Assignment – 1
TOPIC: BASIC UNIX COMMANDS & FILE SYSTEM COMMANDS

1. Display the date using the "**date**" command.

Ans.
```
abhignya@hplaptop:~$ date
Wed Jul 24 12:30:19 IST 2024
```

2. Check who are the users logged in using the "**who**" command.

Ans.
```
abhignya@hplaptop:~$ who
abhignya@hplaptop:~$ whoami
abhignya
```

3. Check the running processes using the "**ps**" command.

Ans.
```
abhignya@hplaptop:~$ ps
  PID TTY          TIME CMD
   52 pts/0    00:00:00 bash
   89 pts/0    00:00:00 ps
```

4. List the files with "**ls**" command with and without **-l** option.

Ans.
```
abhignya@hplaptop:~$ ls
MCA2022  Unix_File_System
```

```
abhignya@hplaptop:~$ ls -l
total 8
drwxr-xr-x 3 abhignya abhignya 4096 Jul 24 12:59 MCA2022
drwxr-xr-x 2 abhignya abhignya 4096 Jul 24 13:03 Unix_File_System
```

5. Check the *manual of* **ls** command.

Ans.
```
abhignya@hplaptop:~$ 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
```

6. Show the commands used to display (i) filenames (ii) processes (iii) users.

Ans.i)

```
abhignya@hplaptop:~$ ls
MCA2022  Unix_File_System
```

ii)

```
abhignya@hplaptop:~$ ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   1912  1192 ?        Sl   13:07   0:00 /init
root          57  0.0  0.0   2172   372 ?        Ss   13:17   0:00 /init
root          58  0.0  0.0   2180   372 ?        R    13:17   0:00 /init
abhignya      59  0.1  0.1   6076  5124 pts/0    Ss   13:17   0:00 -bash
abhignya      74  0.0  0.0   7476  3092 pts/0    R+   13:17   0:00 ps aux
```

iii)

```
abhignya@hplaptop:~$ whoami
abhignya
abhignya@hplaptop:~$ w
 13:17:39 up 48 min,  0 users,  load average: 0.04, 0.01, 0.00
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
abhignya@hplaptop:~$
```

7. Check and state the difference between man and whatis command by checking **man cp** & **whatis cp**.

Ans. man cp

```
CP(1)                                   User Commands                                   CP(1)

NAME
       cp - copy files and directories

SYNOPSIS
       cp [OPTION]... [-T] SOURCE DEST
       cp [OPTION]... SOURCE... DIRECTORY
       cp [OPTION]... -t DIRECTORY SOURCE...

DESCRIPTION
       Copy SOURCE to DEST, or multiple SOURCE(s) to DIRECTORY.

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

       -a, --archive
              same as -dR --preserve=all

       --attributes-only
              don't copy the file data, just the attributes

       --backup[=CONTROL]
              make a backup of each existing destination file

       -b     like --backup but does not accept an argument

       --copy-contents
              copy contents of special files when recursive

       -d     same as --no-dereference --preserve=links

       -f, --force
              if  an existing destination file cannot be opened, remove it and try again (this option is ignored when
              the -n option is also used)

       -i, --interactive
              prompt before overwrite (overrides a previous -n option)

       -H     follow command-line symbolic links in SOURCE
```

Whatis cp

```
abhignya@hplaptop:~$ whatis cp
cp (1)               - copy files and directories
```

Summary of Differences:

- **Detail Level**:
  - man: Provides a detailed manual page with extensive information about the command, its options, and usage.

  - whatis: Provides a brief, one-line summary of what the command does.
- **Usage Context**:
  - man: Useful when you need in-depth information and guidance on how to use a command, including all available options.
  - whatis: Useful when you need a quick summary to understand what a command does.

8. What is the primary difference between **printf** and **echo** command? Check and print.

Ans. **echo Command**:

- echo is simpler and primarily used to display a line of text.
- It automatically adds a newline character at the end of the output.
- It has limited formatting capabilities.

```
abhignya@hplaptop:~$ echo "Hi, I am Abhignya"
Hi, I am Abhignya
```

printf Command:

- printf is more powerful and is used for formatted output.
- It does not automatically add a newline character; you need to specify it.
- It allows for complex formatting, similar to the printf function in C.

```
abhignya@hplaptop:~$ printf "Hello World\!\n"
Hello World\!
```

Summary of Differences:

- **Newline Handling**:
  - echo adds a newline by default.
  - printf does not add a newline unless specified.
- **Formatting Capabilities**:
  - echo has limited formatting capabilities.
  - printf supports complex formatting similar to the printf function in C.
- **Use Cases**:
  - echo is suitable for simple text output.
  - printf is preferred when precise formatting is required.

9. In the home directory, create a directory *MCA2022*. Inside the *MCA2022,* create another directory *<FistName_Section_ClassRoll>* and get into the directory [**~ /MCA2022/Ankur_A_00$**].

Ans.

```
abhignya@hplaptop:~$ mkdir ~/MCA2022
abhignya@hplaptop:~$ mkdir ~/MCA2022/Abhignya_B_16
abhignya@hplaptop:~$ cd  ~/MCA2022/Abhignya_B_16
abhignya@hplaptop:~/MCA2022/Abhignya_B_16$
```

10. Go to the subdirectory and create another subdirectory "**Unix_File_System**" within it.

Ans.

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16$ mkdir Unix_File_System
abhignya@hplaptop:~/MCA2022/Abhignya_B_16$ ls
Unix_File_System
```

11. Create the subdirectories TestA, TestB, TestC and corresponding sub-subdirectories TestA-1, TestA-2, TestB-1, TestB-2, TestB-3, TestC-1, TestB-2-i in a single command.

Ans.

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ mkdir -p ~/MCA2022/Abhignya_B_16/Unix_File_System/{TestA/{TestA-1
,TestA-2},TestB/{TestB-1,TestB-2,TestB-3,Testb-2-i},TestC/TestC-1}
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ ls
TestA  TestB  TestC
```

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ cd TestA
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA$ ls
TestA-1  TestA-2
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA$ cd ..
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ cd TestB
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestB$ ls
TestB-1  TestB-2  TestB-3  Testb-2-i
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestB$ cd ..
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ cd TestC
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestC$ ls
TestC-1
```

12. Show the absolute path of TestB-2-i.

Ans.

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ realpath Testb-2-i
/home/abhignya/MCA2022/Abhignya_B_16/Unix_File_System/Testb-2-i
```

Assignment – 2
TOPIC: FILE SYSTEM COMMANDS

1. Create two C files to print "**Hello World!**" in two different ways:
   a. Program containing normal statement terminator→HelloWorld1.c.
   b. Program without any statement terminator → HelloWorld2.c.

   Ans. a)

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ vim HelloWorld1.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ gcc HelloWorld1.c -o HelloWorld1
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ./HelloWorld1
Hello World!
```

   b)

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ vim HelloWorld2.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ gcc HelloWorld2.c -o HelloWorld2
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ./HelloWorld2
Hello World!
```

2. Display the contents of the files.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cat HelloWorld1.c
#include<stdio.h>
int main(){
        printf("Hello World!\n");
        return 0;
}
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cat HelloWorld2.c
#include<stdio.h>
int main(){
        printf("Hello World!\n");
        return 0;
}
```

3. Concatenate the two files to a third file.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cat HelloWorld1.c HelloWorld2.c > Program.txt
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ls
HelloWorld1  HelloWorld1.c  HelloWorld2  HelloWorld2.c  Program.txt
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cat Program.txt
#include<stdio.h>
int main(){
        printf("Hello World!\n");
        return 0;
}
#include<stdio.h>
int main(){
        printf("Hello World!\n");
        return 0;
}
```

4. Show the above file types.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ file HelloWorld1.c
HelloWorld1.c: C source, ASCII text
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ file HelloWorld2.c
HelloWorld2.c: C source, ASCII text
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ file Program.txt
Program.txt: C source, ASCII text
```

5. Copy all the files to the home directory in an interactive manner.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cp -i HelloWorld1.c HelloWorld2.c Program.txt ~
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cd ~
abhignya@hplaptop:~$ ls
2023  Abhignya_B_16  Assignment1  HelloWorld1.c  HelloWorld2.c  MCA2022  MCA2023  Program.txt  Unix_File_System
```

6. Create a copy of the C file in TestA-1.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cp HelloWorld1.c ~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1/
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ls ~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1/
HelloWorld1.c
```

7. Copy the file to the home directory in an interactive manner.

   Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cd ~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1/
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ ls
HelloWorld1.c
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ cp -i HelloWorld1.c ~
cp: overwrite '/home/abhignya/HelloWorld1.c'? y
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ cd ~
abhignya@hplaptop:~$ ls
2023  Abhignya_B_16  Assignment1  HelloWorld1.c  HelloWorld2.c  MCA2022  MCA2023  Program.txt  Unix_File_System
```

8. Remove the directories TestC & TestC-1.

   Ans.

```
abhignya@hplaptop:~$ cd ~/MCA2022/Abhignya_B_16/Unix_File_System/TestC
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestC$ ls
TestC-1
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestC$ rmdir TestC-1
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestC$ cd ..
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ rmdir TestC
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ ls
TestA  TestB
```

9. Delete the file C file from TestA-1.

   Ans.

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System$ cd TestA/TestA-1
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ ls
HelloWorld1.c
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ rm HelloWorld1.c
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ ls
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$
```

10. Rename the text file in the home directory.

    Ans.

```
abhignya@hplaptop:~/MCA2022/Abhignya_B_16/Unix_File_System/TestA/TestA-1$ cd ~
abhignya@hplaptop:~$ ls
2023  Abhignya_B_16  Assignment1  HelloWorld1.c  HelloWorld2.c  MCA2022  MCA2023  Program.txt  Unix_File_System
abhignya@hplaptop:~$ mv Program.txt C_Program.txt
abhignya@hplaptop:~$ ls
2023  Abhignya_B_16  Assignment1  C_Program.txt  HelloWorld1.c  HelloWorld2.c  MCA2022  MCA2023  Unix_File_System
abhignya@hplaptop:~$
```

11. Create a C file for a menu driven calculator.

    Ans.

```
abhignya@hplaptop:~$ cd ~/MCA2023/Abhignya_B_16/Assignment1
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ vim Calculator.c
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ gcc Calculator.c -o Calculator
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ ./Calculator
Enter an operator (+, -, *, /): +
Enter two operands: 12 25
12.00 + 25.00 = 37.00
```

12. Show the C file in the paged manner using **more** and **less** commands.

Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ more Calculator.c
#include <stdio.h>

int main() {
    char operator;
    double num1, num2;
    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);
    printf("Enter two operands: ");
    scanf("%lf %lf", &num1, &num2);

    switch(operator) {
        case '+':
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, num1 + num2);
            break;
        case '-':
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, num1 - num2);
            break;
        case '*':
            printf("%.2lf * %.2lf = %.2lf\n", num1, num2, num1 * num2);
            break;
        case '/':
            if (num2 != 0)
                printf("%.2lf / %.2lf = %.2lf\n", num1, num2, num1 / num2);
            else
                printf("Error! Division by zero.\n");
            break;
        default:
            printf("Error! Operator is not correct\n");
    }

    return 0;
}
```

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ less Calculator.c
```

```c
#include <stdio.h>

int main() {
    char operator;
    double num1, num2;
    printf("Enter an operator (+, -, *, /): ");
    scanf("%c", &operator);
    printf("Enter two operands: ");
    scanf("%lf %lf", &num1, &num2);

    switch(operator) {
        case '+':
            printf("%.2lf + %.2lf = %.2lf\n", num1, num2, num1 + num2);
            break;
        case '-':
            printf("%.2lf - %.2lf = %.2lf\n", num1, num2, num1 - num2);
            break;
        case '*':
            printf("%.2lf * %.2lf = %.2lf\n", num1, num2, num1 * num2);
            break;
        case '/':
            if (num2 != 0)
                printf("%.2lf / %.2lf = %.2lf\n", num1, num2, num1 / num2);
            else
                printf("Error! Division by zero.\n");
            break;
        default:
            printf("Error! Operator is not correct\n");
    }

    return 0;
}
```

13. Count the number of lines, words and characters separately.

    Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ wc Calculator.c
 33 101 882 Calculator.c
```

14. Compare the two C files.

    Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ cmp HelloWorld1.c HelloWorld2.c
HelloWorld1.c HelloWorld2.c differ: byte 59, line 4
```

15. Find what is common in two C files.

    Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ comm HelloWorld1.c HelloWorld2.c
                #include<stdio.h>
                int main(){
                        printf("Hello World!\n");


                        return 0;
                }
```

16. Find the difference in two C files.

    Ans.

```
abhignya@hplaptop:~/MCA2023/Abhignya_B_16/Assignment1$ diff HelloWorld1.c HelloWorld2.c
4,5d3
<
<
```

Assignment - 03
TOPIC: Process Scheduling- PART1

17. Write a C program to simulate the following non-preemptive CPU scheduling algorithms to find the turnaround time and waiting time for the above problem.
1. **FCFS**
2. **SJF**
3. **Priority**

a) FCFS

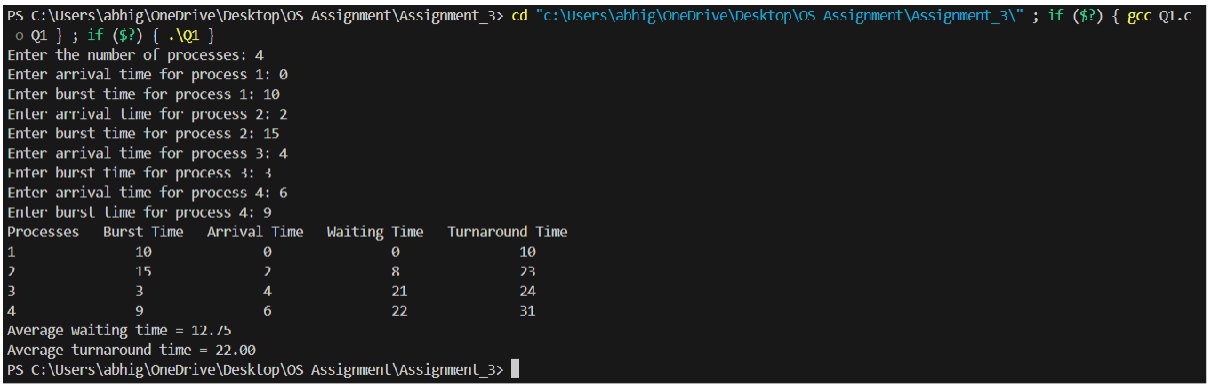
Code:-

```
#include <stdio.h>
struct Process {
    int pid;
    int burstTime;
    int arrivalTime;
    int waitingTime;
    int turnaroundTime;
};
void findWaitingTime(struct Process proc[], int n) {
    int serviceTime[n];
    serviceTime[0] = proc[0].arrivalTime;
    proc[0].waitingTime = 0;

    for (int i = 1; i < n ; i++) {
        serviceTime[i] = serviceTime[i-1] + proc[i-1].burstTime;
        proc[i].waitingTime = serviceTime[i] - proc[i].arrivalTime;
        if (proc[i].waitingTime < 0)
            proc[i].waitingTime = 0;
    }
}
void findTurnaroundTime(struct Process proc[], int n) {
    for (int i = 0; i < n ; i++) {
        proc[i].turnaroundTime = proc[i].burstTime + proc[i].waitingTime;
    }
}
void findAvgTime(struct Process proc[], int n) {
    findWaitingTime(proc, n);
    findTurnaroundTime(proc, n);
    int totalWaitingTime = 0, totalTurnaroundTime = 0;
    printf("Processes   Burst Time   Arrival Time   Waiting
Time   Turnaround Time\n");
    for (int i = 0; i < n; i++) {
        totalWaitingTime += proc[i].waitingTime;
        totalTurnaroundTime += proc[i].turnaroundTime;
        printf("%d\t\t%d\t\t%d\t\t%d\t\t%d\n", proc[i].pid,
proc[i].burstTime, proc[i].arrivalTime, proc[i].waitingTime,
proc[i].turnaroundTime);
    }
    printf("Average waiting time = %.2f\n", (float)totalWaitingTime /
(float)n);
    printf("Average turnaround time = %.2f\n", (float)totalTurnaroundTime /
(float)n);
}
```

```
int main() {
    int n;
    printf("Enter the number of processes: ");
    scanf("%d", &n);
    struct Process proc[n];
    for (int i = 0; i < n; i++) {
        proc[i].pid = i + 1;
        printf("Enter arrival time for process %d: ", i + 1);
        scanf("%d", &proc[i].arrivalTime);
        printf("Enter burst time for process %d: ", i + 1);
        scanf("%d", &proc[i].burstTime);
    }
    findAvgTime(proc, n);
    return 0;
}
```

Output:



b) SJF:

Code:-

```
class Process:
    def __init__(self, pid, arrivalTime, burstTime):
        self.pid = pid
        self.arrivalTime = arrivalTime
        self.burstTime = burstTime
        self.waitingTime = 0
        self.turnAroundTime = 0
        self.isCompleted = False

def SJF (processes, n):
    currentTime = 0
    completed = 0

    while completed < n:
        idx = -1
        lowestBurstTime = float('inf')

        for i in range(n):
            if processes[i].arrivalTime <= currentTime and not
processes[i].isCompleted:
```

```
                    if processes[i].burstTime < lowestBurstTime:
                        lowestBurstTime = processes[i].burstTime
                        idx = i

                    if processes[i].burstTime == lowestBurstTime:
                        if processes[i].arrivalTime <
processes[idx].arrivalTime:
                            idx = i

            if idx != -1:
                processes[idx].waitingTime = currentTime -
processes[idx].arrivalTime
                processes[idx].turnAroundTime = processes[idx].waitingTime +
processes[idx].burstTime
                currentTime += processes[idx].burstTime
                processes[idx].isCompleted = True
                completed += 1
            else:
                currentTime += 1

        totalWaitingTime = 0
        totalTurnAroundTime = 0

        print(f"\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround
Time")
        for i in range(n):

print(f"P{processes[i].pid}\t\t{processes[i].arrivalTime}\t\t{processes[i].
burstTime}\t\t{processes[i].waitingTime}\t\t{processes[i].turnAroundTime}")
            totalWaitingTime += processes[i].waitingTime
            totalTurnAroundTime += processes[i].turnAroundTime

        print(f"\nAverage Waiting Time = {totalWaitingTime / n}")
        print(f"Average Turn Around Time = {totalTurnAroundTime / n}")

def main():
    n = int(input("Enter the no of Processes: "))
    processes = []

    for i in range(n):
        print("Enter the details for Process:", i + 1)
        arrivalTime = int(input("Arrival Time: "))
        burstTime = int(input("Burst Time: "))
        processes.append(Process(i + 1, arrivalTime, burstTime))

    SJF(processes, n)

if __name__== "__main__":
    main()
```

Output:

```
PS C:\Users\abhig\OneDrive\Desktop\OS Assignment\Assignment_3> python -u "c:\Users\abhig\OneDrive\Desktop\OS Assignment\Assignment_3\Q2.py"
Enter the no of Processes: 5
Enter the details for Process: 1
Arrival Time: 2
Burst Time: 6
Enter the details for Process: 2
Arrival Time: 5
Burst Time: 2
Enter the details for Process: 3
Arrival Time: 1
Burst Time: 8
Enter the details for Process: 4
Arrival Time: 0
Burst Time: 3
Enter the details for Process: 5
Arrival Time: 4
Burst Time: 4

Process Arrival Time    Burst Time      Waiting Time    Turnaround Time
P1              2               6               1               7
P2              5               2               4               6
P3              1               8               14              22
P4              0               3               0               3
P5              4               4               7               11

Average Waiting Time = 5.2
Average Turn Around Time = 9.8
PS C:\Users\abhig\OneDrive\Desktop\OS Assignment\Assignment_3>
```

c) **Priority:**

Code:-

```
import java.util.*;
class Process {
    int pid;
    int arrivalTime;
    int burstTime;
    int priority;
    int waitingTime;
    int turnAroundTime;
    boolean isCompleted;
    public Process(int pid, int arrivalTime, int burstTime, int priority) {
        this.pid = pid;
        this.arrivalTime = arrivalTime;
        this.burstTime = burstTime;
        this.priority = priority;
        this.isCompleted = false;
    }
}
public class PriorityScheduling {
    static void calculate(Process[] processes, int n) {
        // Calculate waiting time and turnaround time
        int currentTime = 0;
        int completed = 0;
        while (completed < n) {
            int idx = -1;
            int highestPriority = Integer.MAX_VALUE;
            for (int i = 0; i < n; i++) {
                if (!processes[i].isCompleted && processes[i].arrivalTime
<= currentTime) {
                    //Sort the array based on their priority (Lowest Value
Highest Priority)
```

```
                    if (processes[i].priority < highestPriority) {
                        highestPriority = processes[i].priority;
                        idx = i;
                    }
                    // If two processes have the same priority, we choose
the one with the earlier arrival time
                    if (processes[i].priority == highestPriority) {
                        if (processes[i].arrivalTime <
processes[idx].arrivalTime) {
                            idx = i;
                        }
                    }
                }
            }
            if (idx != -1) {
                processes[idx].waitingTime = currentTime -
processes[idx].arrivalTime;
                processes[idx].turnAroundTime = processes[idx].waitingTime
+ processes[idx].burstTime;
                currentTime += processes[idx].burstTime;
                processes[idx].isCompleted = true;
                completed++;
            } else {
                currentTime++;
            }
        }
        double totalWaitingTime = 0;
        double totalTurnAroundTime = 0;
        System.out.println("\nProcess\tArrival Time\tPriority\tBurst
Time\tWaiting Time\tTurnaround Time");
        for (int i = 0; i < n; i++) {
            System.out.println("P" + processes[i].pid + "\t\t" +
processes[i].arrivalTime + "\t\t" + processes[i].priority + "\t\t" +
processes[i].burstTime + "\t\t" + processes[i].waitingTime + "\t\t" +
processes[i].turnAroundTime);
            totalWaitingTime += processes[i].waitingTime;
            totalTurnAroundTime += processes[i].turnAroundTime;
        }
        System.out.println("\nAverage Waiting Time = " + (totalWaitingTime
/ n));
        System.out.println("\nAverage Turn Around Time = " +
(totalTurnAroundTime / n));
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the no of Processes: ");
        int n = sc.nextInt();
        Process[] processes = new Process[n];
        for (int i = 0; i < n; i++) {
            System.out.println("Enter the details for Process: " + (i +
1));
            System.out.print("Arrival Time: ");
            int arrivalTime = sc.nextInt();
            System.out.print("Burst Time: ");
            int burstTime = sc.nextInt();
```

```
            System.out.print("Priority: ");
            int priority = sc.nextInt();
            processes[i] = new Process(i + 1, arrivalTime, burstTime,
priority);
        }
        sc.close();
        calculate(processes, n);
    }
}
```

Output:

```
PS C:\Users\abhig\OneDrive\Desktop\OS Assignment> cd "c:\Users\abhig\OneDrive\Desktop\OS Assignment\Assignment_3\" ; if ($?) { javac PriorityScheduling.java } ; if ($
?) { java PriorityScheduling }
Enter the no of Processes: 5
Enter the details for Process: 1
Arrival Time: 0
Burst Time: 5
Priority: 3
Enter the details for Process: 2
Arrival Time: 4
Burst Time: 3
Priority: 2
Enter the details for Process: 3
Arrival Time: 5
Burst Time: 2
Priority: 4
Enter the details for Process: 4
Arrival Time: 9
Burst Time: 7
Priority: 1
Enter the details for Process: 5
Arrival Time: 12
Burst Time: 6
Priority: 5

Process Arrival Time    Priority        Burst Time      Waiting Time    Turnaround Time
P1              0               3               5               0               5
P2              4               2               3               1               4
P3              5               4               2               3               5
P4              9               1               7               1               8
P5              12              5               6               5               11

Average Waiting Time = 2.0

Average Turn Around Time = 6.6
PS C:\Users\abhig\OneDrive\Desktop\OS Assignment\Assignment_3>
```