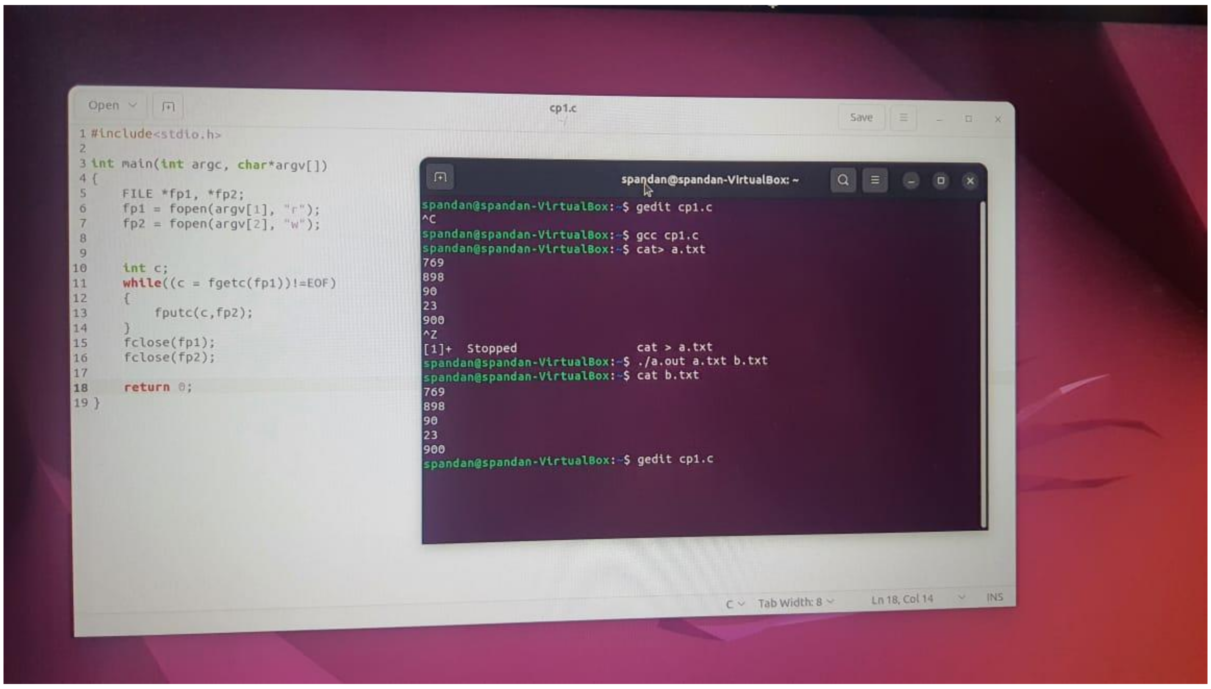
NAME: SPANDAN MUKHERJEE

REGISTRATION NUMBER: 21BCE1132

SUBJECT: OPERATING SYSTEMS LAB - 2

Questions

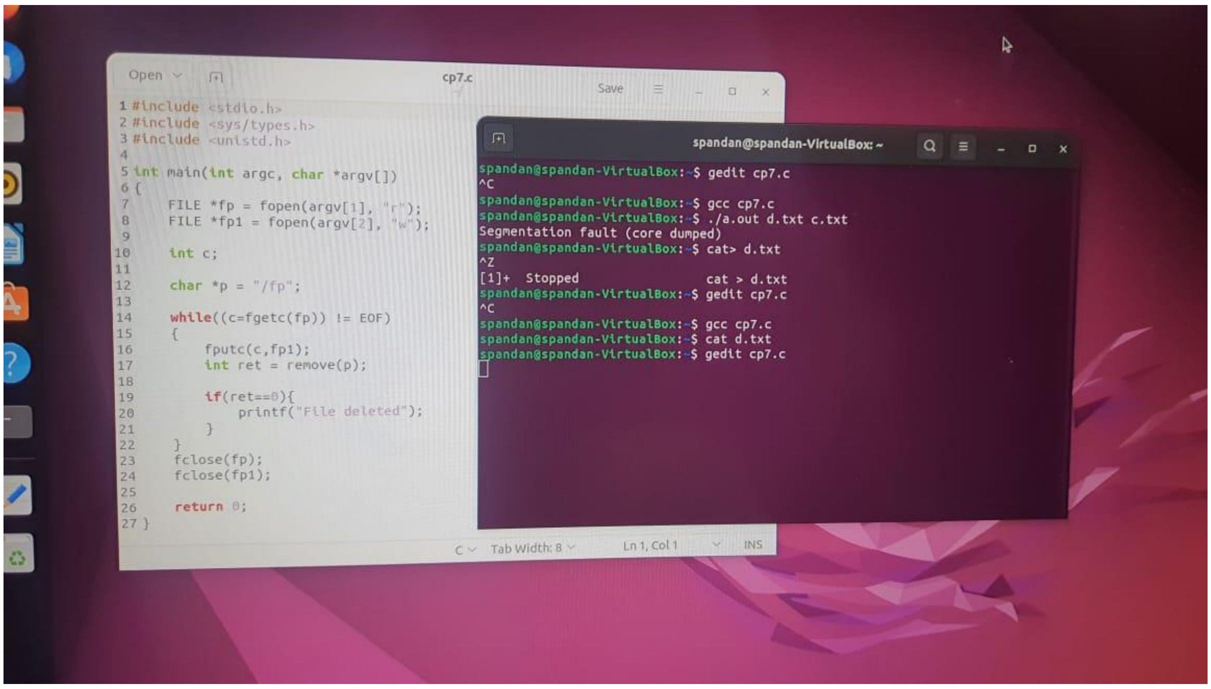
1) Simulate CP linux command using C



```c
#include<stdio.h>

int main(int argc, char*argv[])
{
    FILE *fp1, *fp2;
    fp1 = fopen(argv[1], "r");
    fp2 = fopen(argv[2], "w");


    int c;
    while((c = fgetc(fp1))!=EOF)
    {
        fputc(c,fp2);
    }
    fclose(fp1);
    fclose(fp2);

    return 0;
}
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp1.c
^C
spandan@spandan-VirtualBox:~$ gcc cp1.c
spandan@spandan-VirtualBox:~$ cat> a.txt
769
898
90
23
900
^Z
[1]+  Stopped                 cat > a.txt
spandan@spandan-VirtualBox:~$ ./a.out a.txt b.txt
spandan@spandan-VirtualBox:~$ cat b.txt
769
898
90
23
900
spandan@spandan-VirtualBox:~$ gedit cp1.c
```

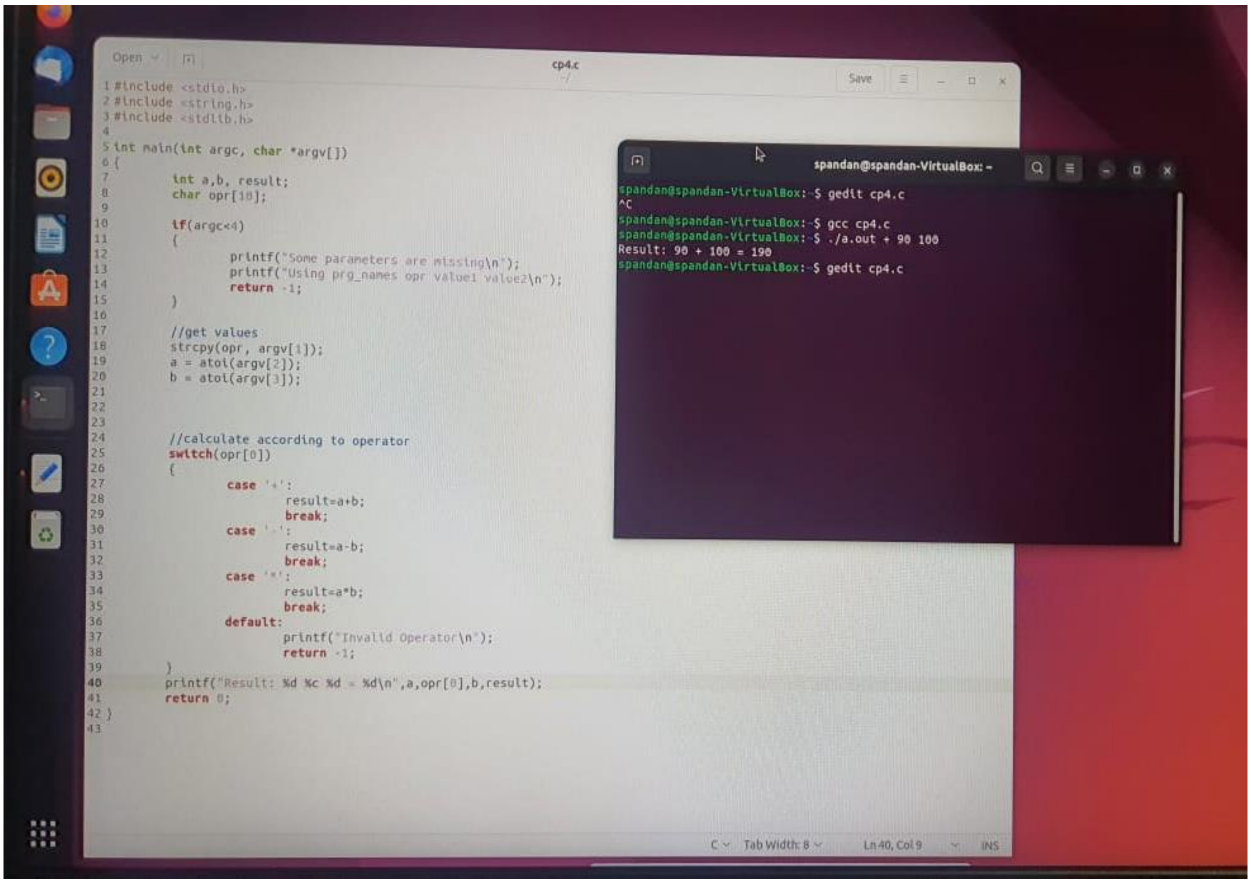
2) Simulate MV linux command using C



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

int main(int argc, char *argv[])
{
    FILE *fp = fopen(argv[1], "r");
    FILE *fp1 = fopen(argv[2], "w");

    int c;

    char *p = "/fp";

    while((c=fgetc(fp)) != EOF)
    {
        fputc(c,fp1);
        int ret = remove(p);

        if(ret==0){
            printf("File deleted");
        }
    }
    fclose(fp);
    fclose(fp1);

    return 0;
}
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp7.c
^C
spandan@spandan-VirtualBox:~$ gcc cp7.c
spandan@spandan-VirtualBox:~$ ./a.out d.txt c.txt
Segmentation fault (core dumped)
spandan@spandan-VirtualBox:~$ cat> d.txt
^Z
[1]+  Stopped                 cat > d.txt
spandan@spandan-VirtualBox:~$ gedit cp7.c
^C
spandan@spandan-VirtualBox:~$ gcc cp7.c
spandan@spandan-VirtualBox:~$ cat d.txt
spandan@spandan-VirtualBox:~$ gedit cp7.c
```

3) Perform arithmetic operations using command line arguments

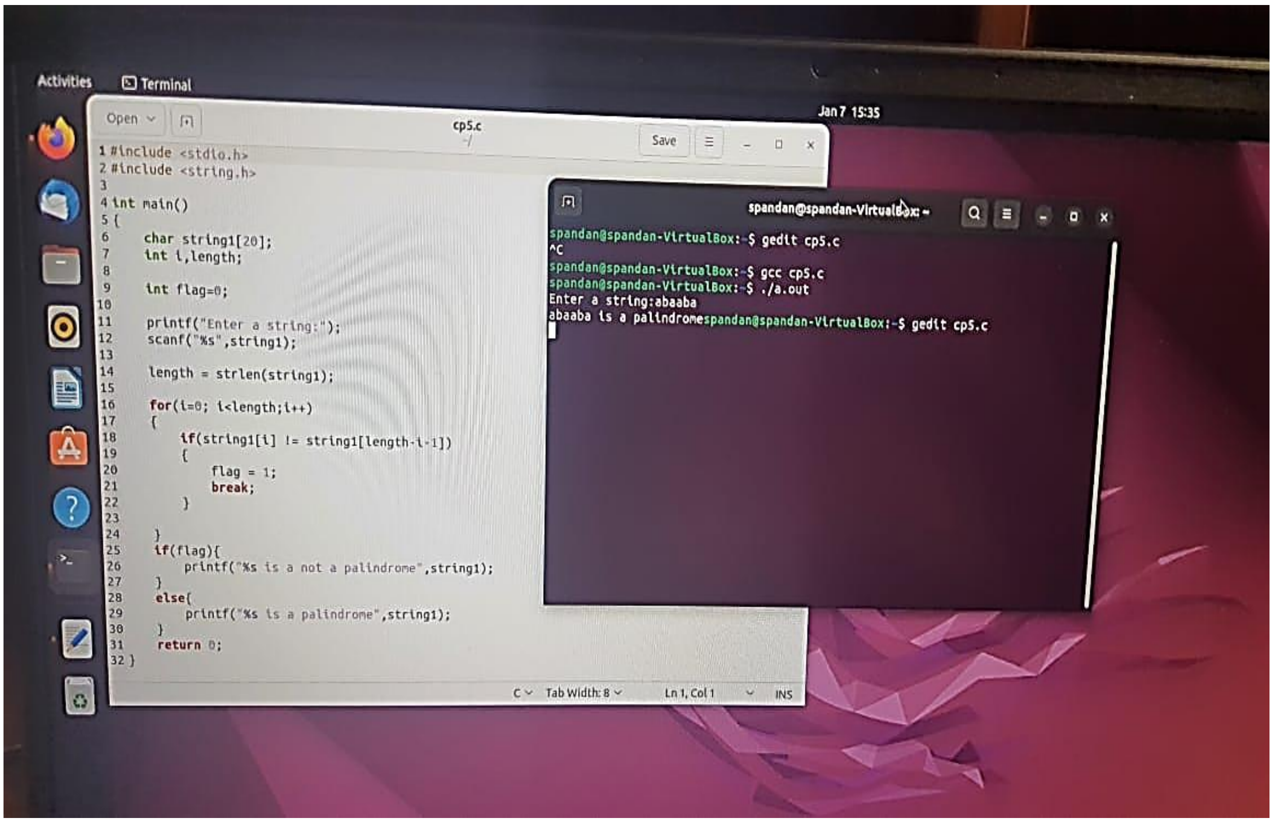


```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
        int a,b, result;
        char opr[10];

        if(argc<4)
        {
                printf("Some parameters are missing\n");
                printf("Using prg_names opr value1 value2\n");
                return -1;
        }

        //get values
        strcpy(opr, argv[1]);
        a = atoi(argv[2]);
        b = atoi(argv[3]);



        //calculate according to operator
        switch(opr[0])
        {
                case '+':
                        result=a+b;
                        break;
                case '-':
                        result=a-b;
                        break;
                case '*':
                        result=a*b;
                        break;
                default:
                        printf("Invalid Operator\n");
                        return -1;
        }
        printf("Result: %d %c %d = %d\n",a,opr[0],b,result);
        return 0;
}

```

spandan@spandan-VirtualBox: ~

```
spandan@spandan-VirtualBox:~$ gedit cp4.c
^C
spandan@spandan-VirtualBox:~$ gcc cp4.c
spandan@spandan-VirtualBox:~$ ./a.out + 90 100
Result: 90 + 100 = 190
spandan@spandan-VirtualBox:~$ gedit cp4.c
```

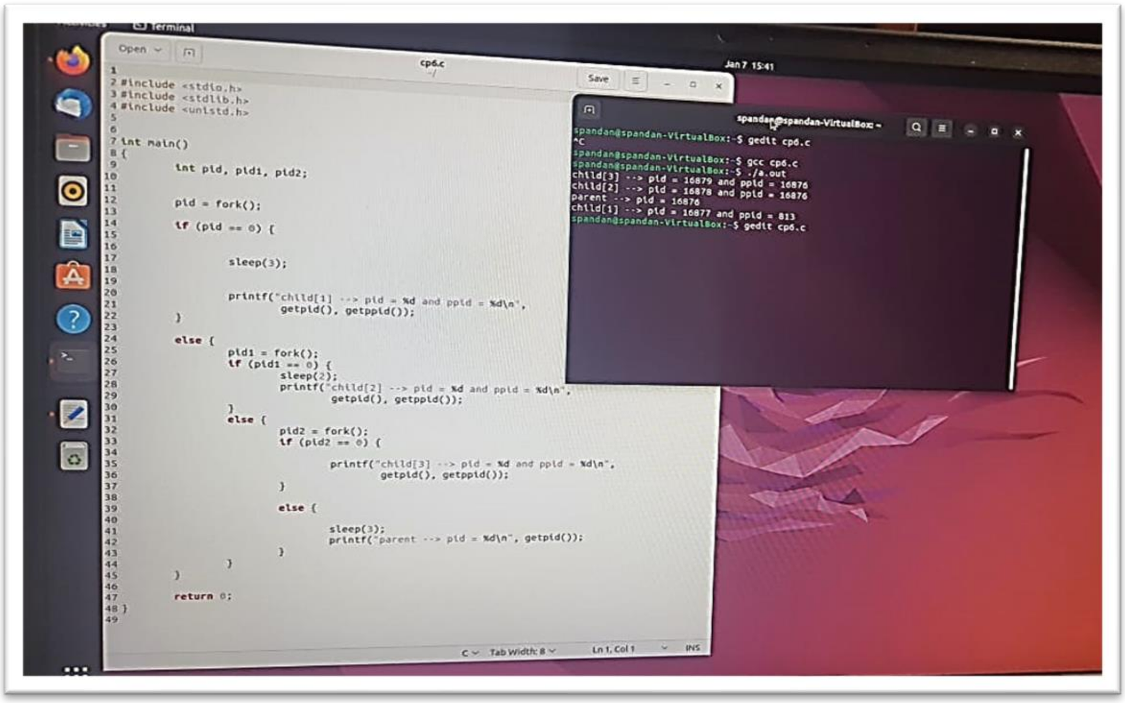
4) Check whether the given string is palindrome or not and ensure to take the input while executing the program



```c
#include <stdio.h>
#include <string.h>

int main()
{
    char string1[20];
    int i,length;

    int flag=0;

    printf("Enter a string:");
    scanf("%s",string1);

    length = strlen(string1);

    for(i=0; i<length;i++)
    {
        if(string1[i] != string1[length-i-1])
        {
            flag = 1;
            break;
        }

    }
    if(flag){
        printf("%s is a not a palindrome",string1);
    }
    else{
        printf("%s is a palindrome",string1);
    }
    return 0;
}
```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp5.c
^C
spandan@spandan-VirtualBox:~$ gcc cp5.c
spandan@spandan-VirtualBox:~$ ./a.out
Enter a string:abaaba
abaaba is a palindromespandan@spandan-VirtualBox:~$ gedit cp5.c
```

5) Create 3 child processes from the same parent process and show the child processes are created from the same parent process.



```c

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


int main()
{
        int pid, pid1, pid2;


        pid = fork();

        if (pid == 0) {


                sleep(3);


                printf("child[1] --> pid = %d and ppid = %d\n",
                        getpid(), getppid());
        }

        else {
                pid1 = fork();
                if (pid1 == 0) {
                        sleep(2);
                        printf("child[2] --> pid = %d and ppid = %d\n",
                                getpid(), getppid());
                }
                else {
                        pid2 = fork();
                        if (pid2 == 0) {

                                printf("child[3] --> pid = %d and ppid = %d\n",
                                        getpid(), getppid());
                        }

                        else {

                                sleep(3);
                                printf("parent --> pid = %d\n", getpid());
                        }

                }
        }

        return 0;
}

```

```
spandan@spandan-VirtualBox: ~
spandan@spandan-VirtualBox:~$ gedit cp6.c
^C
spandan@spandan-VirtualBox:~$ gcc cp6.c
spandan@spandan-VirtualBox:~$ ./a.out
child[3] --> pid = 16879 and ppid = 16876
child[2] --> pid = 16878 and ppid = 16876
parent --> pid = 16876
child[1] --> pid = 16877 and ppid = 813
spandan@spandan-VirtualBox:~$ gedit cp6.c
```

6) Discuss the use of Command line arguments in C

In C programming, command line arguments are an important concept. Using command line parameters, we can perform any task. It is mostly used when you need to control your program from outside.

Before we go any further, let us define certain terms that will be used in this article, such as Command-Line and Command-Line Arguments.

The command line is a text interface for your computer that allows you to enter commands for immediate execution. A command-line can perform almost everything that a graphical user interface can. Many tasks can be performed more rapidly and are easier to automate.

For example:

- You can navigate around your computer's files and directories using the command line.
- The command line can be scripted to automate complex tasks, like the example given below:

If a user wants to put 50+ files' data into a file, this is a highly time-consuming task. Copying data from 50+ files, on the other hand, can be done in less than a minute with a single command at the command line. And many more.