****

**VIVEKANANDA INSTITUTE OF PROFESSIONAL STUDIES - TECHNICAL CAMPUS**

**Grade A++ Accredited Institution by NAAC**

NBA Accredited for MCA Programme; Recognized under Section 2(f) by UGC;

Affiliated to GGSIP University, Delhi; Recognized by Bar Council of India and AICTE

An ISO 9001:2015 Certified Institution

**SCHOOL OF ENGINEERING & TECHNOLOGY**

**BTECH Programme: AI & DS – A [G-1]**

**Course Title: Operating Systems Lab**

**Course Code: AIDS351**

**Submitted To : Submitted By :**

**Dr. Nihar Ranjan Roy Name:** Prachi Gupta

**Enrollment No:** 02217711922

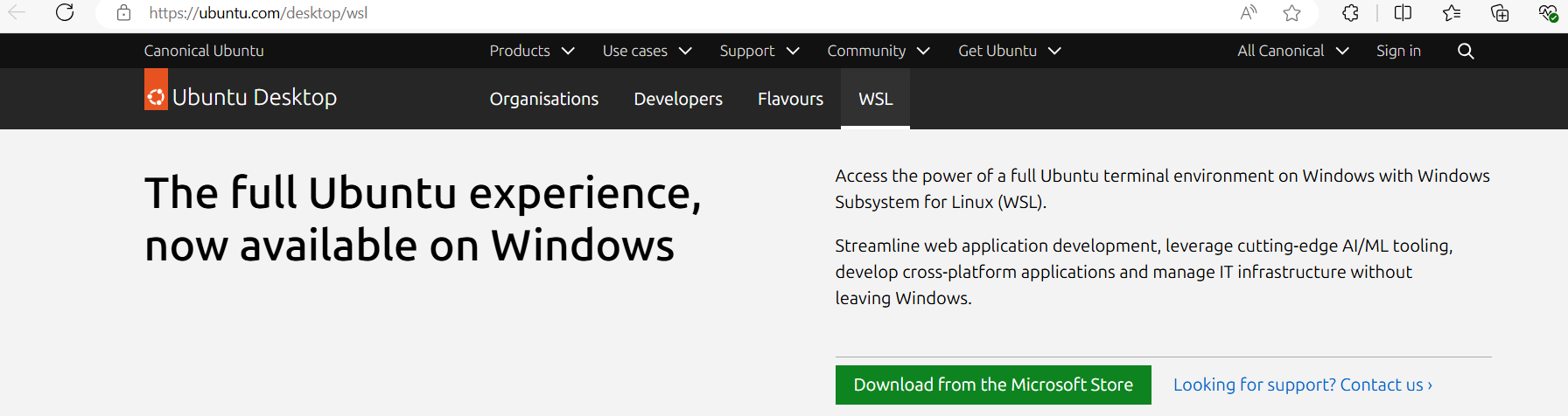
Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXPERIMENT-1**

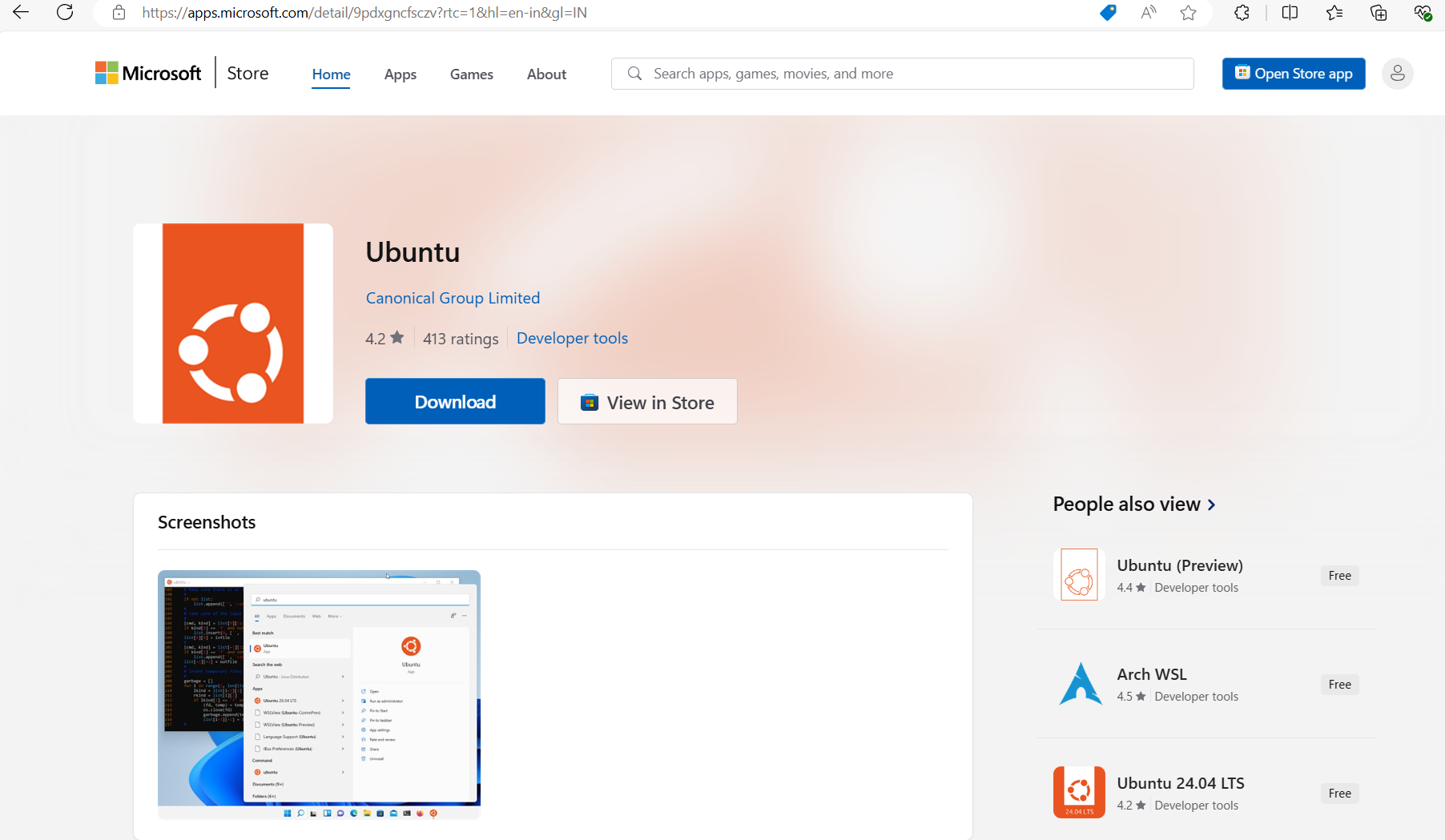
**Experiment 1:** Install Ubuntu on windows and, compile and run first C program using gcc.

**Install Ubuntu Terminal Environment on Windows with Windows Subsystem for Linux(WSL)**

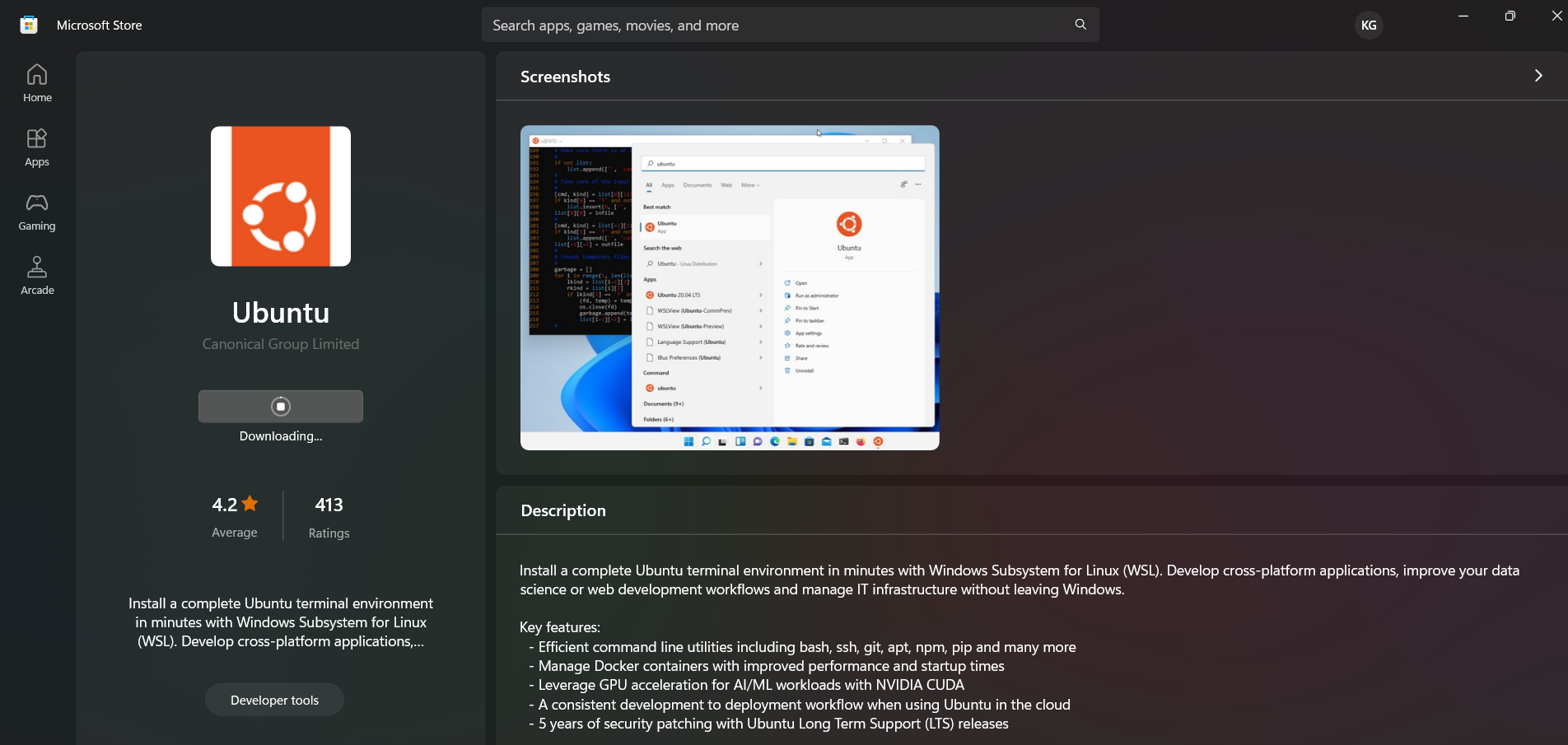
1. Go to following URL https://ubuntu.com/wsl and click download from the Microsoft Store

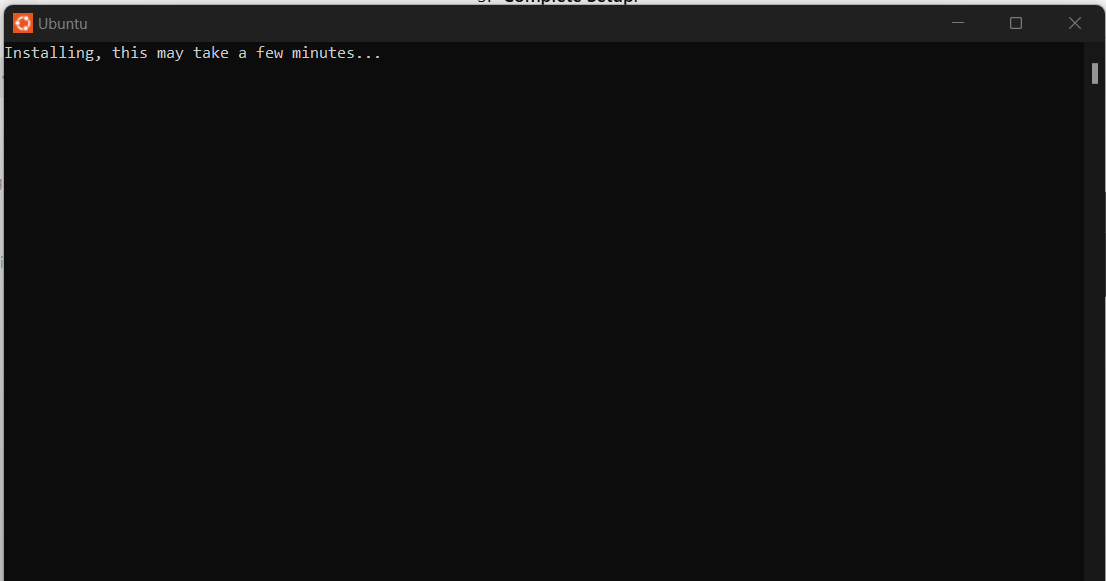


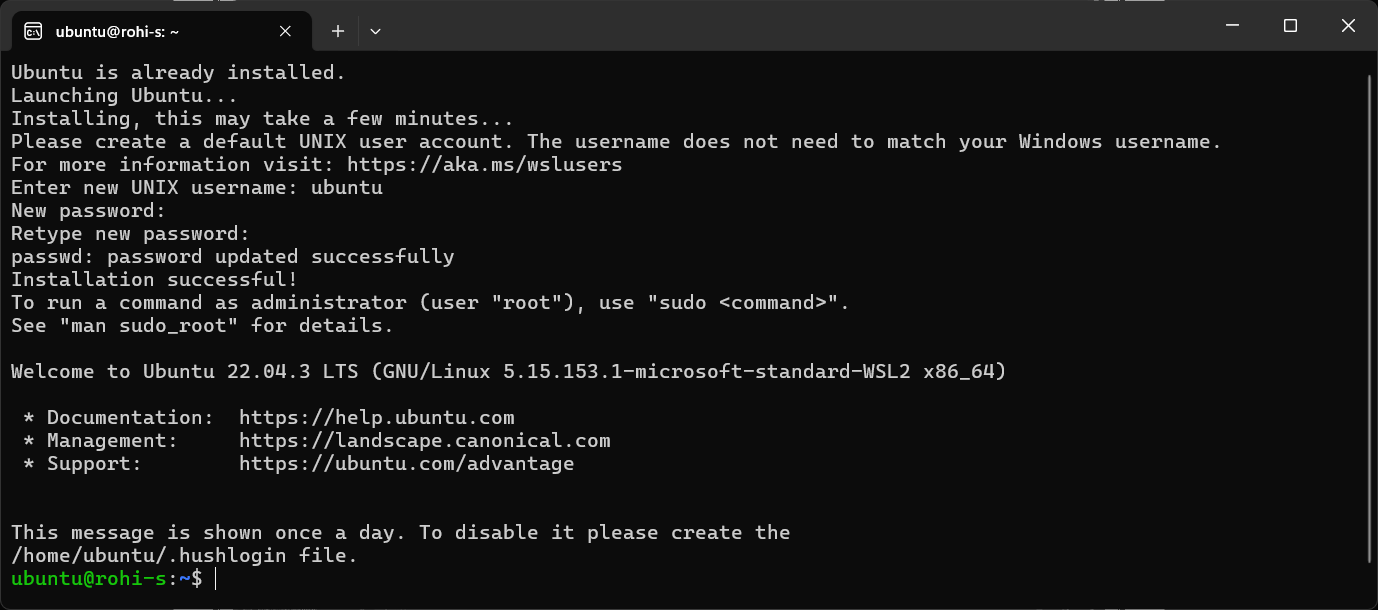
2. It will take you to https://apps.microsoft.com/store/detail/ubuntu/9PDXGNCFSCZV?hl=en-in&gl=in&rtc=1 then click get in Store app



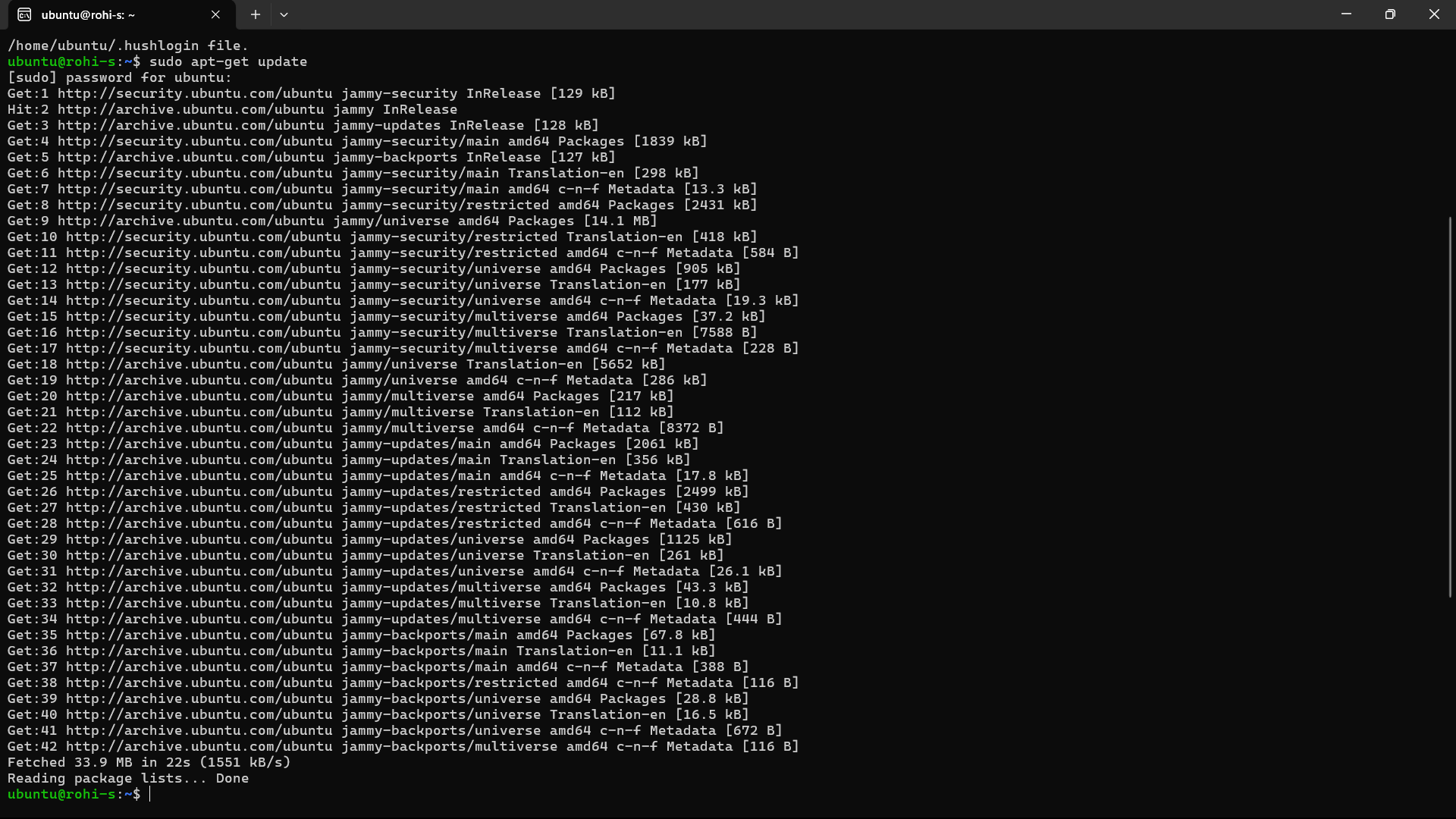
3. Click open Microsoft Store

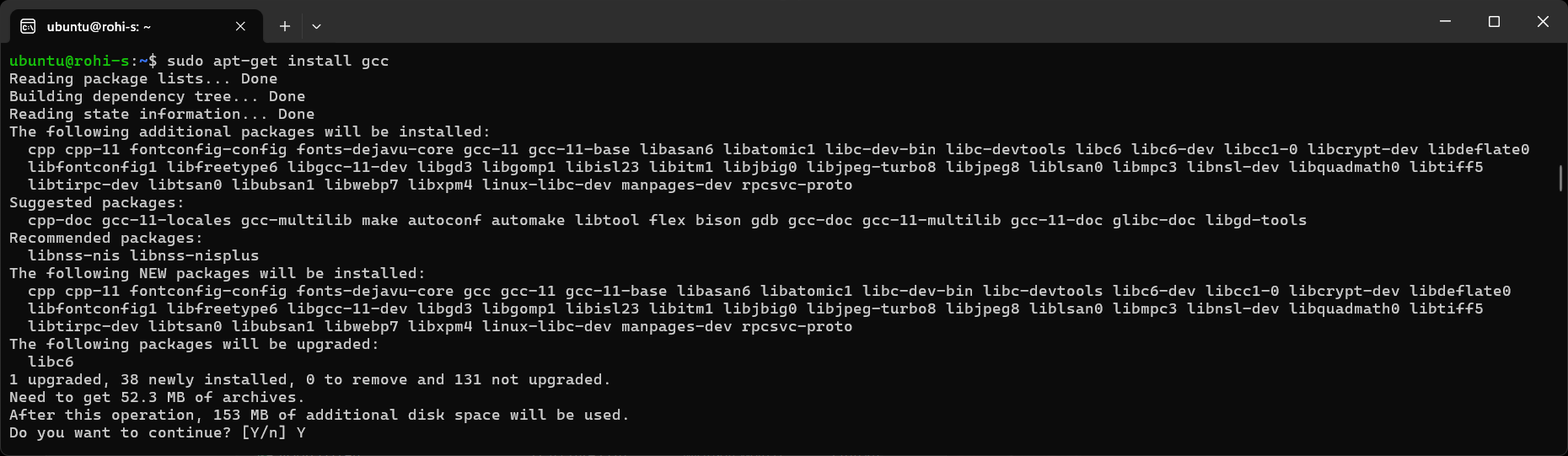




4. Setting up username and password. After installation it will ask for a new username and password

**Updating to latest repository and installing gcc**

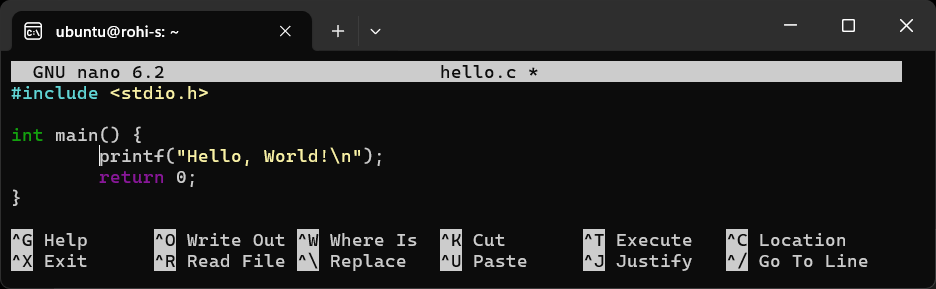
1. Update the system for latest repository list using following command: **sudo apt-get update**

2. Install gcc using following command: **sudo apt-get install gcc**3.Press Y to continue

**Write your first C programme, compile and run Open ubuntu terminal and write the code using any editor, we are going to use nano. Use the following command to open a file named hello.c**

1. Open a file: **nano hello.c**



****

2. Compile the file [here -o option is for giving output file name]: **gcc hello.c -o hello**



3. Run the output file: **./hello**



**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXPERIMENT-2**

**Experiment 2:** Open ubuntu terminal and write the command for following operations and share the output screen.

**a) Command to know your current working directory**



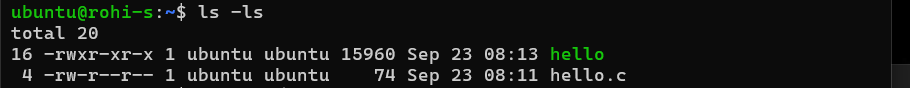
This command will display the path of your current working directory.

**b) List all the files in the current directory**



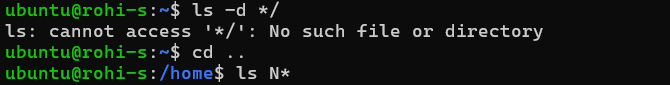
This will list all files and directories in the current working directory.

**c) List all the files in the order of their file size**



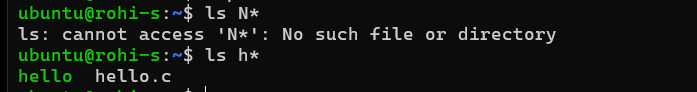
The -l option lists details and -S sorts by file size.

**d) List only directories in the current folder**



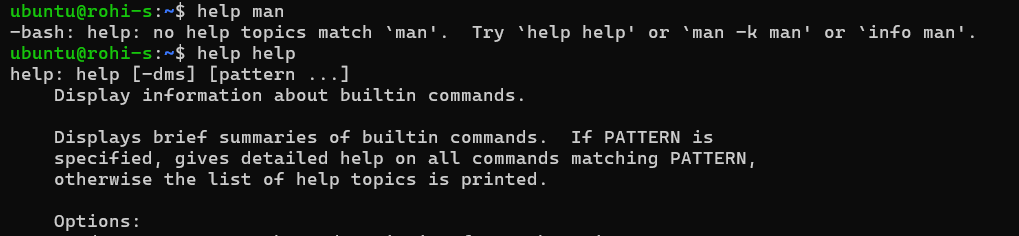
The -d \*/ command shows only directories in the current folder.

**e) List only files starting with “N” alphabet**



This will list only the files that start with the letter "N".

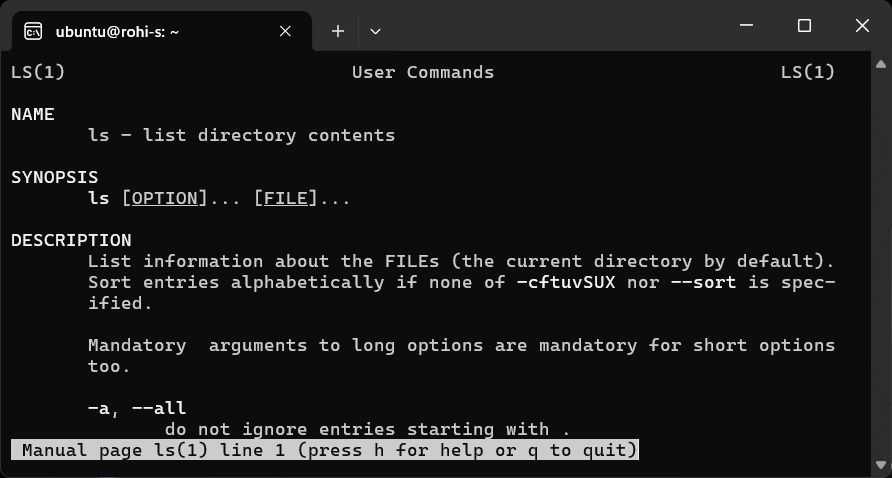
**f) Using help command find the help on ‘man’ command.**



This command shows help on the man command.

**g) Display the content of manual pages on ‘ls’ command**





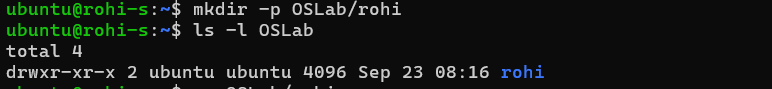
This displays the manual page for the ls command.

**h) Demonstrate the usage of “whatis” command.**



This command provides a brief description of the ls command.

**i) Make directory named “OSLab/<yourname>”.**



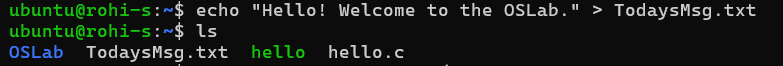
This command creates a directory named OSLab and a subdirectory with your name.

**j) Write command to reach to “yourname” Directory.**



This command changes the current working directory to OSLab/<yourname>.

**k) Create a txt file named “TodaysMsg.txt” and write a greeting message in it.**



This command creates a file TodaysMsg.txt and writes a greeting message in it.

**l) Copy this file “TodaysMsg.txt” to OSLab directory**



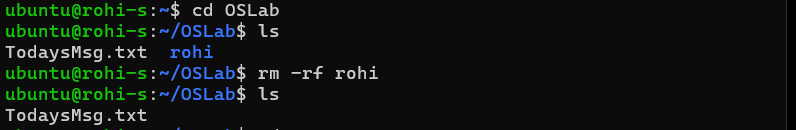
This command copies the TodaysMsg.txt file to the OSLab directory.

**m) Delete the file “TodaysMsg.txt” from the ‘yourname’ Folder**



This deletes the TodaysMsg.txt file from ‘yourname’ Folder

**n) Delete the directory ‘yourname’**



To remove the directory and its contents automatically, we use the **-f** (force) option along with **-r** (recursive). **Be cautious**: Using -f will not prompt for confirmation and will delete everything specified.

After running the command, the Kirty directory is removed from the OSLab directory.

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



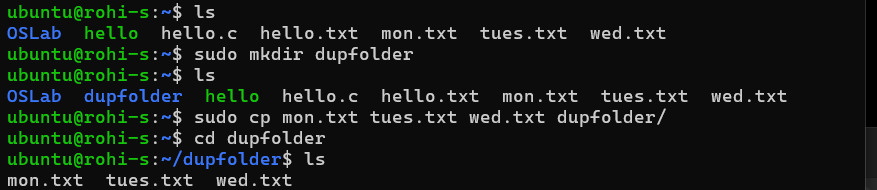
This creates Hello.txt with the given message.

**p) Using touch command create files with names mon.txt, tues.txt, and wed.txt**



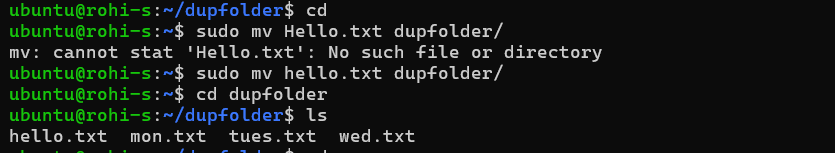
This command creates three empty files: mon.txt, tues.txt, and wed.txt.

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



This creates a folder named dupfolder and copies the three files into it.

**r) Move Hello.txt to dupfolder**



This command moves the Hello.txt file to the dupfolder.

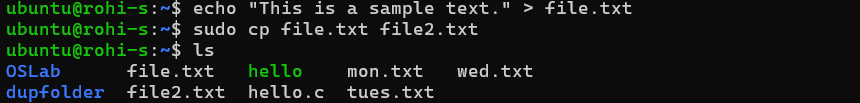
**s) Count number of words in the Hello.txt file**



This counts the number of words in the Hello.txt file inside dupfolder.

**t) Create two files with identical content, change one alphabet in one of these and compare them using cmp command.**

i. Create two files with identical content:

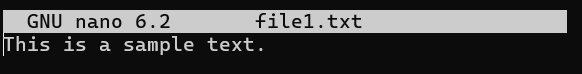
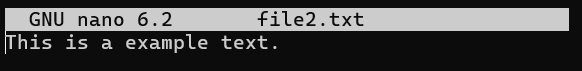


ii. Change one alphabet in file2.txt:



iii. This will compare the two files and point out the difference.



**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

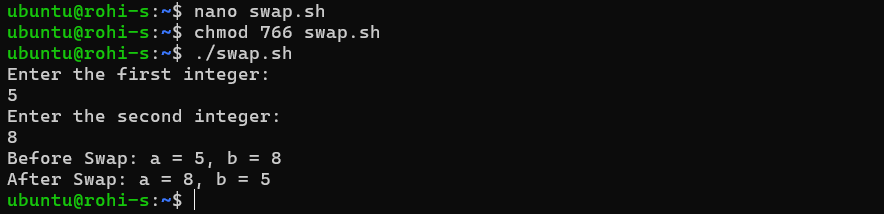
Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXPERIMENT-3**

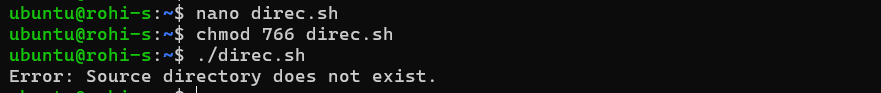
**Experiment 3:** Perform following shell script based programs.

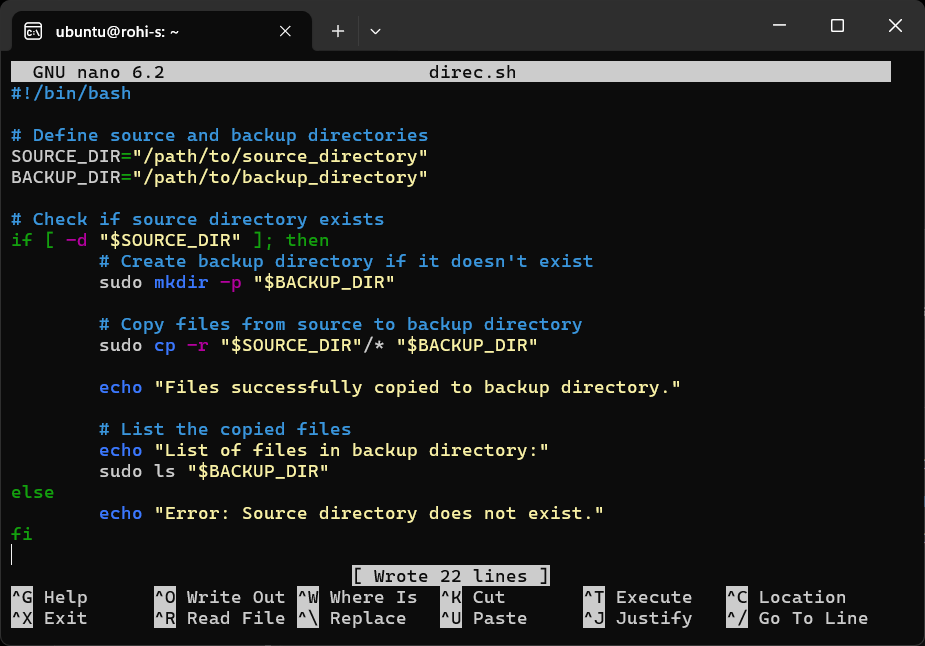
**a. Write a Shell Program to swap the two integers.**



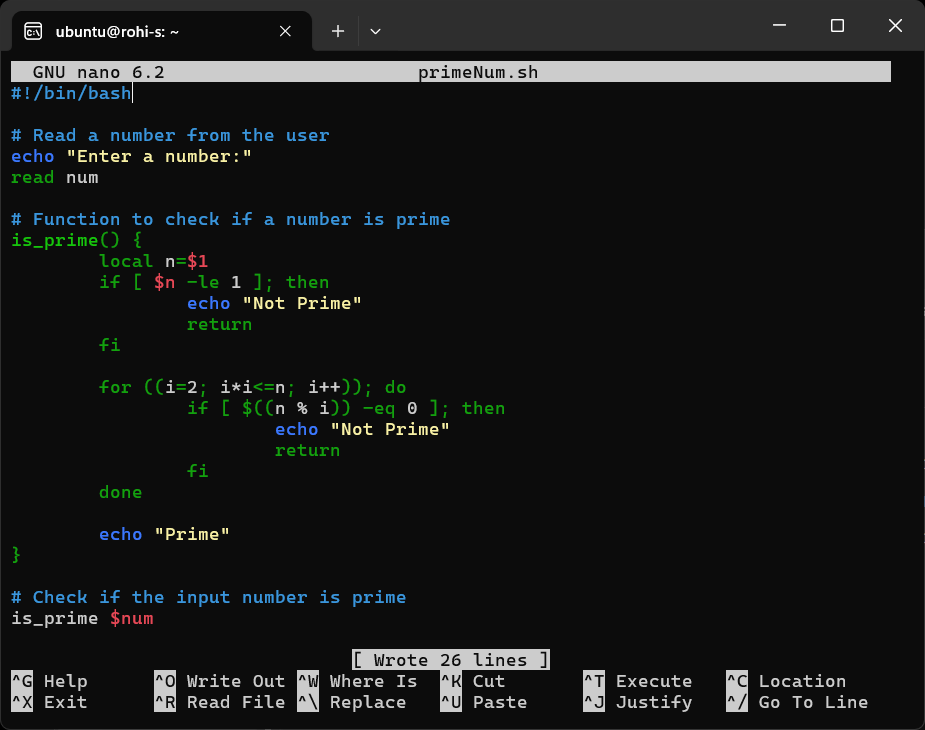


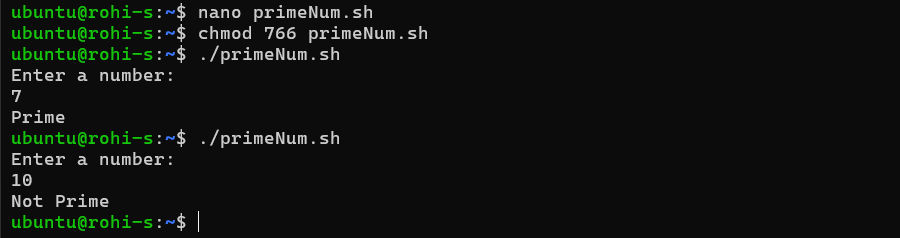
**b. Create a shell script that checks if a specific directory exists. If it does, the script should back up all files from that directory into a specified backup directory. The script should then loop through the files in the backup directory and list all files that were successfully copied. If the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

****

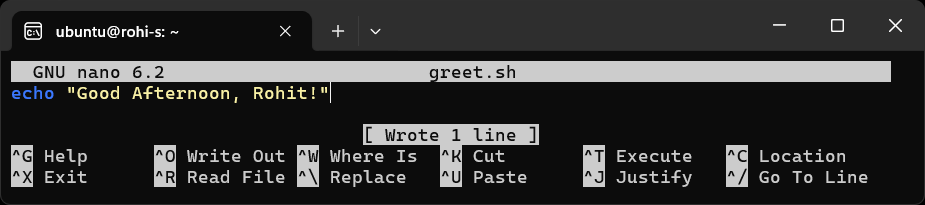
****

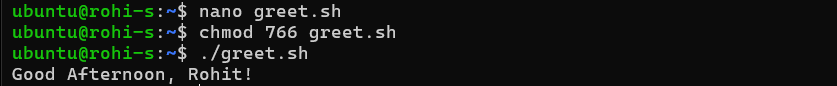
**c. Write a shell script to check if a given number is a prime number or not.**

****

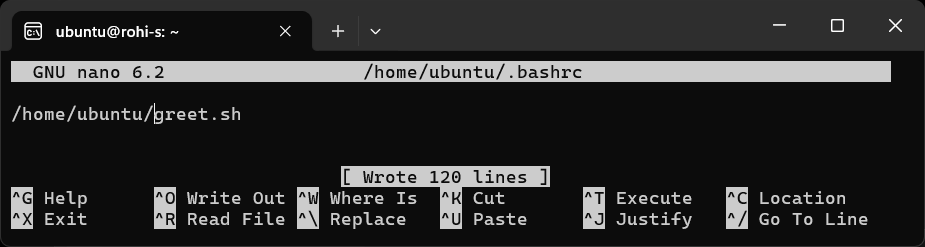
****

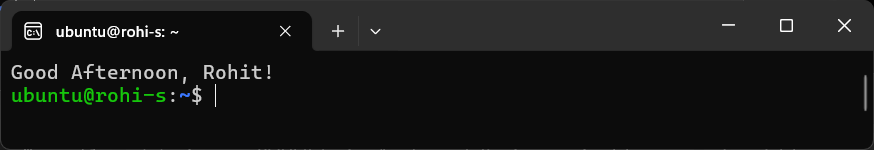
**d. Write a shell script to greet the user as per the time whenever he/ she opens terminal.**











**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXPERIMENT-4**

**Experiment 4:** Write a c program to implement the following scheduling algorithms.  
 a. First come first serve  
 b. Round Robin Scheduling  
 c. Shortest job first  
 d. Shortest Job remaining first.

**Code:**

#include <stdio.h>

#include <limits.h>

#define MAX\_PROCESSES 100

// Function to calculate average waiting time and average turnaround time

void calculateAvgTimes(int processes[], int n, int burst\_time[], int wait\_time[], int tat[]) {

    int total\_wt = 0, total\_tat = 0;

    printf("\nProcesses  Burst Time  Waiting Time  Turnaround Time\n");

    for (int i = 0; i < n; i++) {

        total\_wt += wait\_time[i];

        total\_tat += tat[i];

        printf(" %d\t\t%d\t\t%d\t\t%d\n", processes[i], burst\_time[i], wait\_time[i], tat[i]);

    }

    printf("\nAverage Waiting Time: %.2f", (float)total\_wt / (float)n);

    printf("\nAverage Turnaround Time: %.2f\n", (float)total\_tat / (float)n);

}

// First Come First Serve (FCFS)

void FCFS(int processes[], int n, int burst\_time[]) {

    int wait\_time[MAX\_PROCESSES], tat[MAX\_PROCESSES];

    wait\_time[0] = 0;

    for (int i = 1; i < n; i++) {

        wait\_time[i] = burst\_time[i - 1] + wait\_time[i - 1];

    }

    for (int i = 0; i < n; i++) {

        tat[i] = burst\_time[i] + wait\_time[i];

    }

    printf("\nFirst Come First Serve (FCFS) Scheduling:\n");

    calculateAvgTimes(processes, n, burst\_time, wait\_time, tat);

}

// Round Robin Scheduling

void RoundRobin(int processes[], int n, int burst\_time[], int quantum) {

    int wait\_time[MAX\_PROCESSES], tat[MAX\_PROCESSES];

    int remaining\_time[MAX\_PROCESSES];

    for (int i = 0; i < n; i++)

        remaining\_time[i] = burst\_time[i];

    int t = 0;

    while (1) {

        int done = 1;

        for (int i = 0; i < n; i++) {

            if (remaining\_time[i] > 0) {

                done = 0;

                if (remaining\_time[i] > quantum) {

                    t += quantum;

                    remaining\_time[i] -= quantum;

                } else {

                    t += remaining\_time[i];

                    wait\_time[i] = t - burst\_time[i];

                    remaining\_time[i] = 0;

                }

            }

        }

        if (done == 1)

            break;

    }

    for (int i = 0; i < n; i++)

        tat[i] = burst\_time[i] + wait\_time[i];

    printf("\nRound Robin Scheduling (Quantum = %d):\n", quantum);

    calculateAvgTimes(processes, n, burst\_time, wait\_time, tat);

}

// Shortest Job First (SJF) - Non-preemptive

void SJF(int processes[], int n, int burst\_time[]) {

    int wait\_time[MAX\_PROCESSES], tat[MAX\_PROCESSES];

    int completed[MAX\_PROCESSES] = {0};

    int min\_idx, min\_bt, completed\_count = 0, curr\_time = 0;

    while (completed\_count < n) {

        min\_bt = INT\_MAX;

        min\_idx = -1;

        for (int i = 0; i < n; i++) {

            if (!completed[i] && burst\_time[i] < min\_bt) {

                min\_bt = burst\_time[i];

                min\_idx = i;

            }

        }

        completed[min\_idx] = 1;

        curr\_time += burst\_time[min\_idx];

        wait\_time[min\_idx] = curr\_time - burst\_time[min\_idx];

        tat[min\_idx] = burst\_time[min\_idx] + wait\_time[min\_idx];

        completed\_count++;

    }

    printf("\nShortest Job First (SJF) Scheduling:\n");

    calculateAvgTimes(processes, n, burst\_time, wait\_time, tat);

}

// Shortest Remaining Time First (SRTF) - Preemptive

void SRTF(int processes[], int n, int burst\_time[]) {

    int wait\_time[MAX\_PROCESSES], tat[MAX\_PROCESSES], remaining\_time[MAX\_PROCESSES];

    for (int i = 0; i < n; i++)

        remaining\_time[i] = burst\_time[i];

    int complete = 0, t = 0, min\_rt = INT\_MAX;

    int shortest = 0, finish\_time;

    int check = 0;

    while (complete != n) {

        for (int j = 0; j < n; j++) {

            if ((remaining\_time[j] < min\_rt) && (remaining\_time[j] > 0)) {

                min\_rt = remaining\_time[j];

                shortest = j;

                check = 1;

            }

        }

        if (check == 0) {

            t++;

            continue;

        }

        remaining\_time[shortest]--;

        min\_rt = remaining\_time[shortest];

        if (min\_rt == 0)

            min\_rt = INT\_MAX;

        if (remaining\_time[shortest] == 0) {

            complete++;

            finish\_time = t + 1;

            wait\_time[shortest] = finish\_time - burst\_time[shortest];

            tat[shortest] = finish\_time;

        }

        t++;

    }

    printf("\nShortest Remaining Time First (SRTF) Scheduling:\n");

    calculateAvgTimes(processes, n, burst\_time, wait\_time, tat);

}

int main() {

    int n, processes[MAX\_PROCESSES], burst\_time[MAX\_PROCESSES], quantum;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    printf("Enter burst times for each process:\n");

    for (int i = 0; i < n; i++) {

        processes[i] = i + 1;

        printf("Process %d: ", i + 1);

        scanf("%d", &burst\_time[i]);

    }

    printf("\nFirst Come First Serve (FCFS) Scheduling:\n");

    FCFS(processes, n, burst\_time);

    printf("\nEnter time quantum for Round Robin: ");

    scanf("%d", &quantum);

    RoundRobin(processes, n, burst\_time, quantum);

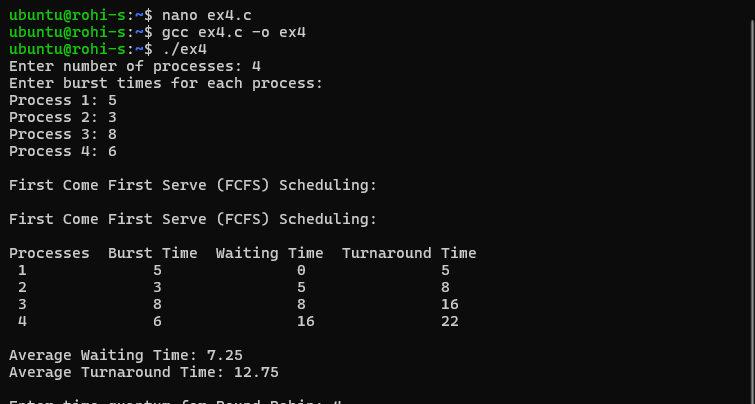
    SJF(processes, n, burst\_time);

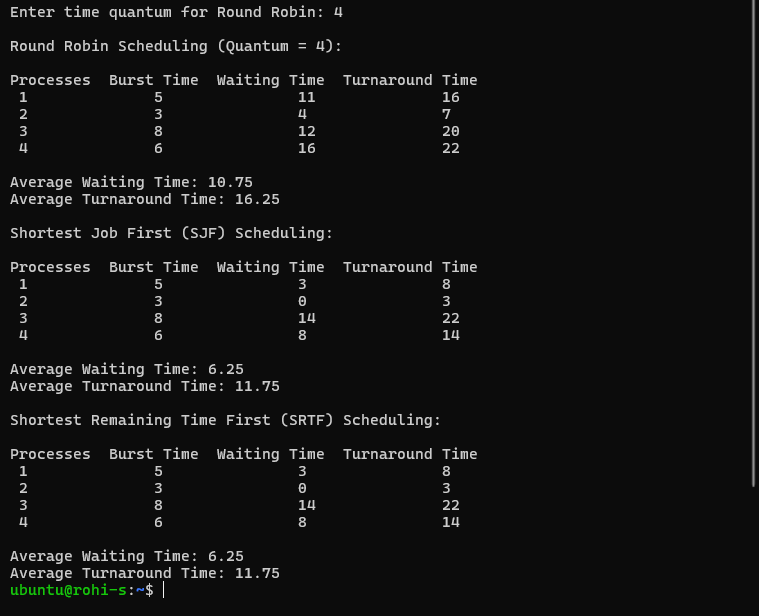
    SRTF(processes, n, burst\_time);

    return 0;

}

**Output:**

****

****

**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

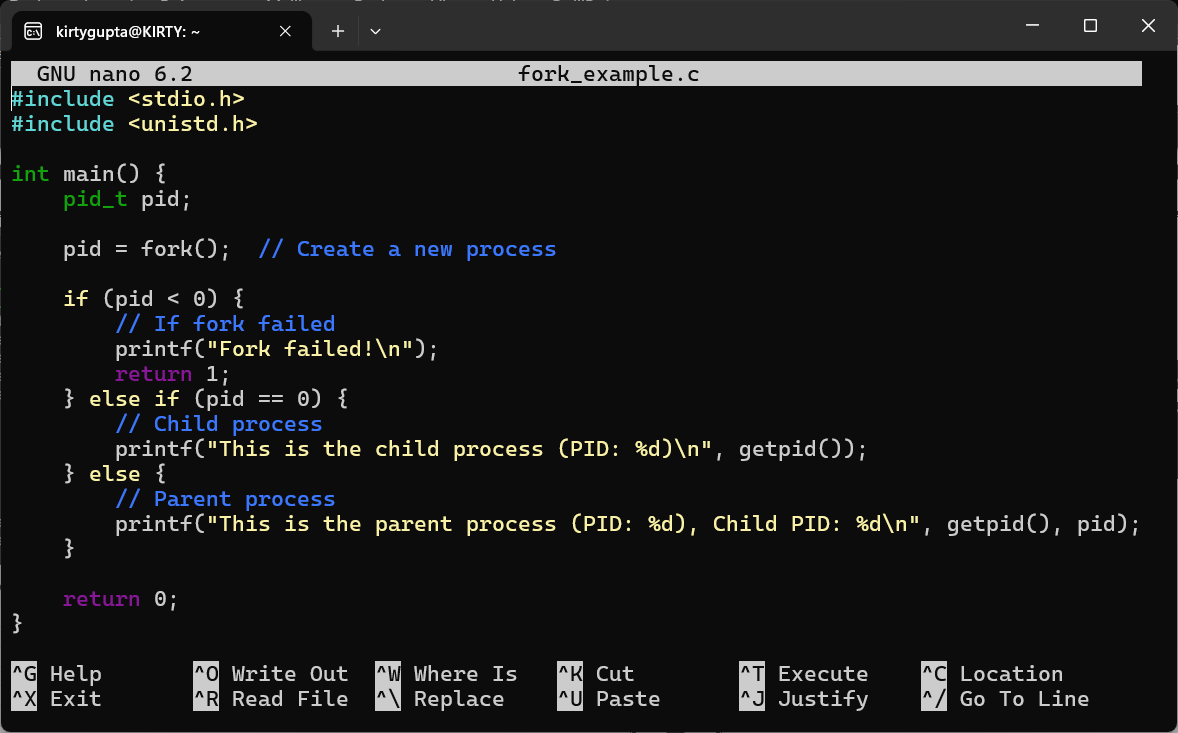
Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

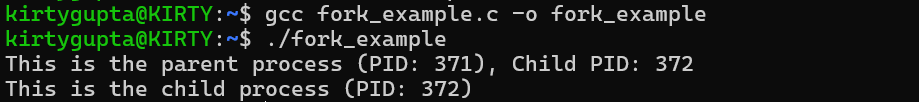
**EXPERIMENT-5**

**Experiment 5:** Process Management a) fork() b) execv() c) execlp() d) wait() and e) sleep()

**a. Program to implement the fork function using C.**

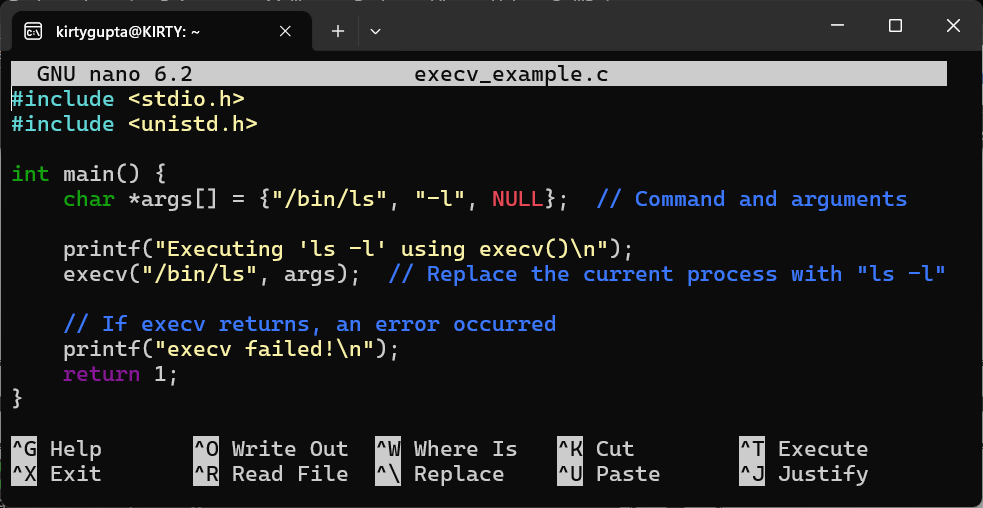
****

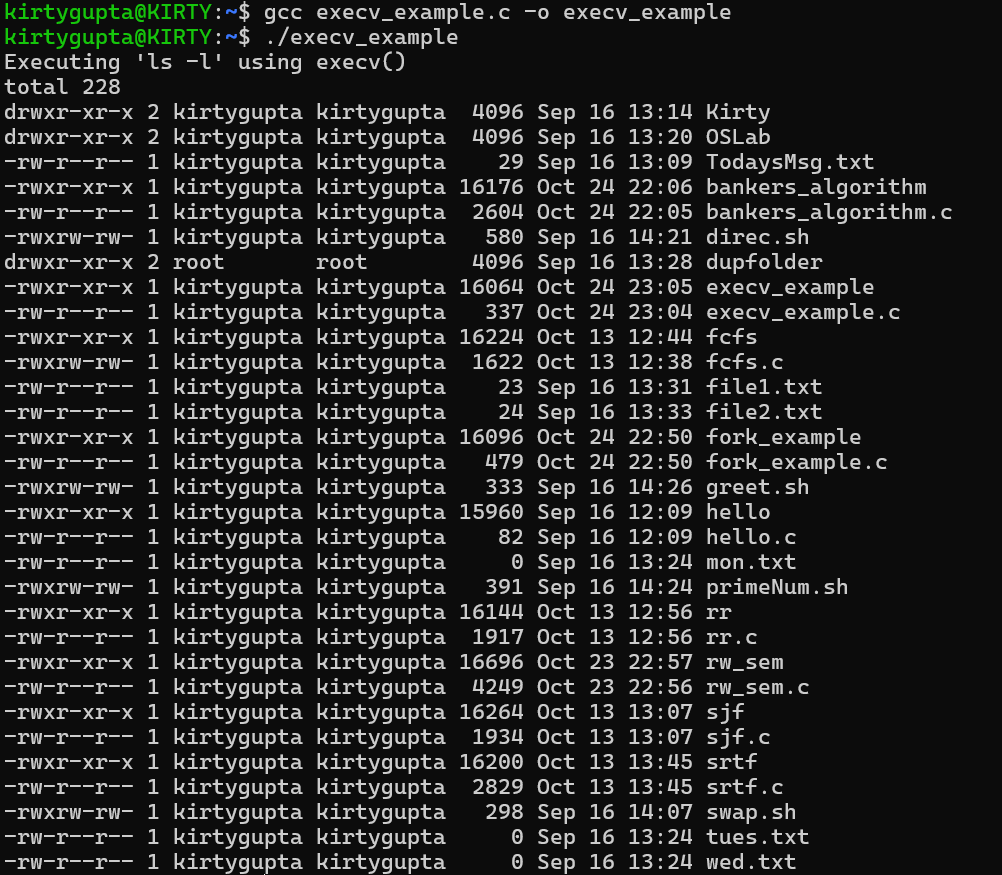
****

****

**b. Program to implement execv function using C.**

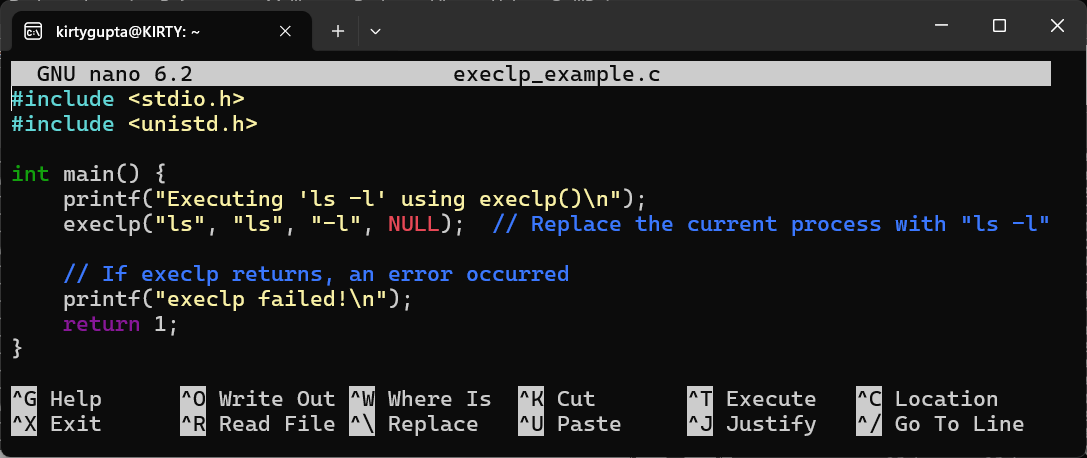
****

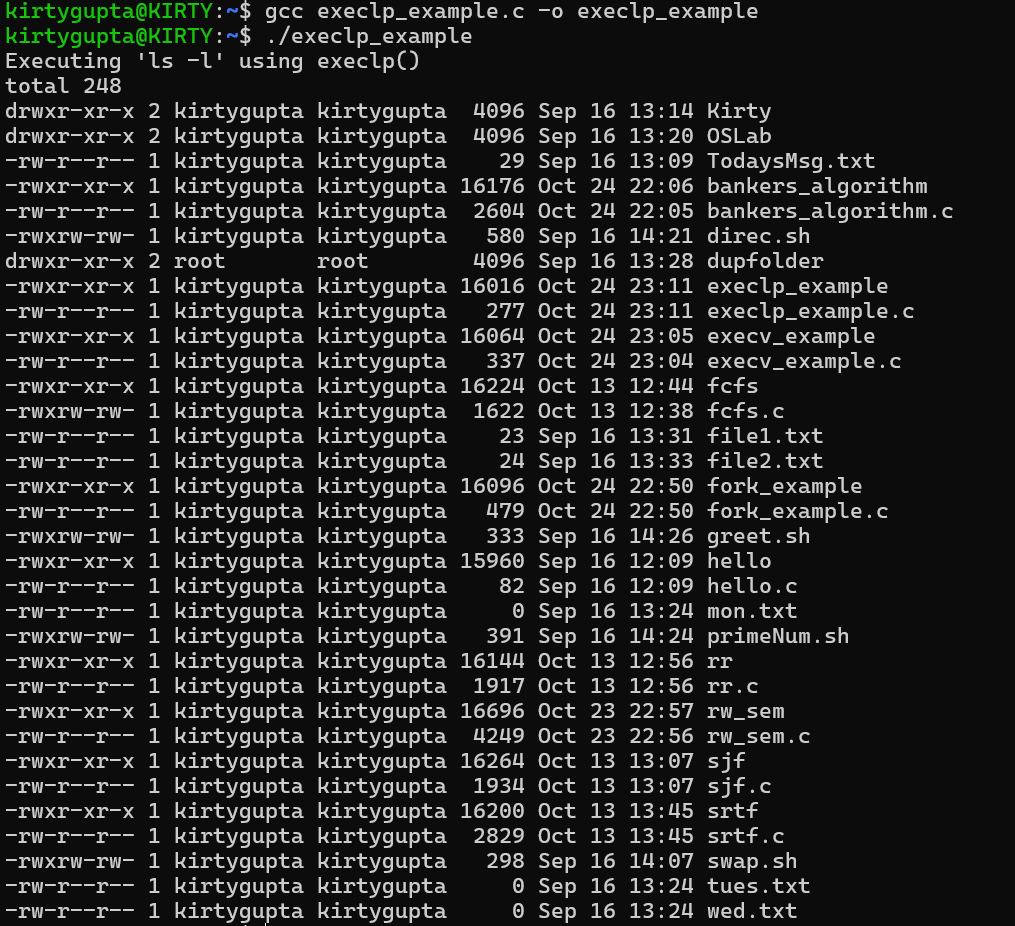
****

****

**c. Program to implement execlp function.**

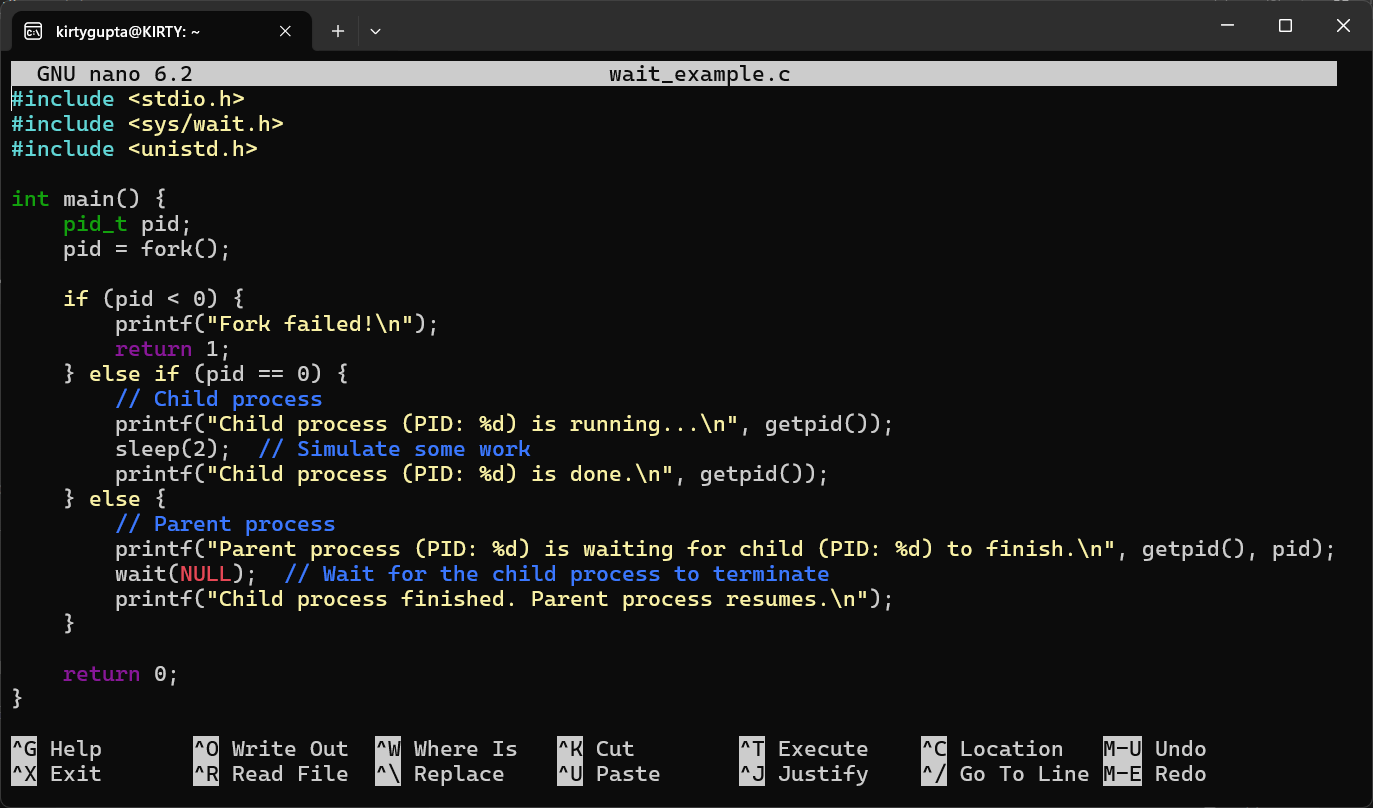
****

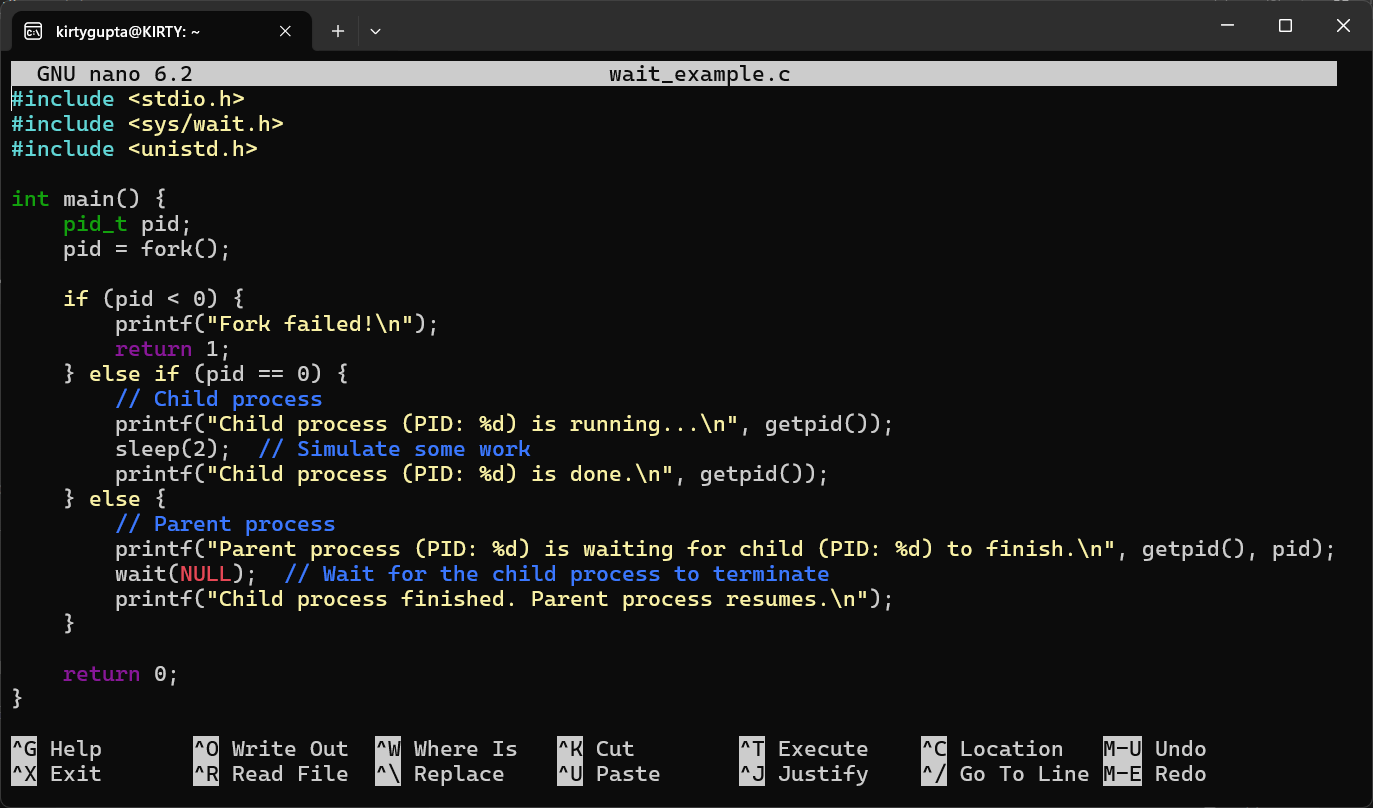
****

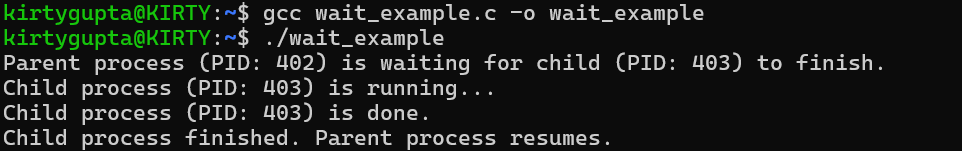
****

**d. Program to implement wait function using C.**

****

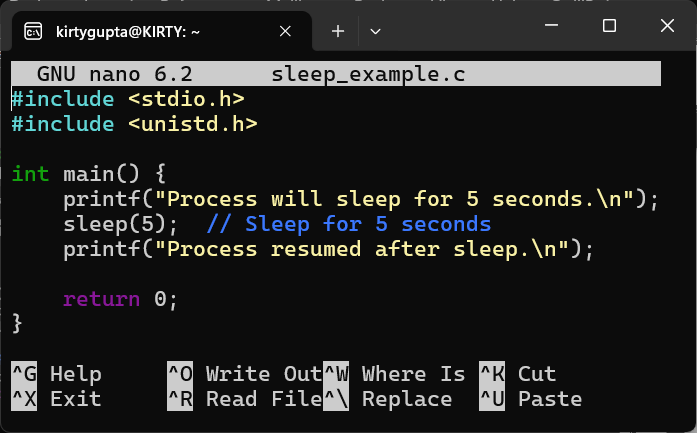
****

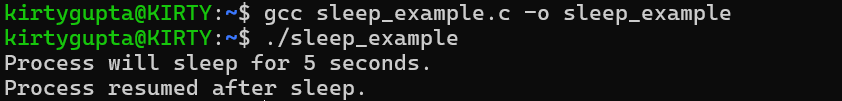
****

****

**e. Program to implement sleep function using C.**







**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXPERIMENT-6**

**Experiment 5:** Write a program to implement reader/writer problems using semaphore.

**Code:**

#include <stdio.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

sem\_t wrt;

pthread\_mutex\_t mutex;

int read\_count = 0;

int data = 0;

int read\_operations = 0;

int write\_operations = 0;

int MAX\_OPS = 5;

void\* writer(void\* arg) {

    int id = \*((int\*)arg);

    while (write\_operations < MAX\_OPS) {

        sem\_wait(&wrt);

        data += 1;

        printf("Writer %d is writing. Data = %d\n", id, data);

        write\_operations++;

        sem\_post(&wrt);

        sleep(1);

    }

    return NULL;

}

void\* reader(void\* arg) {

    int id = \*((int\*)arg);

    while (read\_operations < MAX\_OPS) {

        pthread\_mutex\_lock(&mutex);

        read\_count++;

        if (read\_count == 1) {

            sem\_wait(&wrt);

        }

        pthread\_mutex\_unlock(&mutex);

        printf("Reader %d is reading. Data = %d\n", id, data);

        read\_operations++;

        pthread\_mutex\_lock(&mutex);

        read\_count--;

        if (read\_count == 0) {

            sem\_post(&wrt);

        }

        pthread\_mutex\_unlock(&mutex);

        sleep(1);

    }

    return NULL;

}

int main() {

    pthread\_t rtid[5], wtid[2];

    int reader\_id[5] = {1, 2, 3, 4, 5};

    int writer\_id[2] = {1, 2};

    sem\_init(&wrt, 0, 1);

    pthread\_mutex\_init(&mutex, NULL);

    for (int i = 0; i < 2; i++) {

        pthread\_create(&wtid[i], NULL, writer, &writer\_id[i]);

    }

    for (int i = 0; i < 5; i++) {

        pthread\_create(&rtid[i], NULL, reader, &reader\_id[i]);

    }

    for (int i = 0; i < 2; i++) {

        pthread\_join(wtid[i], NULL);

    }

    for (int i = 0; i < 5; i++) {

        pthread\_join(rtid[i], NULL);

    }

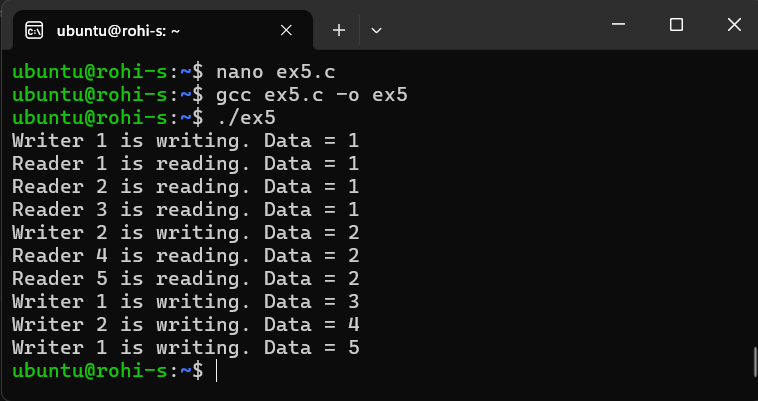
    sem\_destroy(&wrt);

    pthread\_mutex\_destroy(&mutex);

    return 0;

}

**Output:**

****

**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Date:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**EXPERIMENT-7**

**Experiment 6:** Write a program to implement Banker’s algorithm for deadlock avoidance.

**Code:**

#include <stdio.h>

#define MAX\_PROCESSES 5

#define MAX\_RESOURCES 3

int allocation[MAX\_PROCESSES][MAX\_RESOURCES], maximum[MAX\_PROCESSES][MAX\_RESOURCES], need[MAX\_PROCESSES][MAX\_RESOURCES], available[MAX\_RESOURCES];

int n, r; // n = number of processes, r = number of resources

void calculateNeed() {

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < r; j++) {

            need[i][j] = maximum[i][j] - allocation[i][j];

        }

    }

}

int isSafe() {

    int finish[MAX\_PROCESSES] = {0}, safeSequence[MAX\_PROCESSES], work[MAX\_RESOURCES];

    for (int i = 0; i < r; i++)

        work[i] = available[i];

    int count = 0;

    while (count < n) {

        int found = 0;

        for (int p = 0; p < n; p++) {

            if (finish[p] == 0) {

                int j;

                for (j = 0; j < r; j++) {

                    if (need[p][j] > work[j])

                        break;

                }

                if (j == r) {

                    for (int k = 0; k < r; k++)

                        work[k] += allocation[p][k];

                    safeSequence[count++] = p;

                    finish[p] = 1;

                    found = 1;

                }

            }

        }

        if (found == 0) {

            printf("The system is not in a safe state.\n");

            return 0;

        }

    }

    printf("The system is in a safe state.\nSafe sequence is: ");

    for (int i = 0; i < n; i++)

        printf("%d ", safeSequence[i]);

    printf("\n");

    return 1;

}

int main() {

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

    scanf("%d", &n);

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

    scanf("%d", &r);

    printf("Enter the Allocation matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < r; j++) {

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

        }

    }

    printf("Enter the Maximum matrix:\n");

    for (int i = 0; i < n; i++) {

        for (int j = 0; j < r; j++) {

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

        }

    }

    printf("Enter the Available resources:\n");

    for (int i = 0; i < r; i++) {

        scanf("%d", &available[i]);

    }

    calculateNeed();

    if (isSafe()) {

        int pno;

        printf("Enter the process number (0 to %d) that is requesting resources: ", n - 1);

        scanf("%d", &pno);

        int req[MAX\_RESOURCES];

        printf("Enter the requested resources for process %d:\n", pno);

        for (int i = 0; i < r; i++) {

            scanf("%d", &req[i]);

        }

        int canGrant = 1;

        for (int i = 0; i < r; i++) {

            if (req[i] > need[pno][i]) {

                printf("Error: Process has exceeded its maximum claim.\n");

                canGrant = 0;

                break;

            }

            if (req[i] > available[i]) {

                printf("Error: Not enough resources available.\n");

                canGrant = 0;

                break;

            }

        }

        if (canGrant) {

            for (int i = 0; i < r; i++) {

                available[i] -= req[i];

                allocation[pno][i] += req[i];

                need[pno][i] -= req[i];

            }

            if (isSafe()) {

                printf("Request granted.\n");

            } else {

                printf("Request cannot be granted as it leads to an unsafe state.\n");

                for (int i = 0; i < r; i++) {

                    available[i] += req[i];

                    allocation[pno][i] -= req[i];

                    need[pno][i] += req[i];

                }

            }

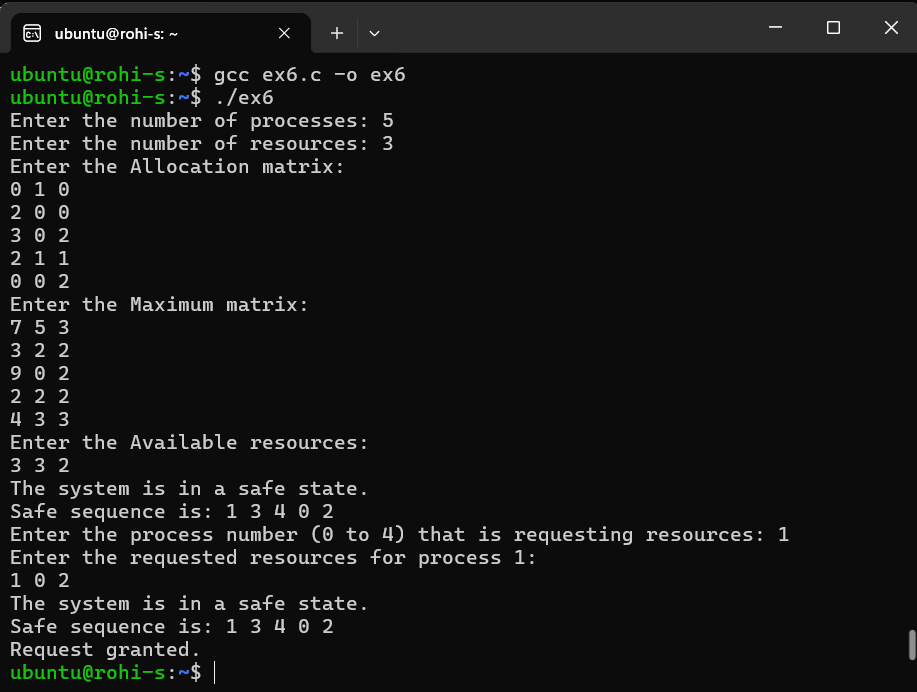
        }

    }

    return 0;

}

**Output:**

****

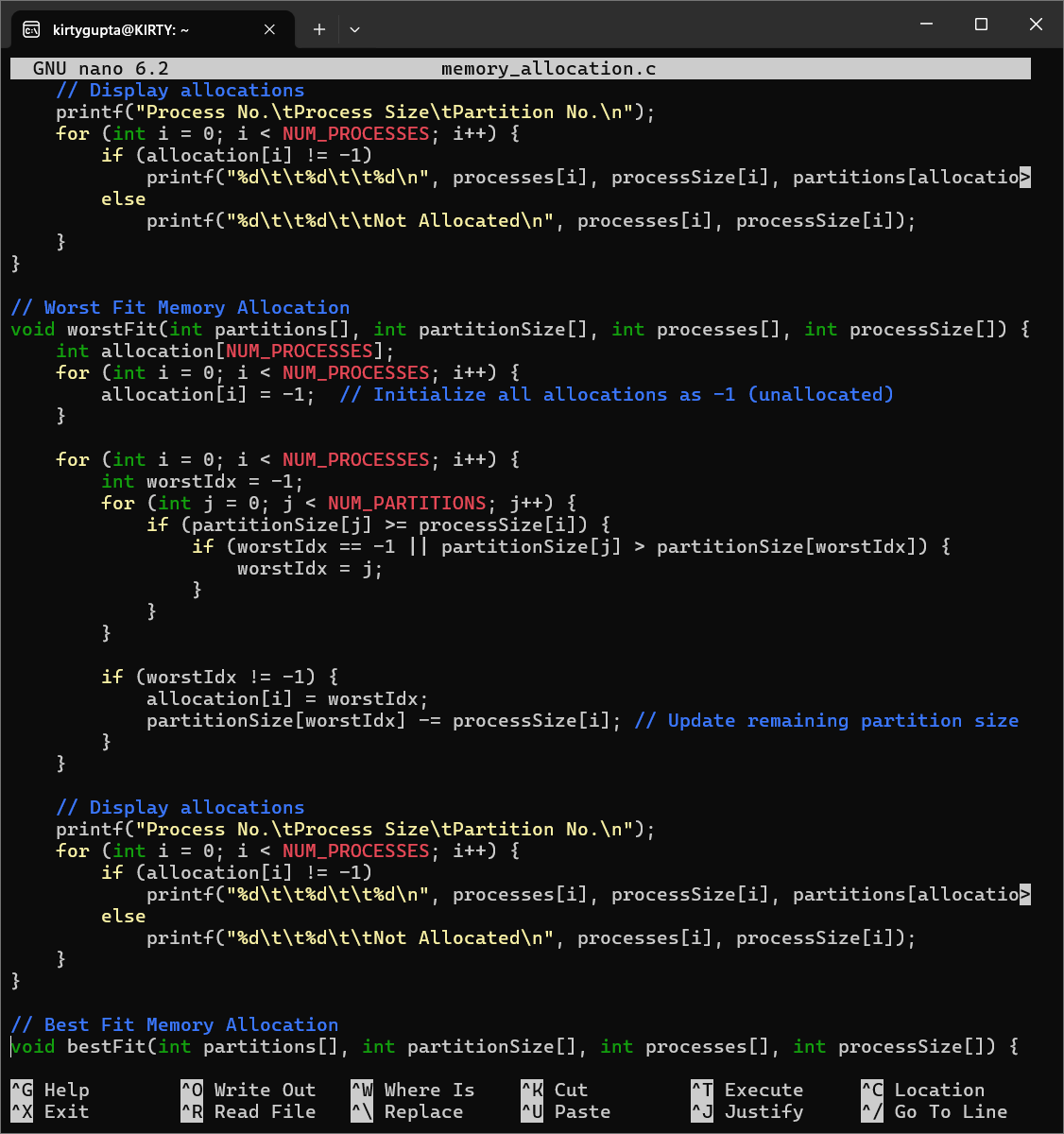
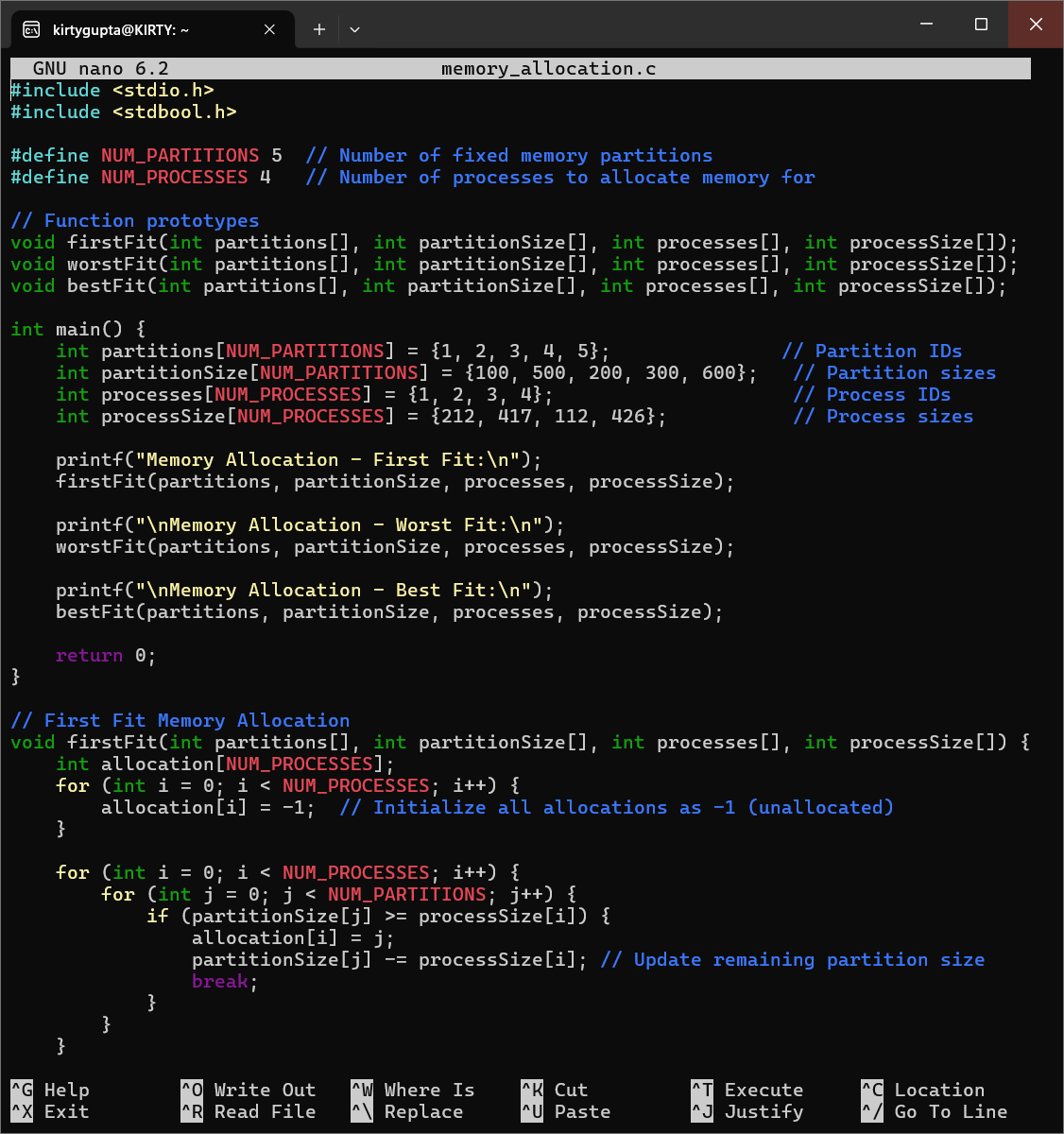
**Learning Outcomes:**

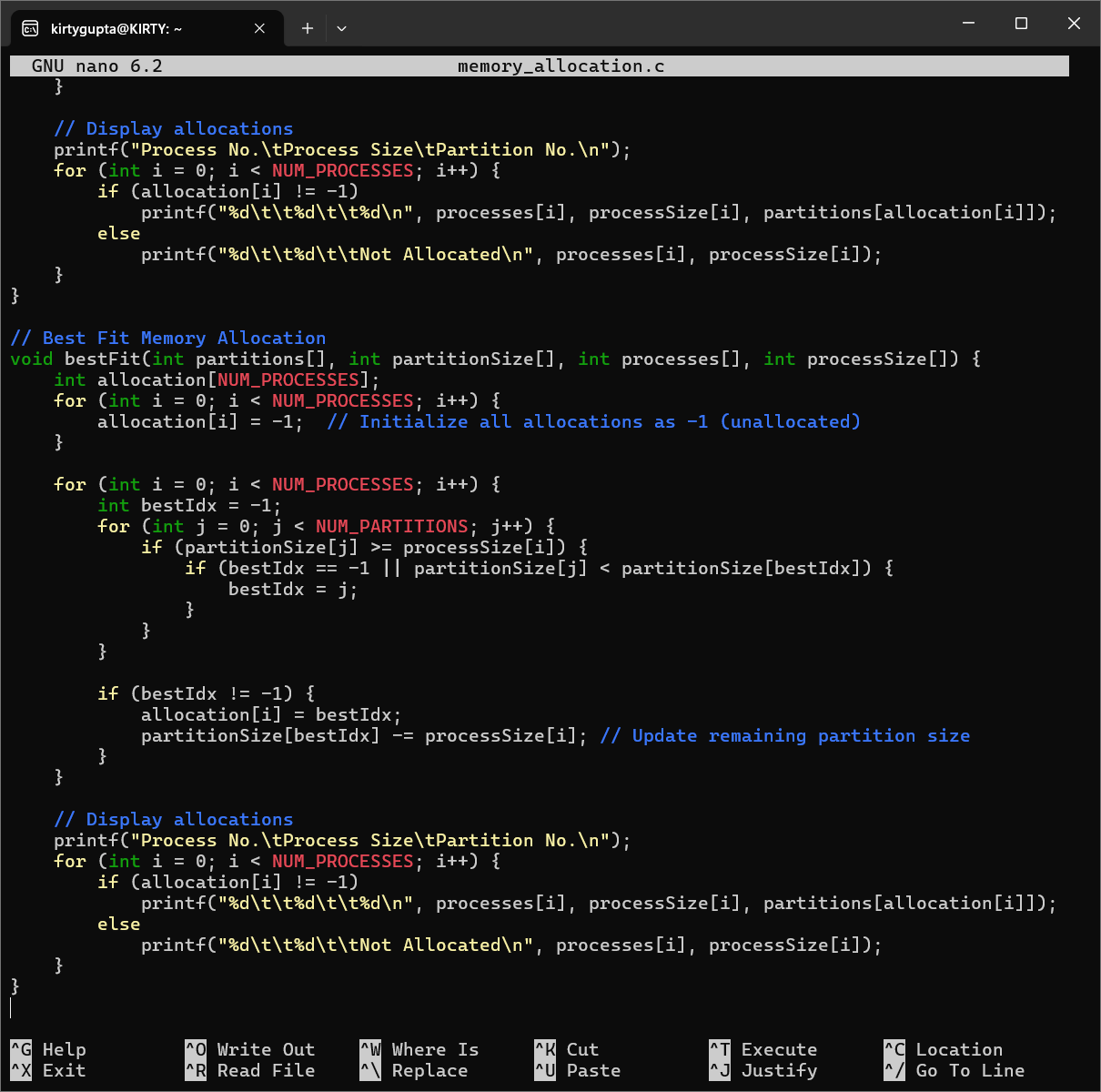
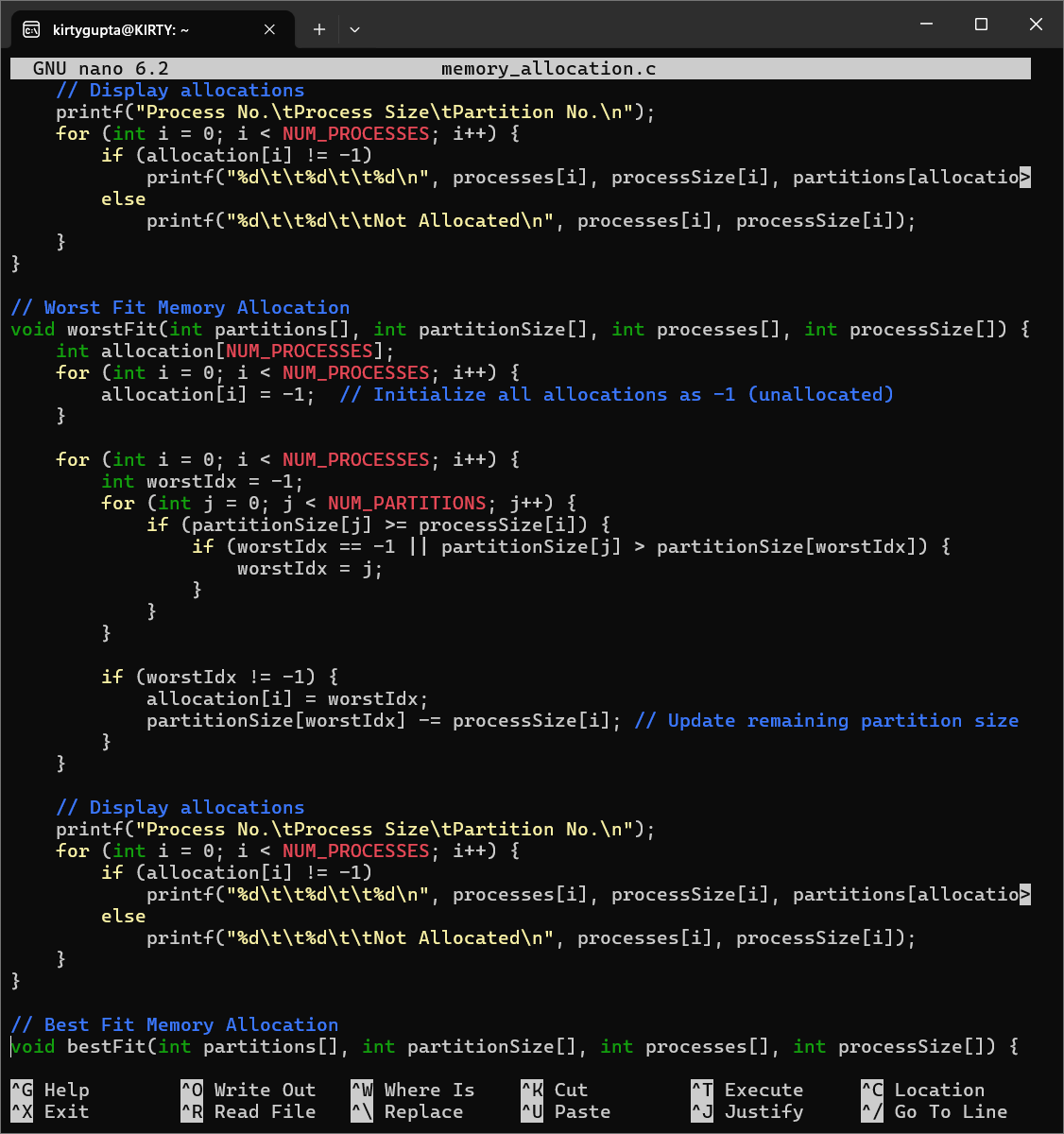
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**EXPERIMENT-8**

**Experiment 8:** Implementation of the following Memory Allocation Methods for fixed partition : a.) First Fit b.) Worst Fit c.) Best Fit.

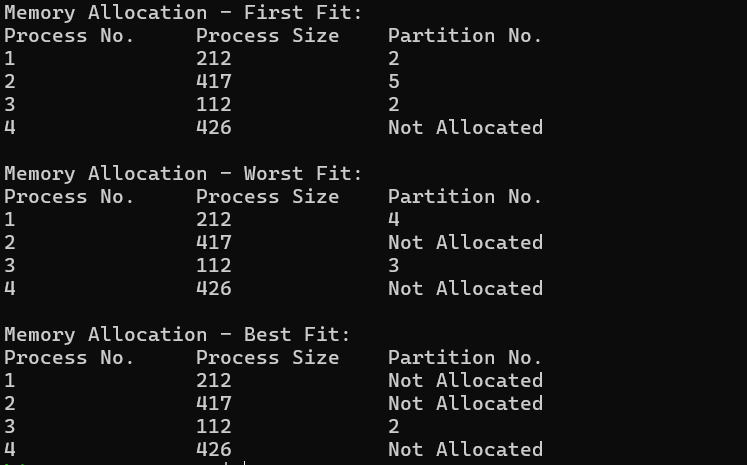


****

****

****

****

****

**Learning Outcomes:**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**