DESKTOP-S5P315A:~$ cd ./Experiment1
vikram@DESKTOP-S5P315A:~/Experiment1$ ls
vikram@DESKTOP-S5P315A:~/Experiment1$ 
```

14. Delete the directory Experiment1: rmdir Experiment1



```
vikram@DESKTOP-S5P315A:~$ rmdir Experiment1
vikram@DESKTOP-S5P315A:~$ ls -l
total 36
drwxr-xr-x 2 vikram vikram  4096 Sep  1 19:41 C_programs
drwxr-xr-x 2 vikram vikram  4096 Aug 22 20:50 NewUbntu
drwxr-xr-x 2 vikram vikram  4096 Sep  1 19:39 New_files
drwxr-xr-x 2 vikram vikram  4096 Sep  1 21:11 OSlab
-rwxr-xr-x 1 vikram vikram 15960 Aug 22 20:47 hello
-rw-r--r-- 1 vikram vikram    65 Aug 22 20:47 hello.c
vikram@DESKTOP-S5P315A:~$ 
```

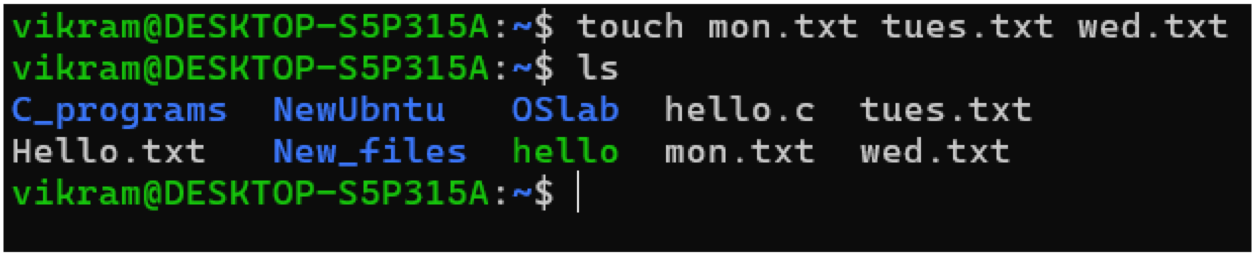
15. Create a text file named “Hello.txt” and write a suitable message in it.



```
  GNU nano 6.2                         Hello.txt *
Hello to all my dear friend.

File Name to Write: Hello.txt
^G Help             M-D DOS Format      M-A Append          M-B Backup File
^C Cancel           M-M Mac Format      M-P Prepend         ^T Browse
```

16. Using touch command create files with names mon.txt, tues.txt, and wed.txt:

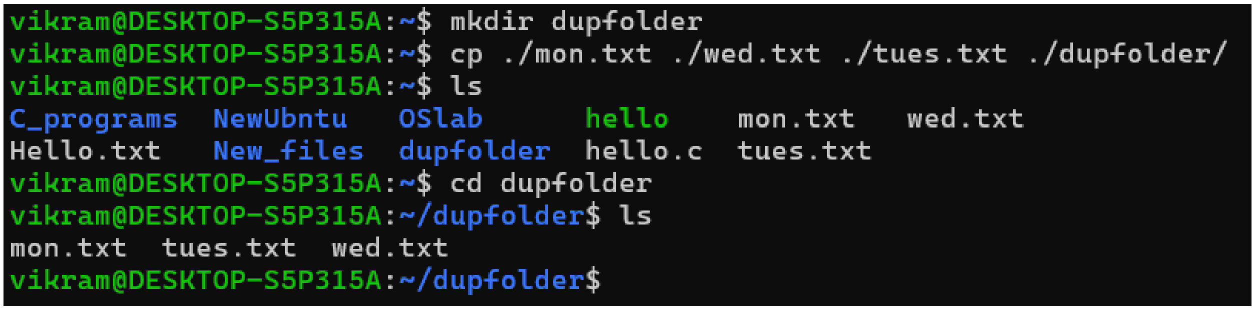
touch mon.txt tues.txt wed.txt.



```
vikram@DESKTOP-S5P315A:~$ touch mon.txt tues.txt wed.txt
vikram@DESKTOP-S5P315A:~$ ls
C_programs  NewUbntu   OSlab  hello.c  tues.txt
Hello.txt   New_files  hello  mon.txt  wed.txt
vikram@DESKTOP-S5P315A:~$
```

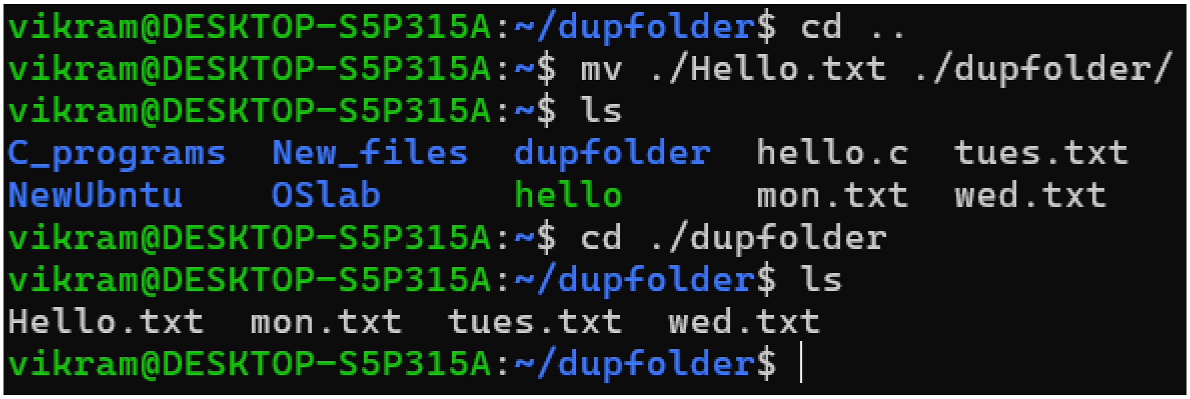
17. Copy these newly created files to a folder named “dupfolder” after creating it.

mkdir dupfolder   cp ./mon.txt ./wed.txt ./tues.txt ./dupfolder/



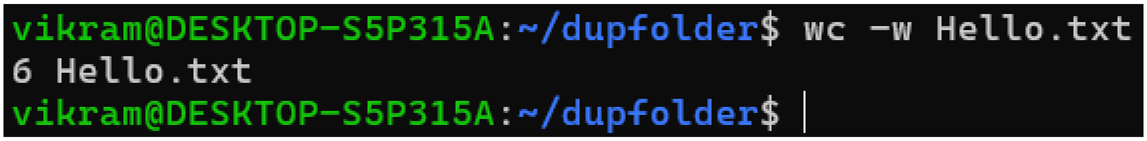
```
vikram@DESKTOP-S5P315A:~$ mkdir dupfolder
vikram@DESKTOP-S5P315A:~$ cp ./mon.txt ./wed.txt ./tues.txt ./dupfolder/
vikram@DESKTOP-S5P315A:~$ ls
C_programs  NewUbntu   OSlab      hello    mon.txt   wed.txt
Hello.txt   New_files  dupfolder  hello.c  tues.txt
vikram@DESKTOP-S5P315A:~$ cd dupfolder
vikram@DESKTOP-S5P315A:~/dupfolder$ ls
mon.txt  tues.txt  wed.txt
vikram@DESKTOP-S5P315A:~/dupfolder$
```

18. Move Hello.txt to dupfolder: mv ./Hello.txt ./dupfolder/



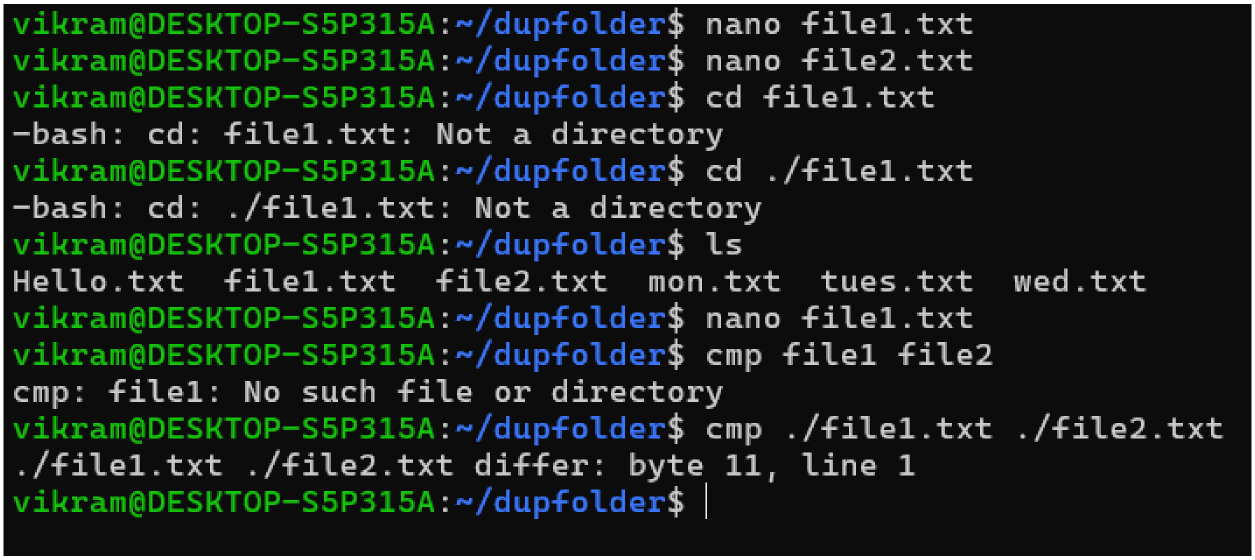
```
vikram@DESKTOP-S5P315A:~/dupfolder$ cd ..
vikram@DESKTOP-S5P315A:~$ mv ./Hello.txt ./dupfolder/
vikram@DESKTOP-S5P315A:~$ ls
C_programs  New_files  dupfolder  hello.c  tues.txt
NewUbntu    OSlab      hello      mon.txt  wed.txt
vikram@DESKTOP-S5P315A:~$ cd ./dupfolder
vikram@DESKTOP-S5P315A:~/dupfolder$ ls
Hello.txt  mon.txt  tues.txt  wed.txt
vikram@DESKTOP-S5P315A:~/dupfolder$
```

19. Count number of words in the Hello.txt file: wc -w Hello.txt



```
vikram@DESKTOP-S5P315A:~/dupfolder$ wc -w Hello.txt
6 Hello.txt
vikram@DESKTOP-S5P315A:~/dupfolder$
```

20. Create two files with identical content, change one alphabet in one of these and compare them using cmp command: cmp ./file1.txt ./file2.txt



```
vikram@DESKTOP-S5P315A:~/dupfolder$ nano file1.txt
vikram@DESKTOP-S5P315A:~/dupfolder$ nano file2.txt
vikram@DESKTOP-S5P315A:~/dupfolder$ cd file1.txt
-bash: cd: file1.txt: Not a directory
vikram@DESKTOP-S5P315A:~/dupfolder$ cd ./file1.txt
-bash: cd: ./file1.txt: Not a directory
vikram@DESKTOP-S5P315A:~/dupfolder$ ls
Hello.txt  file1.txt  file2.txt  mon.txt  tues.txt  wed.txt
vikram@DESKTOP-S5P315A:~/dupfolder$ nano file1.txt
vikram@DESKTOP-S5P315A:~/dupfolder$ cmp file1 file2
cmp: file1: No such file or directory
vikram@DESKTOP-S5P315A:~/dupfolder$ cmp ./file1.txt ./file2.txt
./file1.txt ./file2.txt differ: byte 11, line 1
vikram@DESKTOP-S5P315A:~/dupfolder$
```

Learning Outcome:

Experiment 3

Aim:

Perform following shell script based programs

a. Write a Shell Program to swap the two integers.

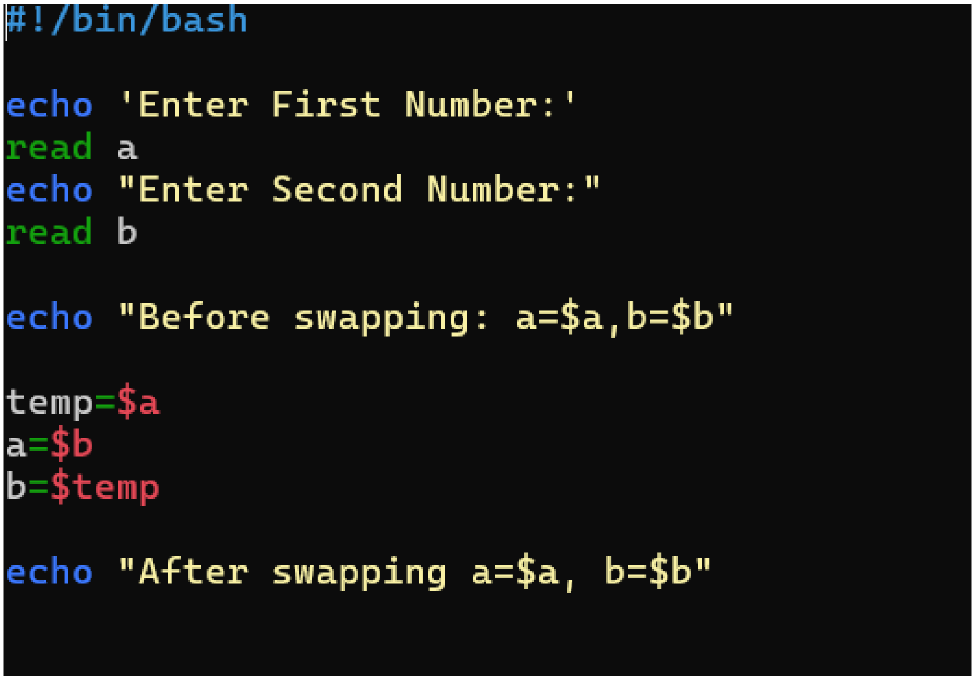
b. Create a shell script that checks if a specific directory exists. If it does, the script should back up all files from that directory into a specified backup directory. The script should then loop through the files in the backup directory and list all files that were successfully copied. If the directory does not exist, the script should print an error message.

c. Write a shell script to check if a given number is a prime number or not

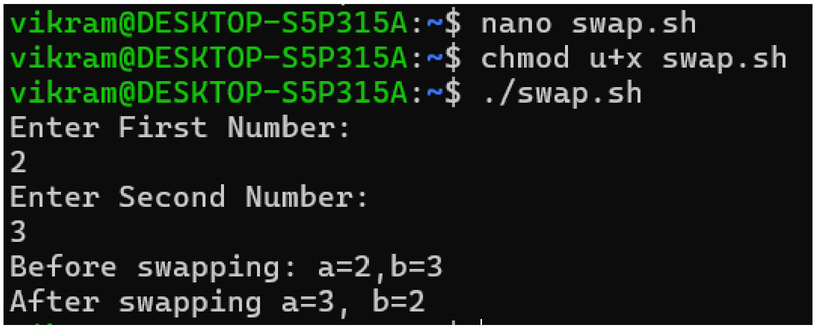
d. Write a shell script to greet the user as per the time whenever he/ she opens terminal.

Theory:

a. Write a Shell Program to swap the two integers.

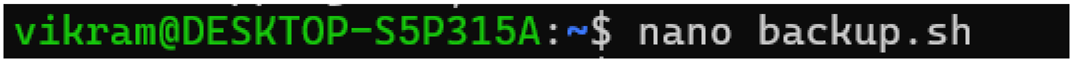


```
#!/bin/bash

echo 'Enter First Number:'
read a
echo "Enter Second Number:"
read b

echo "Before swapping: a=$a,b=$b"

temp=$a
a=$b
b=$temp

echo "After swapping a=$a, b=$b"
```

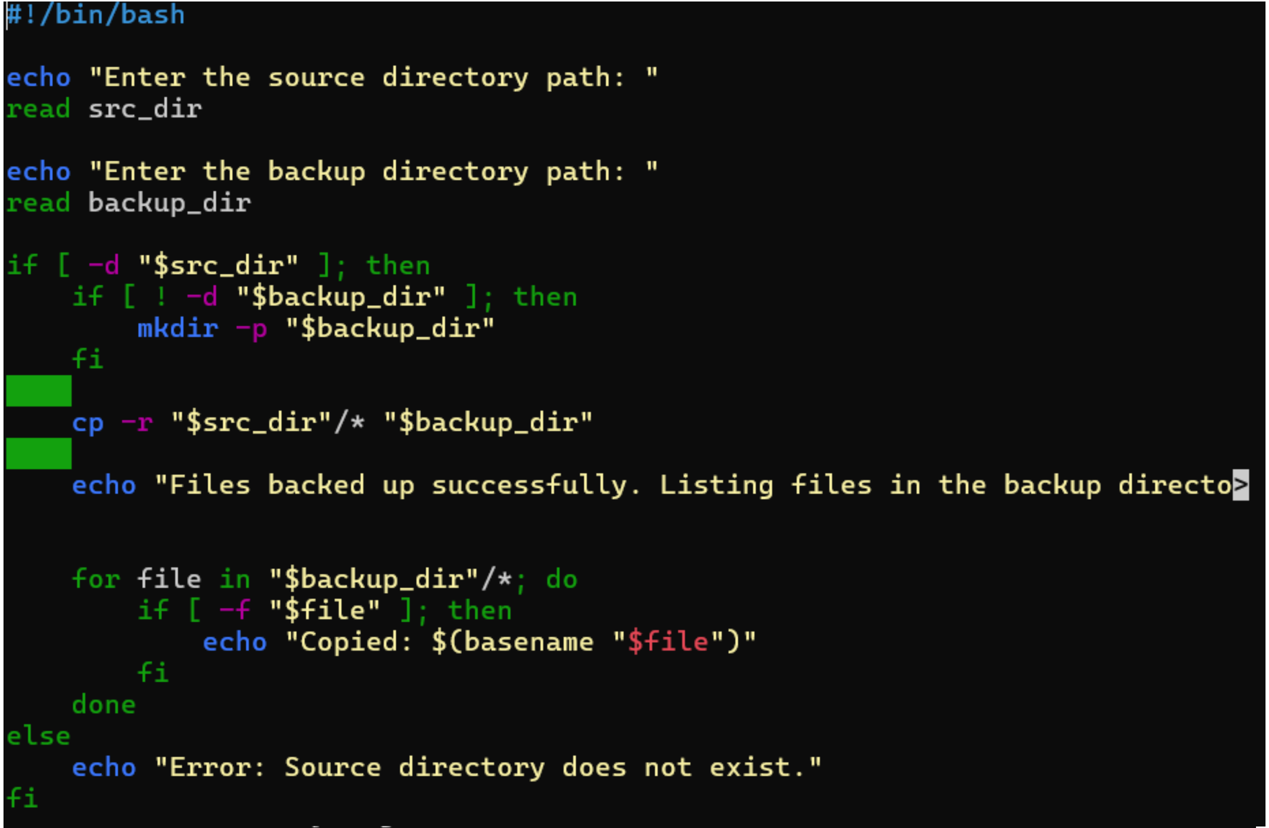


```
vikram@DESKTOP-S5P315A:~$ nano swap.sh
vikram@DESKTOP-S5P315A:~$ chmod u+x swap.sh
vikram@DESKTOP-S5P315A:~$ ./swap.sh
Enter First Number:
2
Enter Second Number:
3
Before swapping: a=2,b=3
After swapping a=3, b=2
```

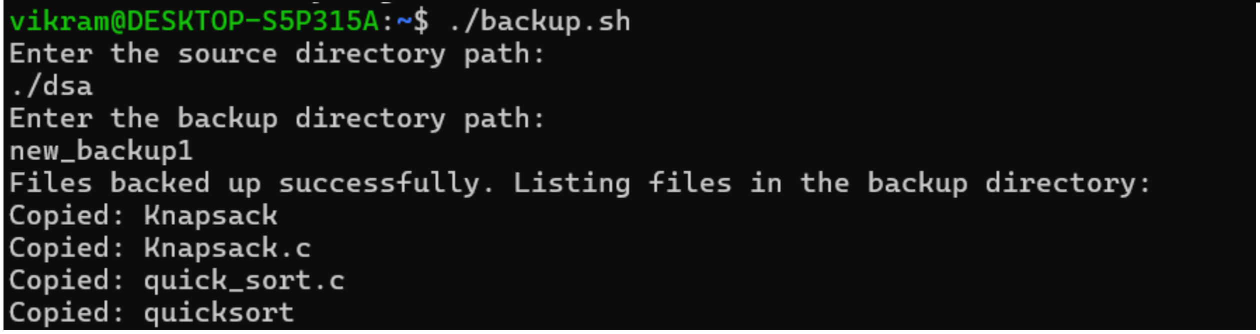
b. Create a shell script that checks if a specific directory exists. If it does, the script should back up all files from that directory into a specified backup directory. The script should then loop through the files in the backup directory and list all files that were successfully copied. If the directory does not exist, the script should print an error message.



```
vikram@DESKTOP-S5P315A:~$ nano backup.sh
```

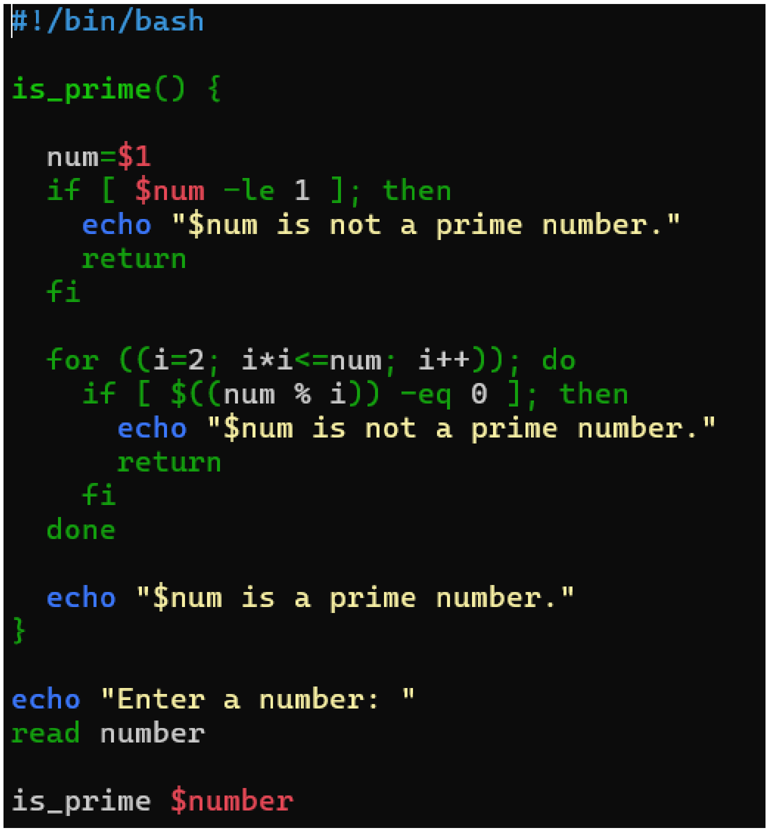


```
#!/bin/bash

echo "Enter the source directory path: "
read src_dir

echo "Enter the backup directory path: "
read backup_dir

if [ -d "$src_dir" ]; then
    if [ ! -d "$backup_dir" ]; then
        mkdir -p "$backup_dir"
    fi

    cp -r "$src_dir"/* "$backup_dir"

    echo "Files backed up successfully. Listing files in the backup directo>


    for file in "$backup_dir"/*; do
        if [ -f "$file" ]; then
            echo "Copied: $(basename "$file")"
        fi
    done
else
    echo "Error: Source directory does not exist."
fi
```

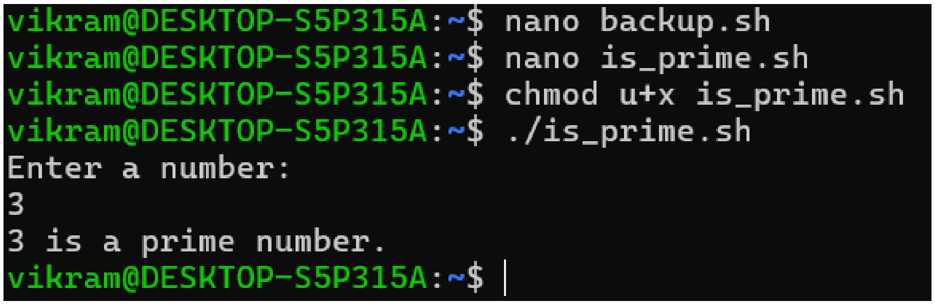


```
vikram@DESKTOP-S5P315A:~$ ./backup.sh
Enter the source directory path:
./dsa
Enter the backup directory path:
new_backup1
Files backed up successfully. Listing files in the backup directory:
Copied: Knapsack
Copied: Knapsack.c
Copied: quick_sort.c
Copied: quicksort
```

c. Write a shell script to check if a given number is a prime number or not.

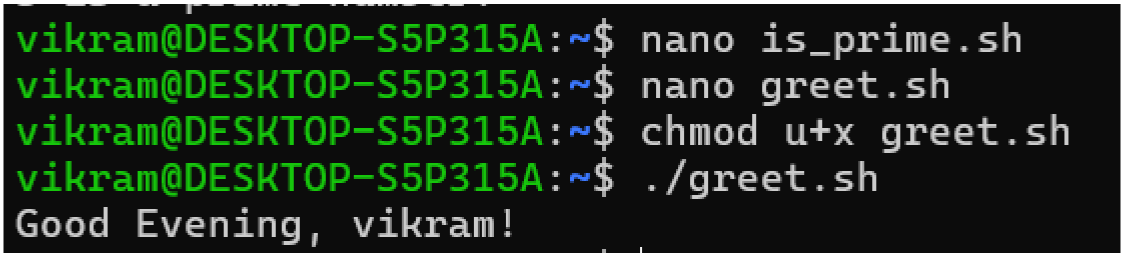


```
#!/bin/bash

is_prime() {

  num=$1
  if [ $num -le 1 ]; then
    echo "$num is not a prime number."
    return
  fi

  for ((i=2; i*i<=num; i++)); do
    if [ $((num % i)) -eq 0 ]; then
      echo "$num is not a prime number."
      return
    fi
  done

  echo "$num is a prime number."
}

echo "Enter a number: "
read number

is_prime $number
```

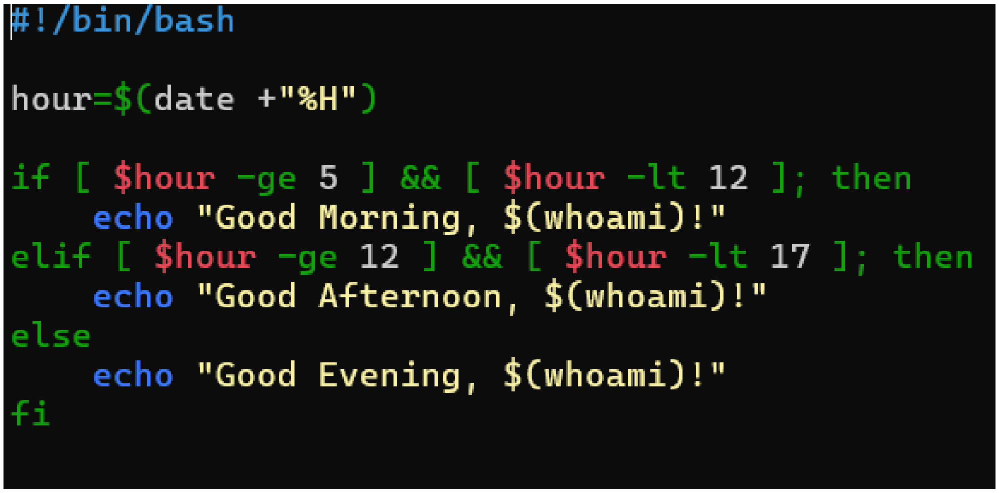


```
vikram@DESKTOP-S5P315A:~$ nano backup.sh
vikram@DESKTOP-S5P315A:~$ nano is_prime.sh
vikram@DESKTOP-S5P315A:~$ chmod u+x is_prime.sh
vikram@DESKTOP-S5P315A:~$ ./is_prime.sh
Enter a number:
3
3 is a prime number.
vikram@DESKTOP-S5P315A:~$
```

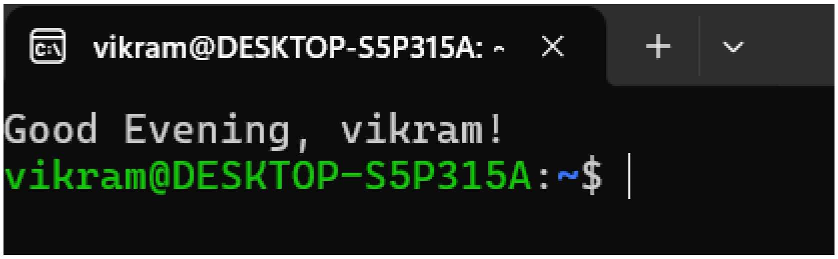
d. Write a shell script to greet the user as per the time whenever he/ she opens terminal.



```
vikram@DESKTOP-S5P315A:~$ nano is_prime.sh
vikram@DESKTOP-S5P315A:~$ nano greet.sh
vikram@DESKTOP-S5P315A:~$ chmod u+x greet.sh
vikram@DESKTOP-S5P315A:~$ ./greet.sh
Good Evening, vikram!
```



```bash
#!/bin/bash

hour=$(date +"%H")

if [ $hour -ge 5 ] && [ $hour -lt 12 ]; then
    echo "Good Morning, $(whoami)!"
elif [ $hour -ge 12 ] && [ $hour -lt 17 ]; then
    echo "Good Afternoon, $(whoami)!"
else
    echo "Good Evening, $(whoami)!"
fi
```



```
Good Evening, vikram!
vikram@DESKTOP-S5P315A:~$
```

Learning Outcome:

Experiment 4

Aim: Write a c program to implement the following scheduling algorithms. First come first serve

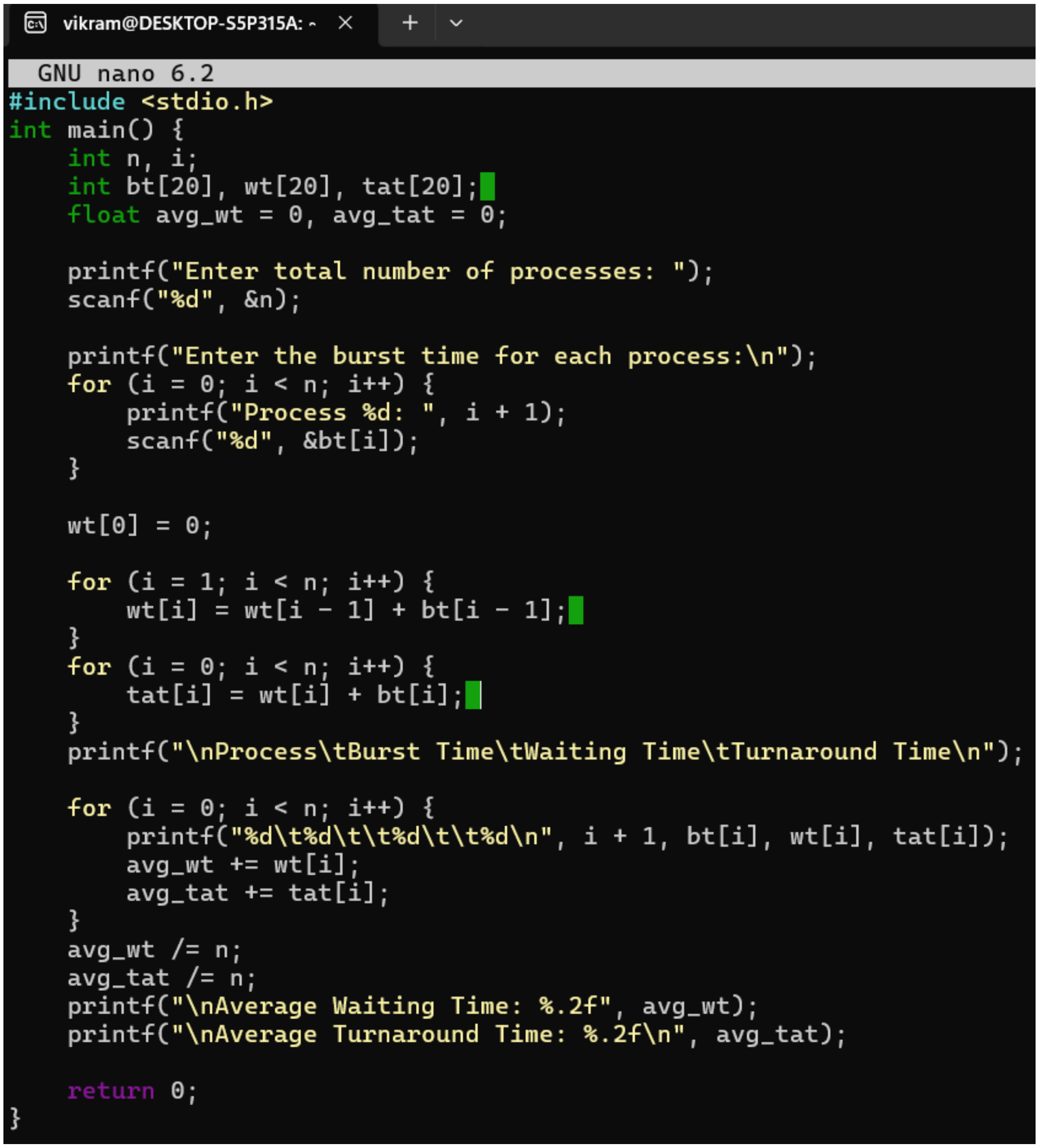
a) Round Robin Scheduling

b) Shortest job first

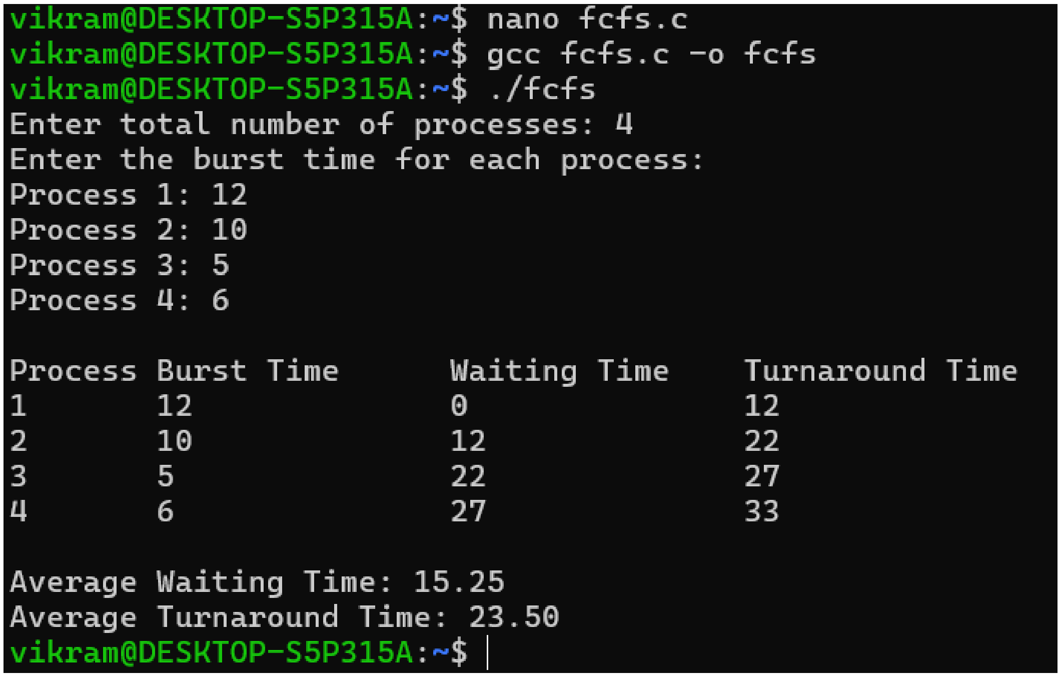
c) Shortest Job remaining first.

Theory:

a. First come first serve

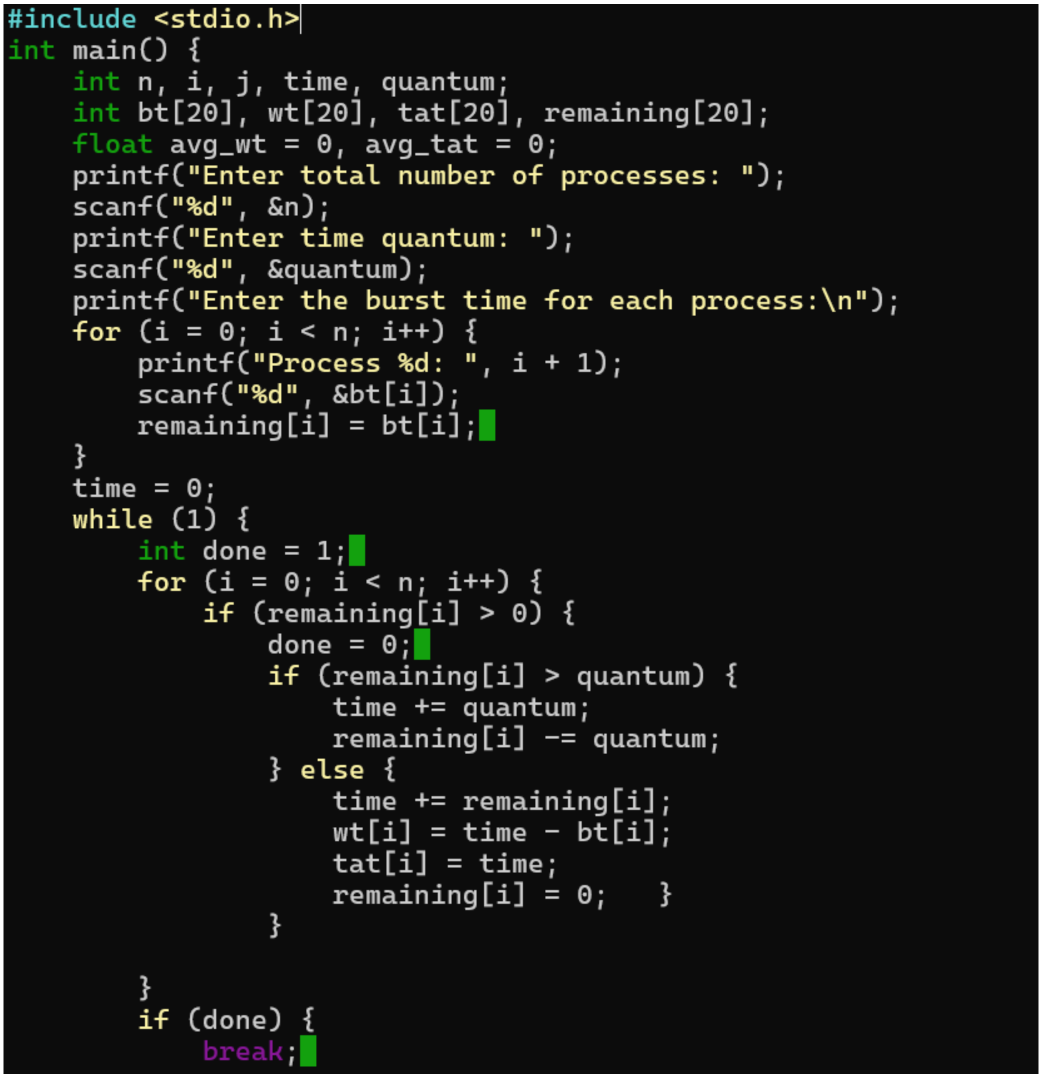


```
  GNU nano 6.2
#include <stdio.h>
int main() {
    int n, i;
    int bt[20], wt[20], tat[20];
    float avg_wt = 0, avg_tat = 0;

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

    printf("Enter the burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
    }

    wt[0] = 0;

    for (i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
    }
    for (i = 0; i < n; i++) {
        tat[i] = wt[i] + bt[i];
    }
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");

    for (i = 0; i < n; i++) {
        printf("%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
        avg_wt += wt[i];
        avg_tat += tat[i];
    }
    avg_wt /= n;
    avg_tat /= n;
    printf("\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);

    return 0;
}
```

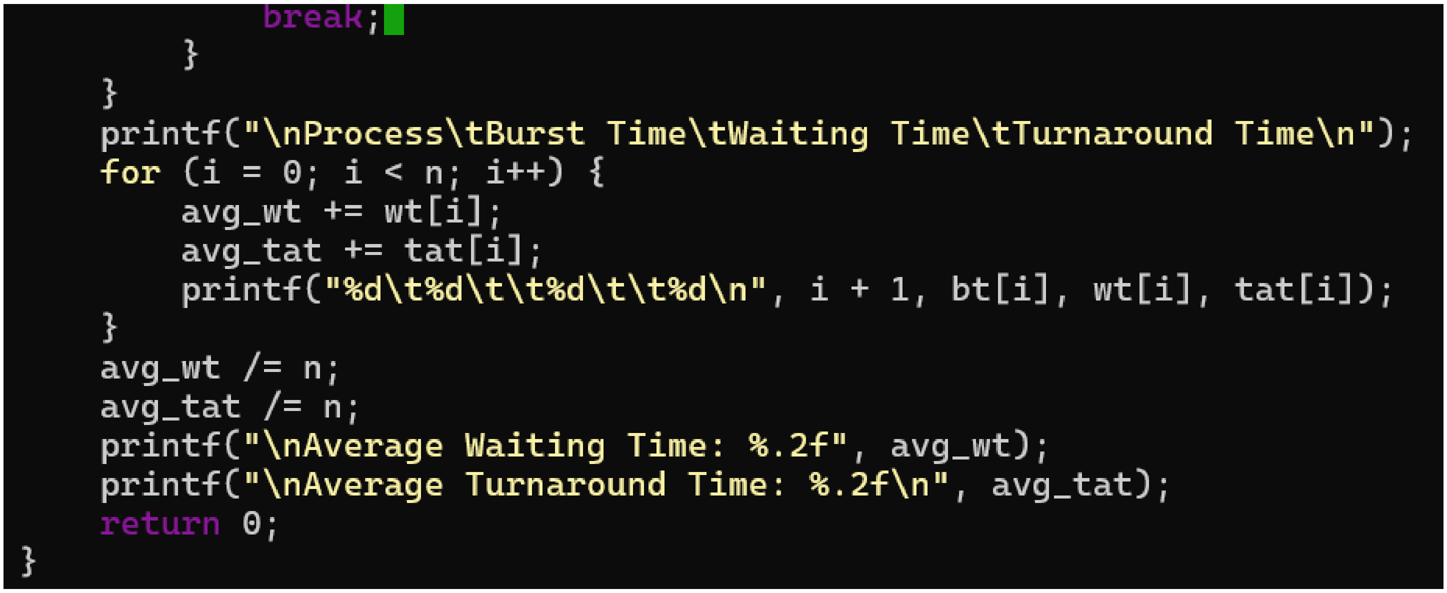


```
vikram@DESKTOP-S5P315A:~$ nano fcfs.c
vikram@DESKTOP-S5P315A:~$ gcc fcfs.c -o fcfs
vikram@DESKTOP-S5P315A:~$ ./fcfs
Enter total number of processes: 4
Enter the burst time for each process:
Process 1: 12
Process 2: 10
Process 3: 5
Process 4: 6

Process Burst Time      Waiting Time    Turnaround Time
1       12              0               12
2       10              12              22
3       5               22              27
4       6               27              33

Average Waiting Time: 15.25
Average Turnaround Time: 23.50
vikram@DESKTOP-S5P315A:~$ 
```

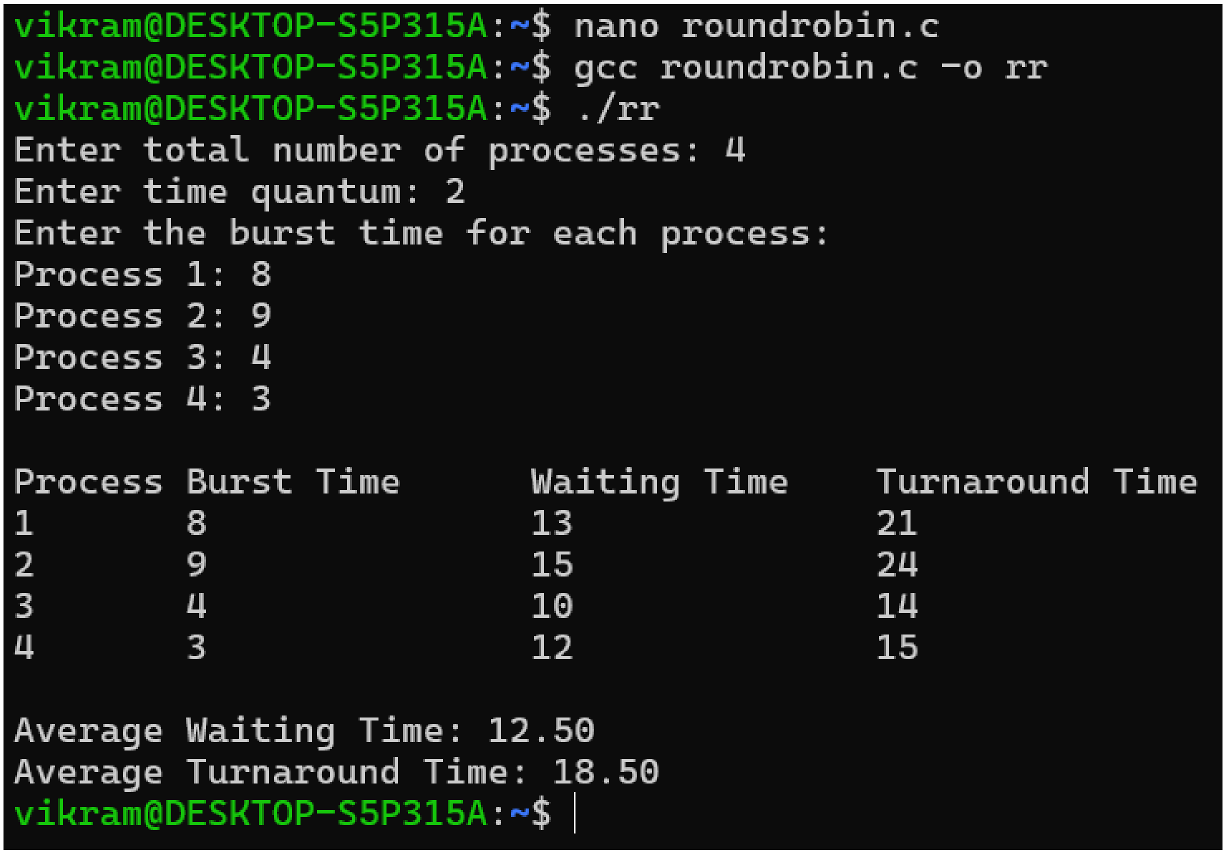
b. Round Robin Scheduling



```c
#include <stdio.h>
int main() {
    int n, i, j, time, quantum;
    int bt[20], wt[20], tat[20], remaining[20];
    float avg_wt = 0, avg_tat = 0;
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    printf("Enter time quantum: ");
    scanf("%d", &quantum);
    printf("Enter the burst time for each process:\n");
    for (i = 0; i < n; i++) {
        printf("Process %d: ", i + 1);
        scanf("%d", &bt[i]);
        remaining[i] = bt[i];
    }
    time = 0;
    while (1) {
        int done = 1;
        for (i = 0; i < n; i++) {
            if (remaining[i] > 0) {
                done = 0;
                if (remaining[i] > quantum) {
                    time += quantum;
                    remaining[i] -= quantum;
                } else {
                    time += remaining[i];
                    wt[i] = time - bt[i];
                    tat[i] = time;
                    remaining[i] = 0;    }
                }

        }
        if (done) {
            break;
```

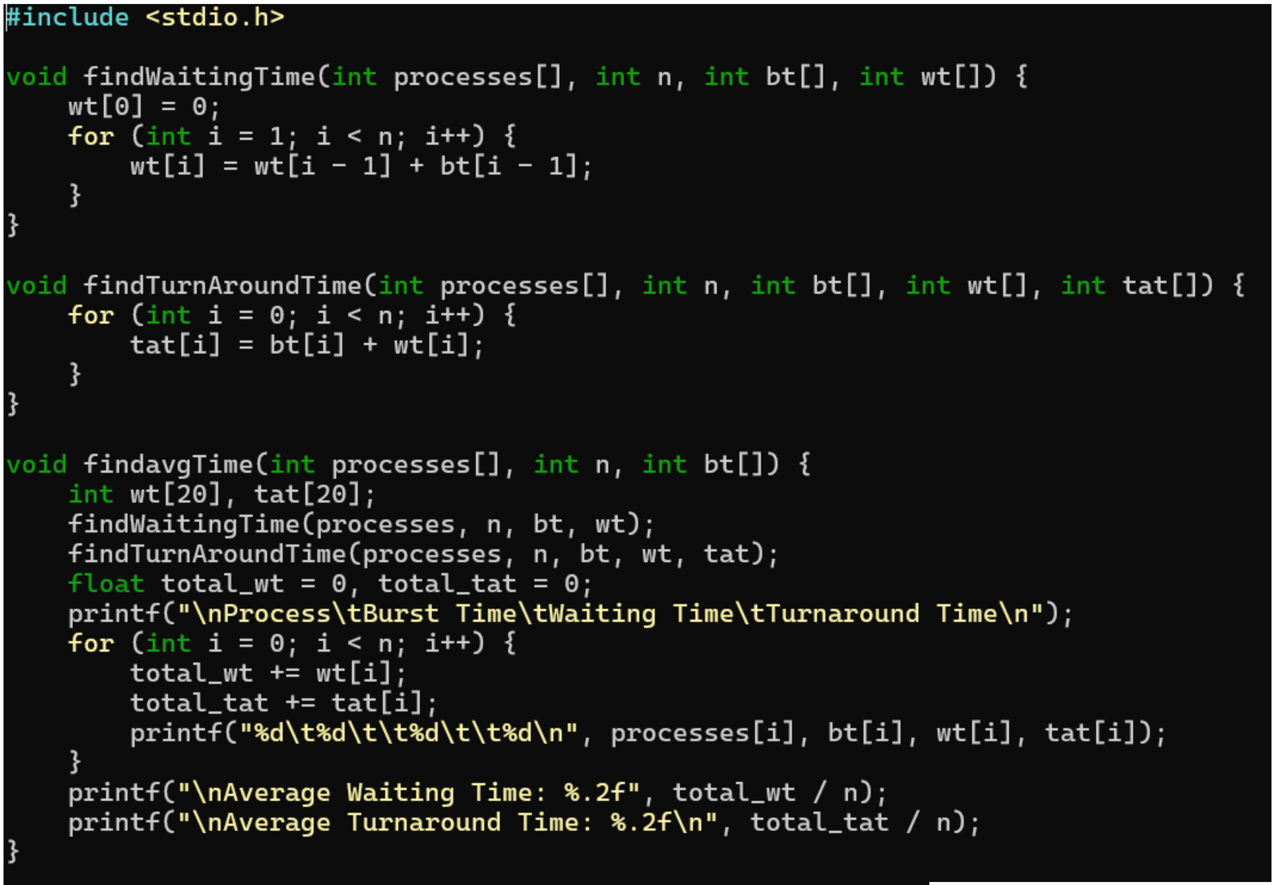


```
            break;
        }
    }
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (i = 0; i < n; i++) {
        avg_wt += wt[i];
        avg_tat += tat[i];
        printf("%d\t%d\t\t%d\t\t%d\n", i + 1, bt[i], wt[i], tat[i]);
    }
    avg_wt /= n;
    avg_tat /= n;
    printf("\nAverage Waiting Time: %.2f", avg_wt);
    printf("\nAverage Turnaround Time: %.2f\n", avg_tat);
    return 0;
}
```

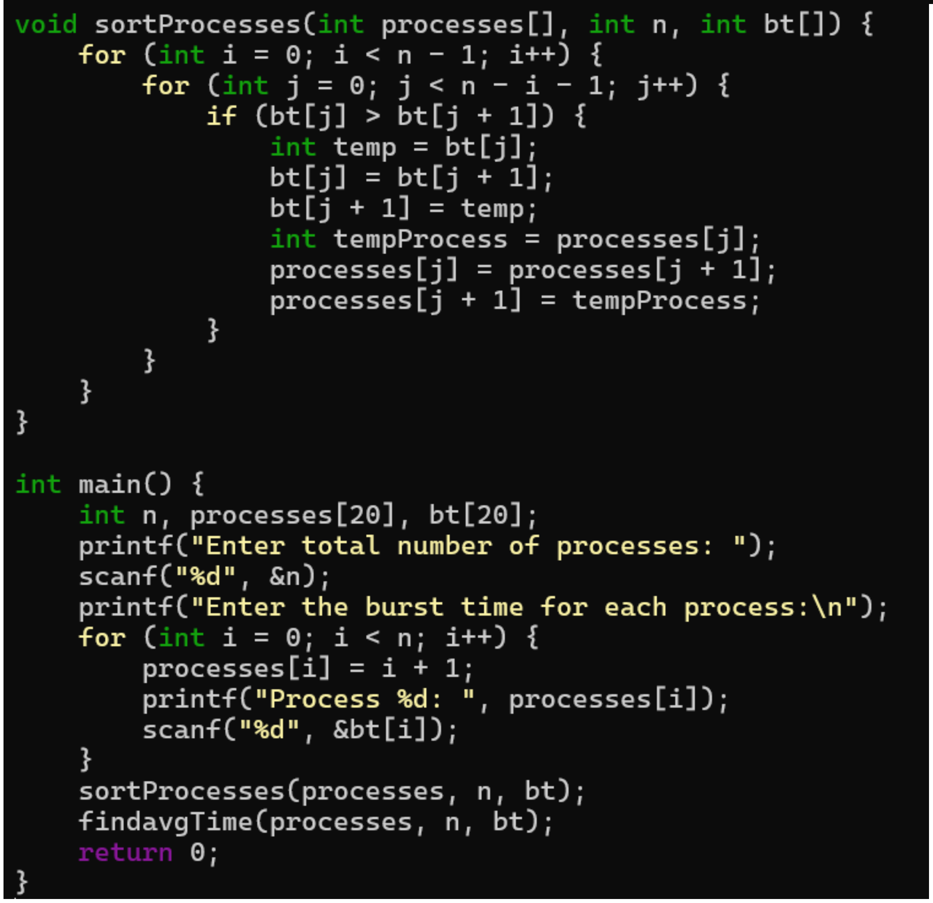


```
vikram@DESKTOP-S5P315A:~$ nano roundrobin.c
vikram@DESKTOP-S5P315A:~$ gcc roundrobin.c -o rr
vikram@DESKTOP-S5P315A:~$ ./rr
Enter total number of processes: 4
Enter time quantum: 2
Enter the burst time for each process:
Process 1: 8
Process 2: 9
Process 3: 4
Process 4: 3

Process Burst Time      Waiting Time    Turnaround Time
1       8               13              21
2       9               15              24
3       4               10              14
4       3               12              15

Average Waiting Time: 12.50
Average Turnaround Time: 18.50
vikram@DESKTOP-S5P315A:~$
```

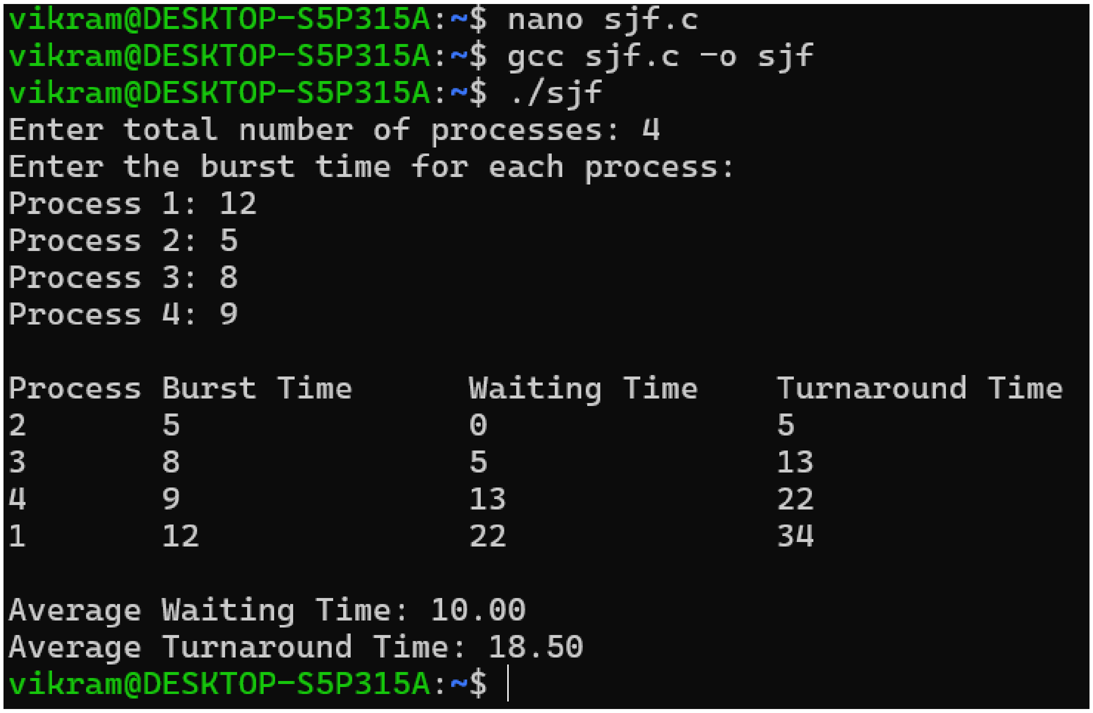
c. Shortest job first



```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    wt[0] = 0;
    for (int i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
    }
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}

void findavgTime(int processes[], int n, int bt[]) {
    int wt[20], tat[20];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);
    float total_wt = 0, total_tat = 0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("%d\t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i], tat[i]);
    }
    printf("\nAverage Waiting Time: %.2f", total_wt / n);
    printf("\nAverage Turnaround Time: %.2f\n", total_tat / n);
}
```

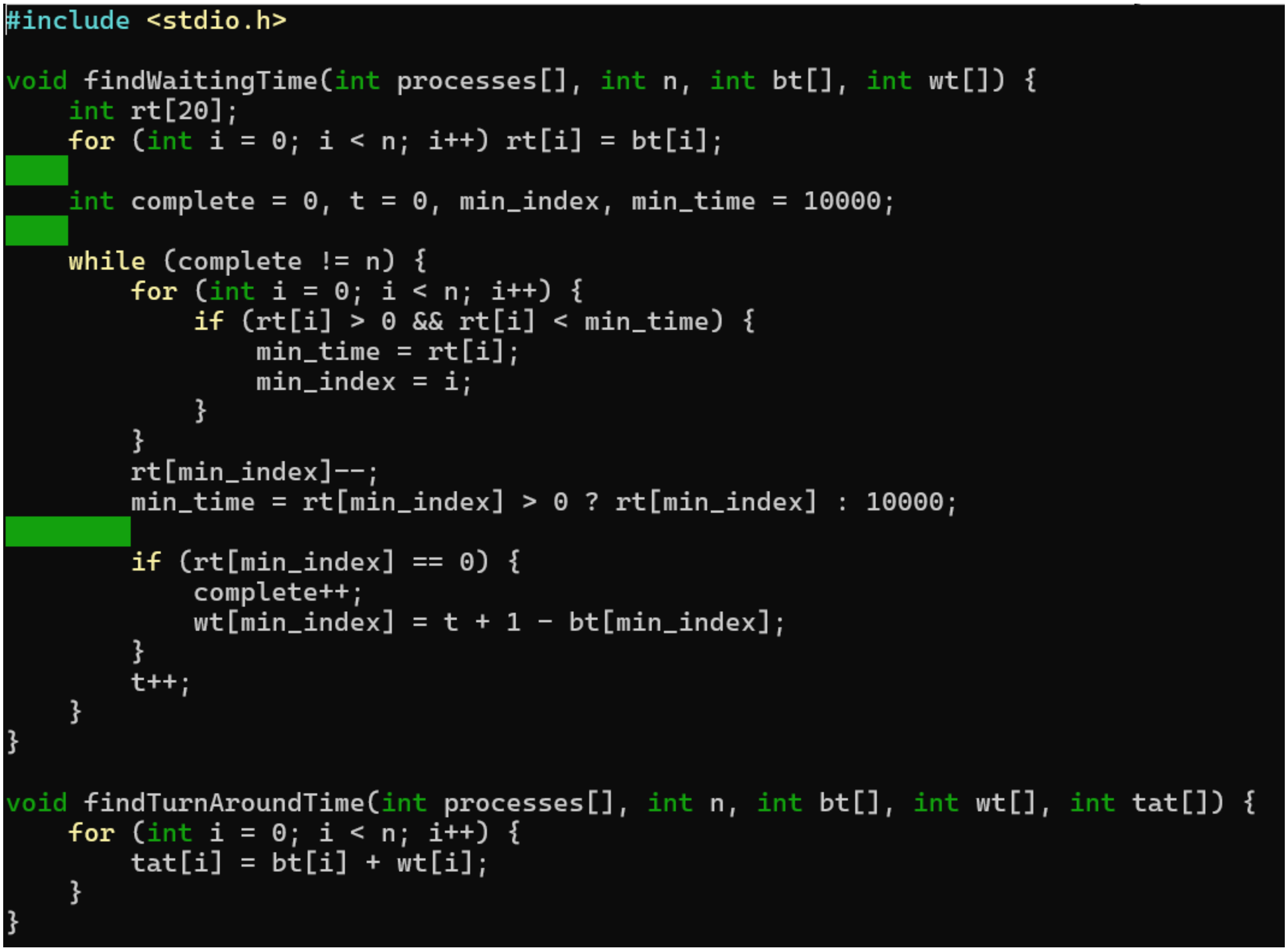


```
void sortProcesses(int processes[], int n, int bt[]) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (bt[j] > bt[j + 1]) {
                int temp = bt[j];
                bt[j] = bt[j + 1];
                bt[j + 1] = temp;
                int tempProcess = processes[j];
                processes[j] = processes[j + 1];
                processes[j + 1] = tempProcess;
            }
        }
    }
}

int main() {
    int n, processes[20], bt[20];
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    printf("Enter the burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Process %d: ", processes[i]);
        scanf("%d", &bt[i]);
    }
    sortProcesses(processes, n, bt);
    findavgTime(processes, n, bt);
    return 0;
}
```

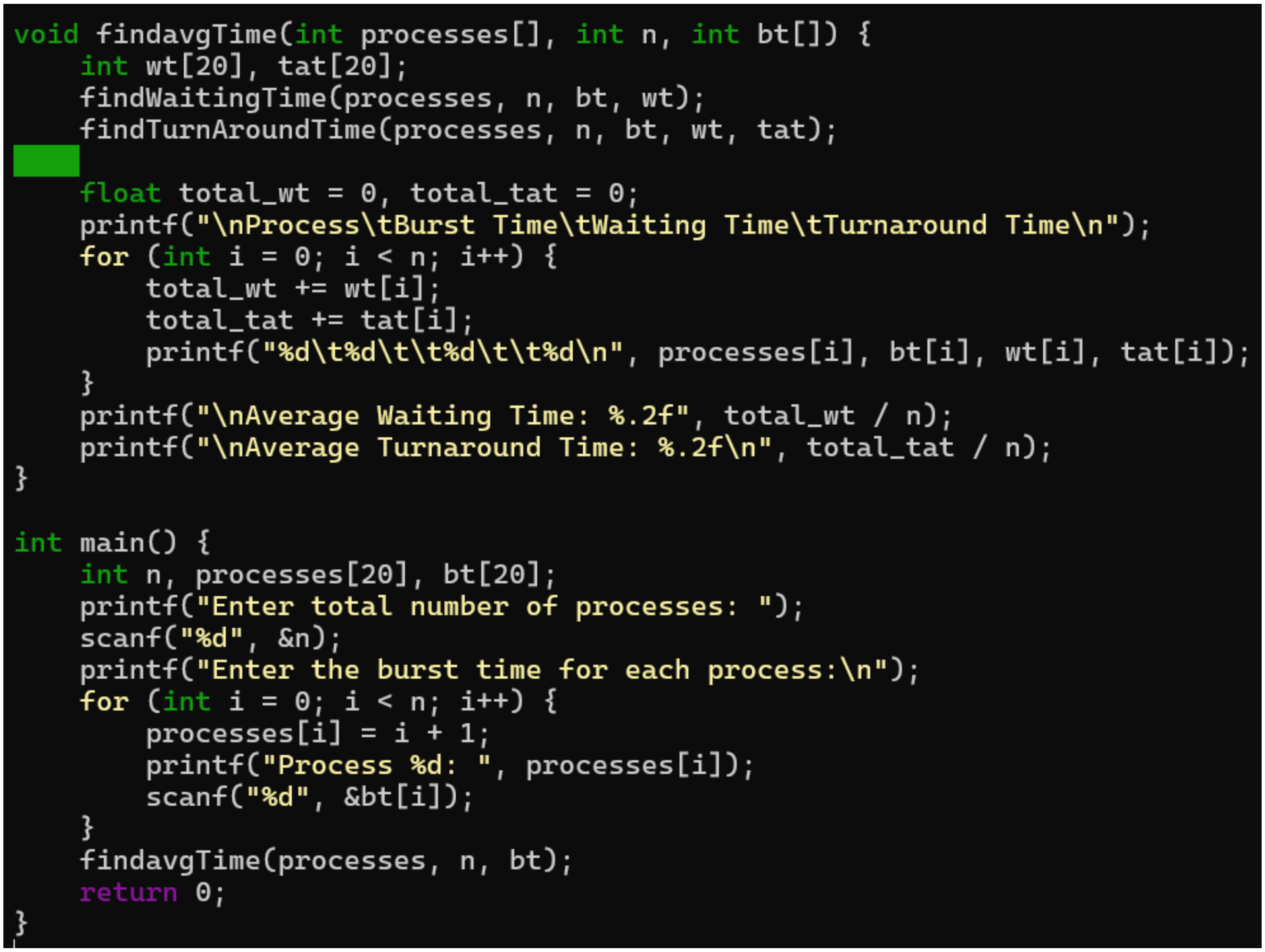


```
vikram@DESKTOP-S5P315A:~$ nano sjf.c
vikram@DESKTOP-S5P315A:~$ gcc sjf.c -o sjf
vikram@DESKTOP-S5P315A:~$ ./sjf
Enter total number of processes: 4
Enter the burst time for each process:
Process 1: 12
Process 2: 5
Process 3: 8
Process 4: 9

Process Burst Time      Waiting Time    Turnaround Time
2       5               0               5
3       8               5               13
4       9               13              22
1       12              22              34

Average Waiting Time: 10.00
Average Turnaround Time: 18.50
vikram@DESKTOP-S5P315A:~$ 
```

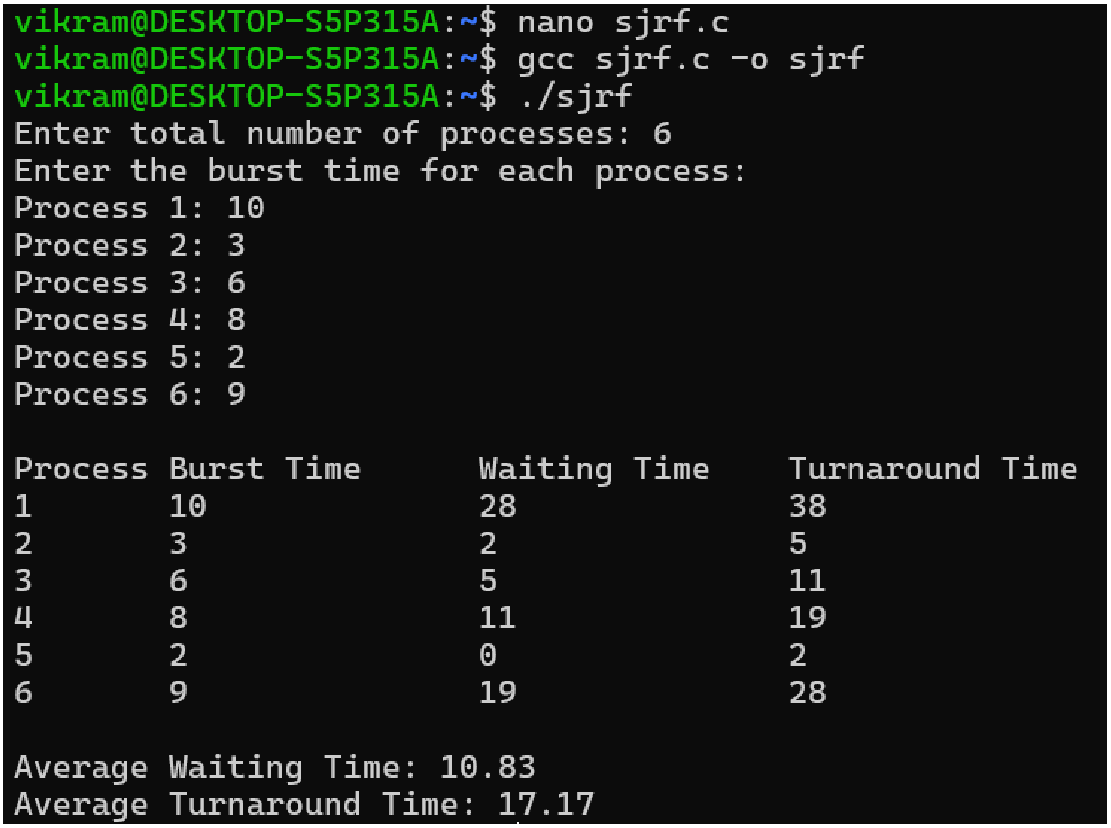
d. Shortest Job remaining first.



```
#include <stdio.h>

void findWaitingTime(int processes[], int n, int bt[], int wt[]) {
    int rt[20];
    for (int i = 0; i < n; i++) rt[i] = bt[i];

    int complete = 0, t = 0, min_index, min_time = 10000;

    while (complete != n) {
        for (int i = 0; i < n; i++) {
            if (rt[i] > 0 && rt[i] < min_time) {
                min_time = rt[i];
                min_index = i;
            }
        }
        rt[min_index]--;
        min_time = rt[min_index] > 0 ? rt[min_index] : 10000;

        if (rt[min_index] == 0) {
            complete++;
            wt[min_index] = t + 1 - bt[min_index];
        }
        t++;
    }
}

void findTurnAroundTime(int processes[], int n, int bt[], int wt[], int tat[]) {
    for (int i = 0; i < n; i++) {
        tat[i] = bt[i] + wt[i];
    }
}
```



```c
void findavgTime(int processes[], int n, int bt[]) {
    int wt[20], tat[20];
    findWaitingTime(processes, n, bt, wt);
    findTurnAroundTime(processes, n, bt, wt, tat);

    float total_wt = 0, total_tat = 0;
    printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
    for (int i = 0; i < n; i++) {
        total_wt += wt[i];
        total_tat += tat[i];
        printf("%d\t%d\t\t%d\t\t%d\n", processes[i], bt[i], wt[i], tat[i]);
    }
    printf("\nAverage Waiting Time: %.2f", total_wt / n);
    printf("\nAverage Turnaround Time: %.2f\n", total_tat / n);
}

int main() {
    int n, processes[20], bt[20];
    printf("Enter total number of processes: ");
    scanf("%d", &n);
    printf("Enter the burst time for each process:\n");
    for (int i = 0; i < n; i++) {
        processes[i] = i + 1;
        printf("Process %d: ", processes[i]);
        scanf("%d", &bt[i]);
    }
    findavgTime(processes, n, bt);
    return 0;
}
```



```
vikram@DESKTOP-S5P315A:~$ nano sjrf.c
vikram@DESKTOP-S5P315A:~$ gcc sjrf.c -o sjrf
vikram@DESKTOP-S5P315A:~$ ./sjrf
Enter total number of processes: 6
Enter the burst time for each process:
Process 1: 10
Process 2: 3
Process 3: 6
Process 4: 8
Process 5: 2
Process 6: 9

Process Burst Time      Waiting Time    Turnaround Time
1       10              28              38
2       3               2               5
3       6               5               11
4       8               11              19
5       2               0               2
6       9               19              28

Average Waiting Time: 10.83
Average Turnaround Time: 17.17
```

Learning Outcome:

Experiment 5

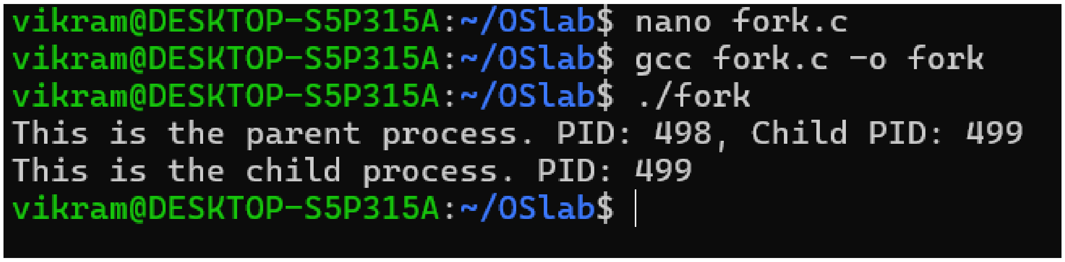
Aim: Process Management a) fork() b) execv() c) execlp() d) wait() and e) sleep()

Theory:

A. Program to implement the fork function using C.

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>

int main() {
    pid_t pid;
    pid = fork();

    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return 1;
    }
    else if (pid == 0) {
        printf("This is the child process. PID: %d\n", getpid());
    }
    else {
        printf("This is the parent process. PID: %d, Child PID: %d\n", getpid(), pid);
    }

    return 0;
}
```

OUTPUT

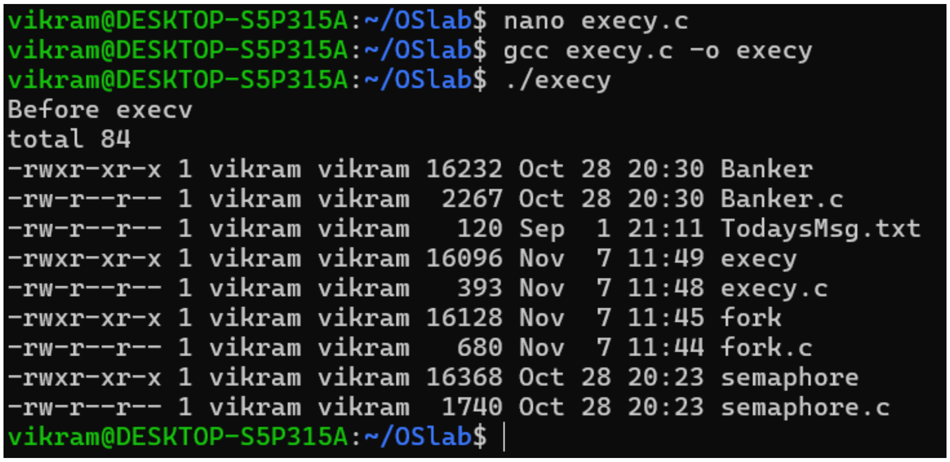


```
vikram@DESKTOP-S5P315A:~/OSlab$ nano fork.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc fork.c -o fork
vikram@DESKTOP-S5P315A:~/OSlab$ ./fork
This is the parent process. PID: 498, Child PID: 499
This is the child process. PID: 499
vikram@DESKTOP-S5P315A:~/OSlab$
```

B. Program to implement execv function using C.

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

int main() {
    char *args[] = {"/bin/ls", "-l", NULL};  // Arguments for execv

    printf("Before execv\n");

    // Execute the ls -l command using execv
    if (execv(args[0], args) == -1) {
        perror("execv failed");  // If execv fails, print an error
    }

    printf("This line will not be executed if execv succeeds\n");

    return 0;
}
```

OUTPUT

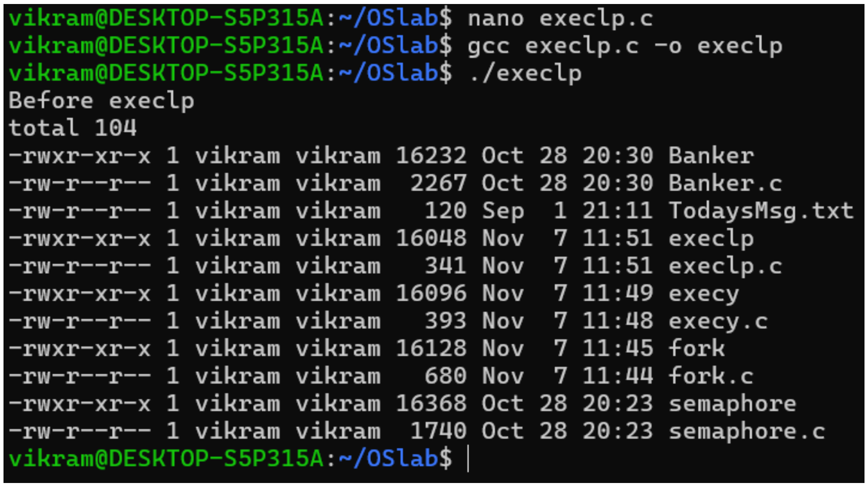


```
vikram@DESKTOP-S5P315A:~/OSlab$ nano execy.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc execy.c -o execy
vikram@DESKTOP-S5P315A:~/OSlab$ ./execy
Before execv
total 84
-rwxr-xr-x 1 vikram vikram 16232 Oct 28 20:30 Banker
-rw-r--r-- 1 vikram vikram  2267 Oct 28 20:30 Banker.c
-rw-r--r-- 1 vikram vikram   120 Sep  1 21:11 TodaysMsg.txt
-rwxr-xr-x 1 vikram vikram 16096 Nov  7 11:49 execy
-rw-r--r-- 1 vikram vikram   393 Nov  7 11:48 execy.c
-rwxr-xr-x 1 vikram vikram 16128 Nov  7 11:45 fork
-rw-r--r-- 1 vikram vikram   680 Nov  7 11:44 fork.c
-rwxr-xr-x 1 vikram vikram 16368 Oct 28 20:23 semaphore
-rw-r--r-- 1 vikram vikram  1740 Oct 28 20:23 semaphore.c
vikram@DESKTOP-S5P315A:~/OSlab$ |
```

C. Program to implement execlp function.

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

int main() {
    printf("Before execlp\n");
    if (execlp("ls", "ls", "-l", NULL) == -1) {
        perror("execlp failed");  // If execlp fails, print an error
    }
    printf("This line will not be executed if execlp succeeds\n");

    return 0;
}
```

OUTPUT

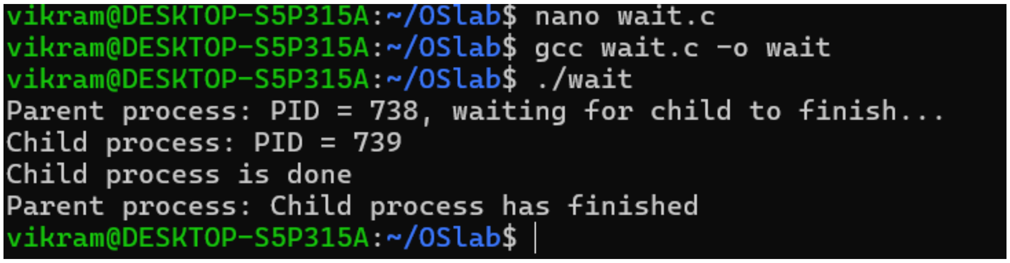


```
vikram@DESKTOP-S5P315A:~/OSlab$ nano execlp.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc execlp.c -o execlp
vikram@DESKTOP-S5P315A:~/OSlab$ ./execlp
Before execlp
total 104
-rwxr-xr-x 1 vikram vikram 16232 Oct 28 20:30 Banker
-rw-r--r-- 1 vikram vikram  2267 Oct 28 20:30 Banker.c
-rw-r--r-- 1 vikram vikram   120 Sep  1 21:11 TodaysMsg.txt
-rwxr-xr-x 1 vikram vikram 16048 Nov  7 11:51 execlp
-rw-r--r-- 1 vikram vikram   341 Nov  7 11:51 execlp.c
-rwxr-xr-x 1 vikram vikram 16096 Nov  7 11:49 execy
-rw-r--r-- 1 vikram vikram   393 Nov  7 11:48 execy.c
-rwxr-xr-x 1 vikram vikram 16128 Nov  7 11:45 fork
-rw-r--r-- 1 vikram vikram   680 Nov  7 11:44 fork.c
-rwxr-xr-x 1 vikram vikram 16368 Oct 28 20:23 semaphore
-rw-r--r-- 1 vikram vikram  1740 Oct 28 20:23 semaphore.c
vikram@DESKTOP-S5P315A:~/OSlab$
```

D. Program to implement wait function using C.

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

int main() {
    pid_t pid;

    // Create a new process
    pid = fork();

    if (pid < 0) {
        // If fork() returns a negative value, creation of child process was unsuccessful
        fprintf(stderr, "Fork failed\n");
        return 1;
    }
    else if (pid == 0) {
        // Child process
        printf("Child process: PID = %d\n", getpid());
        sleep(2);  // Simulate some work in the child process
        printf("Child process is done\n");
    }
    else {
        // Parent process
        printf("Parent process: PID = %d, waiting for child to finish...\n", getpid());
        wait(NULL);  // Wait for the child process to finish
        printf("Parent process: Child process has finished\n");
    }

    return 0;
}
```

OUTPUT

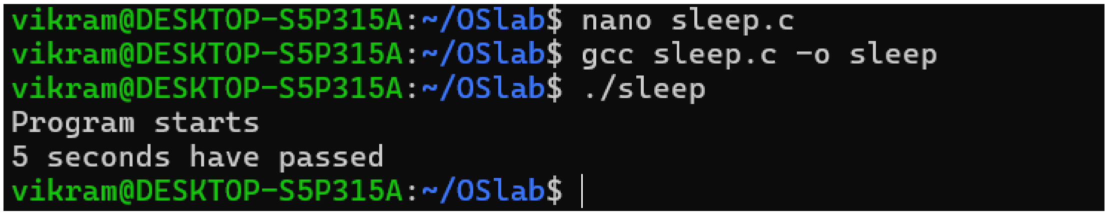


```
vikram@DESKTOP-S5P315A:~/OSlab$ nano wait.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc wait.c -o wait
vikram@DESKTOP-S5P315A:~/OSlab$ ./wait
Parent process: PID = 738, waiting for child to finish...
Child process: PID = 739
Child process is done
Parent process: Child process has finished
vikram@DESKTOP-S5P315A:~/OSlab$ |
```

E. Program to implement sleep function using C.

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

int main() {
    printf("Program starts\n");
    // Pause the program for 5 seconds
    sleep(5);
    printf("5 seconds have passed\n");
    return 0;
}
```

OUTPUT



```
vikram@DESKTOP-S5P315A:~/OSlab$ nano sleep.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc sleep.c -o sleep
vikram@DESKTOP-S5P315A:~/OSlab$ ./sleep
Program starts
5 seconds have passed
vikram@DESKTOP-S5P315A:~/OSlab$ |
```

Learning Outcome

Experiment 6

Aim: Write a program to implement reader/writer problems using semaphore.

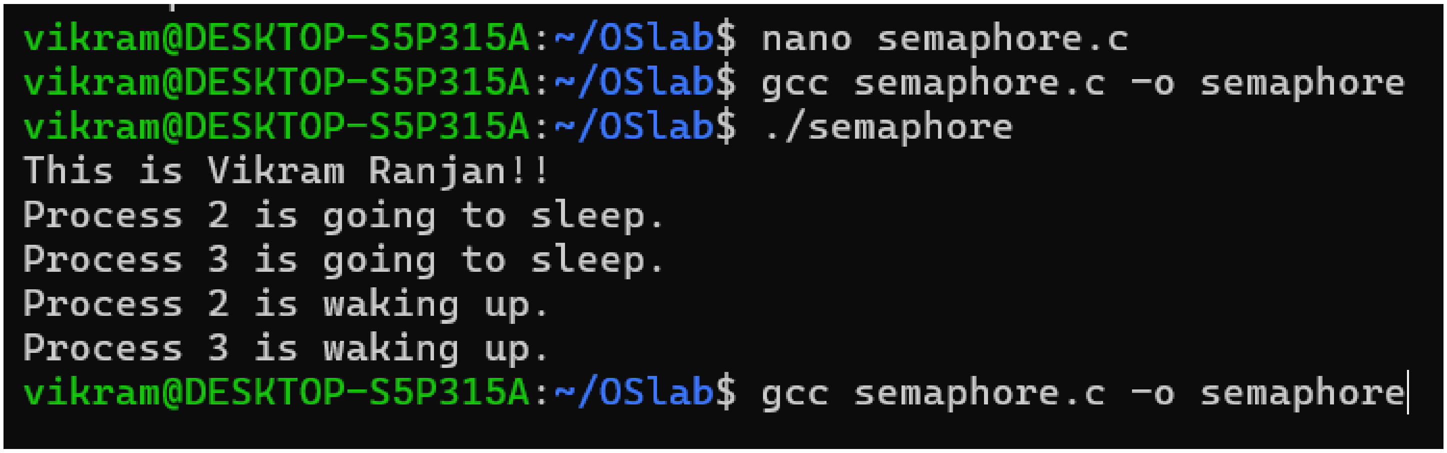
Theory:

Program

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

typedef int process;

struct node {
    process p;
    struct node* next;
};

struct Queue {
    struct node* front;
    struct node* rear;
};

void initQueue(struct Queue* q) {
    q->front = NULL;
    q->rear = NULL;
}

void enqueue(struct Queue* q, process p) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->p = p;
    newNode->next = NULL;

    if (q->rear == NULL) {
        q->front = q->rear = newNode;
    } else {
        q->rear->next = newNode;
        q->rear = newNode;
    }
}

int isEmpty(struct Queue* q) {
    return (q->front == NULL);
}

process dequeue(struct Queue* q) {
    if (isEmpty(q)) {
        return -1;
    }
    struct node* temp = q->front;
    process p = temp->p;
    q->front = q->front->next;

    if (q->front == NULL) {
        q->rear = NULL;
    }

    free(temp);
    return p;
}

struct semaphore {
    struct Queue q;
    int value;
};

void P(struct semaphore* s, process currentProcess) {
    if (s->value == 1) {
        s->value = 0;
    } else {
        enqueue(&s->q, currentProcess);
        printf("Process %d is going to sleep.\n", currentProcess);
        sleep(1);
    }
}

void V(struct semaphore* s) {
    if (isEmpty(&s->q)) {
        s->value = 1;
    } else {
        process p = dequeue(&s->q);
        if (p != -1) {
            printf("Process %d is waking up.\n", p);
        }
    }
}
```

```
int main() {
    printf("This is Vikram Ranjan!!\n");

    struct semaphore sem;
    sem.value = 1;
    initQueue(&sem.q);

    process p1 = 1;
    process p2 = 2;
    process p3 = 3;

    P(&sem, p1);
    P(&sem, p2);
    P(&sem, p3);

    V(&sem);
    V(&sem);
    V(&sem);

    return 0;
}
```

Output



```
vikram@DESKTOP-S5P315A:~/OSlab$ nano semaphore.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc semaphore.c -o semaphore
vikram@DESKTOP-S5P315A:~/OSlab$ ./semaphore
This is Vikram Ranjan!!
Process 2 is going to sleep.
Process 3 is going to sleep.
Process 2 is waking up.
Process 3 is waking up.
vikram@DESKTOP-S5P315A:~/OSlab$ gcc semaphore.c -o semaphore
```

Learning Outcome:

Experiment 7

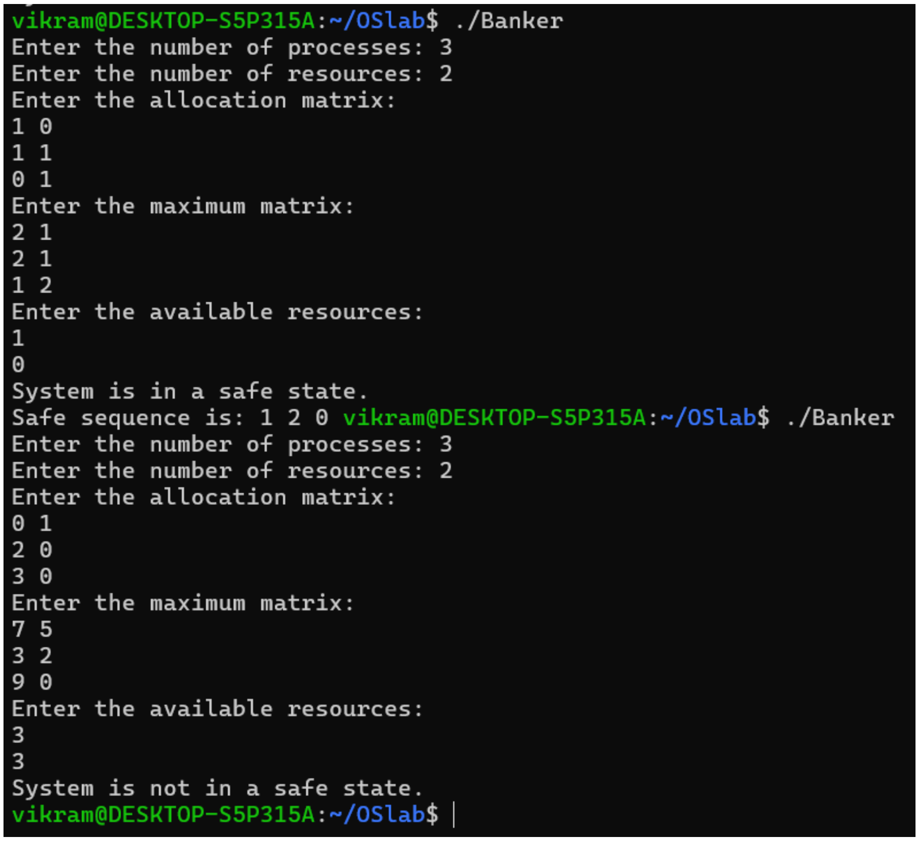
Aim: Write a program to implement Banker’s algorithm for deadlock avoidance.

Theory:

Program

```
#include <stdio.h>

int n, m;

int checkSafeState(int available[], int max[][10], int allocation[][10], int need[][10], int safeSequence[]) {
    int work[10], finish[10] = {0};
    int count = 0;

    for (int i = 0; i < m; i++) {
        work[i] = available[i];
    }

    while (count < n) {
        int found = 0;
        for (int i = 0; i < n; i++) {
            if (finish[i] == 0) {
                int j;
                for (j = 0; j < m; j++) {
                    if (need[i][j] > work[j]) {
                        break;
                    }
                }

                if (j == m) {
                    for (int k = 0; k < m; k++) {
                        work[k] += allocation[i][k];
                    }
                    safeSequence[count++] = i;
                    finish[i] = 1;
                    found = 1;
                }
            }
        }

        if (!found) {
            return 0;
        }
    }

    return 1;
}

void calculateNeed(int max[][10], int allocation[][10], int need[][10]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            need[i][j] = max[i][j] - allocation[i][j];
        }
    }
}

int main() {
    int allocation[10][10], max[10][10], available[10], need[10][10];
    int safeSequence[10];

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

    printf("Enter the allocation matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &allocation[i][j]);
        }
    }

    printf("Enter the maximum matrix:\n");
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            scanf("%d", &max[i][j]);
        }
    }

    printf("Enter the available resources:\n");
    for (int i = 0; i < m; i++) {
        scanf("%d", &available[i]);
    }

    calculateNeed(max, allocation, need);
    if (checkSafeState(available, max, allocation, need, safeSequence)) {
        printf("System is in a safe state.\nSafe sequence is: ");
        for (int i = 0; i < n; i++) {
            printf("%d ", safeSequence[i]);
        }
    } else {
        printf("System is not in a safe state.\n");
    }

    return 0;
}
```

Output:



```
vikram@DESKTOP-S5P315A:~/OSlab$ ./Banker
Enter the number of processes: 3
Enter the number of resources: 2
Enter the allocation matrix:
1 0
1 1
0 1
Enter the maximum matrix:
2 1
2 1
1 2
Enter the available resources:
1
0
System is in a safe state.
Safe sequence is: 1 2 0 vikram@DESKTOP-S5P315A:~/OSlab$ ./Banker
Enter the number of processes: 3
Enter the number of resources: 2
Enter the allocation matrix:
0 1
2 0
3 0
Enter the maximum matrix:
7 5
3 2
9 0
Enter the available resources:
3
3
System is not in a safe state.
vikram@DESKTOP-S5P315A:~/OSlab$
```

Learning Outcome:

Experiment 8

Aim: Implementation of the following Memory Allocation Methods for fixed partition

a. First Fit

b. Worst Fit

c. Best Fit.

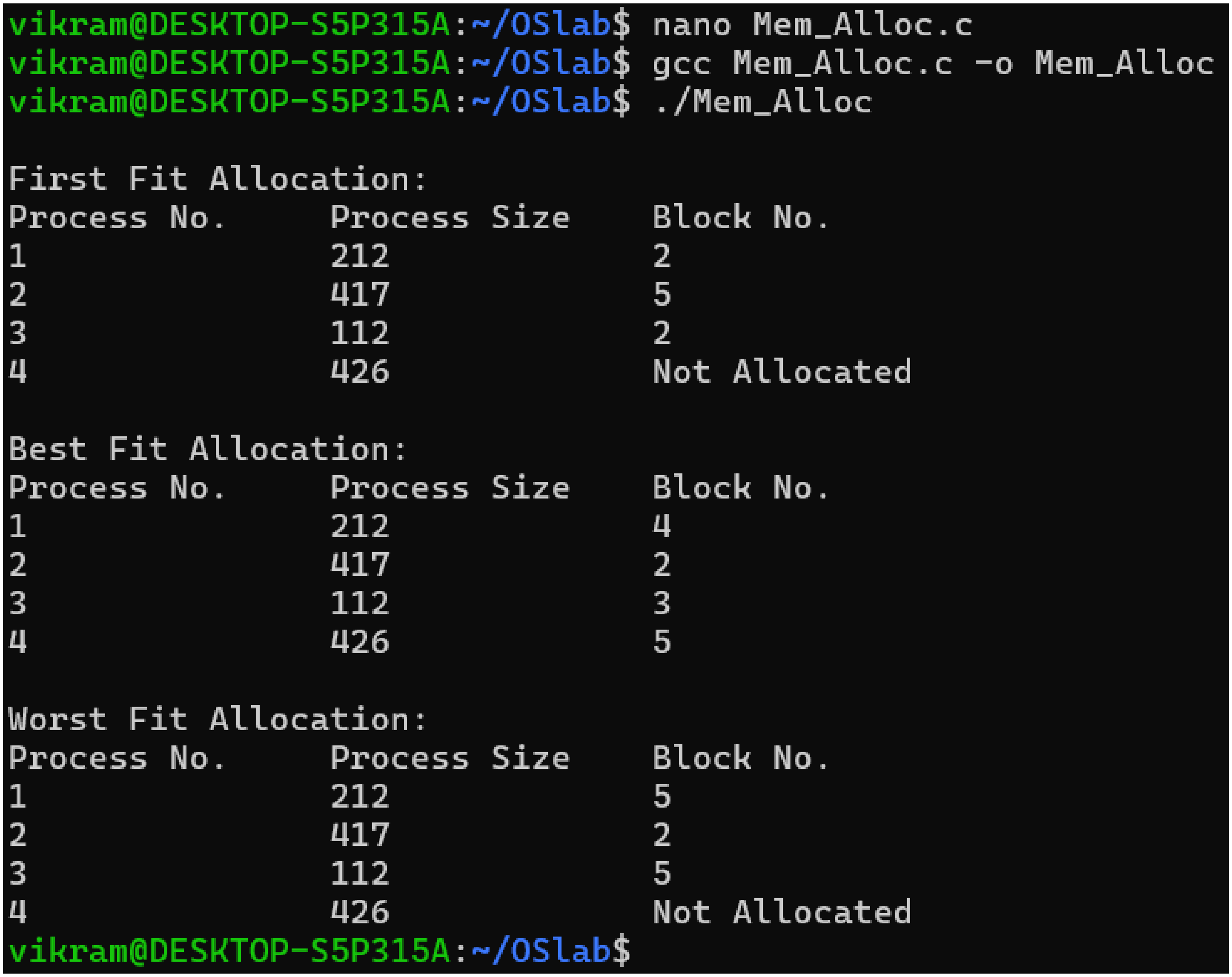
Theory:

Program

```
#include <stdio.h>

void firstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++) {
        allocation[i] = -1; // Initially no block is assigned to any process
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                allocation[i] = j;       // Assign block j to process i
                blockSize[j] -= processSize[i];  // Reduce available memory in this block
                break;
            }
        }
    }

    printf("\nFirst Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
```

```
}

void bestFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++) {
        allocation[i] = -1; // Initially no block is assigned to any process
    }

    for (int i = 0; i < n; i++) {
        int bestIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (bestIdx == -1 || blockSize[j] < blockSize[bestIdx]) {
                    bestIdx = j;
                }
            }
        }

        if (bestIdx != -1) {
            allocation[i] = bestIdx;
            blockSize[bestIdx] -= processSize[i];
        }
    }

    printf("\nBest Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

void worstFit(int blockSize[], int m, int processSize[], int n) {
    int allocation[n];
    for (int i = 0; i < n; i++) {
        allocation[i] = -1; // Initially no block is assigned to any process
    }

    for (int i = 0; i < n; i++) {
        int worstIdx = -1;
        for (int j = 0; j < m; j++) {
            if (blockSize[j] >= processSize[i]) {
                if (worstIdx == -1 || blockSize[j] > blockSize[worstIdx]) {
                    worstIdx = j;
                }
            }
        }
        if (worstIdx != -1) {
            allocation[i] = worstIdx;
            blockSize[worstIdx] -= processSize[i];
        }
    }

    printf("\nWorst Fit Allocation:\n");
    printf("Process No.\tProcess Size\tBlock No.\n");
    for (int i = 0; i < n; i++) {
        printf("%d\t\t%d\t\t", i + 1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n", allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}

int main() {
    int blockSize[] = {100, 500, 200, 300, 600};
    int processSize[] = {212, 417, 112, 426};
    int m = sizeof(blockSize) / sizeof(blockSize[0]);
    int n = sizeof(processSize) / sizeof(processSize[0]);

    int blockSize1[m], blockSize2[m], blockSize3[m];

    // Copy the original block sizes for each method
    for (int i = 0; i < m; i++) {
        blockSize1[i] = blockSize[i];
        blockSize2[i] = blockSize[i];
        blockSize3[i] = blockSize[i];
    }

    firstFit(blockSize1, m, processSize, n);
    bestFit(blockSize2, m, processSize, n);
    worstFit(blockSize3, m, processSize, n);

    return 0;
}
```

OUTPUT



```
vikram@DESKTOP-S5P315A:~/OSlab$ nano Mem_Alloc.c
vikram@DESKTOP-S5P315A:~/OSlab$ gcc Mem_Alloc.c -o Mem_Alloc
vikram@DESKTOP-S5P315A:~/OSlab$ ./Mem_Alloc

First Fit Allocation:
Process No.     Process Size    Block No.
1               212             2
2               417             5
3               112             2
4               426             Not Allocated

Best Fit Allocation:
Process No.     Process Size    Block No.
1               212             4
2               417             2
3               112             3
4               426             5

Worst Fit Allocation:
Process No.     Process Size    Block No.
1               212             5
2               417             2
3               112             5
4               426             Not Allocated
vikram@DESKTOP-S5P315A:~/OSlab$
```

Learning Outcome