Lab Evaluation 1

Q.1 Implement the following Linux commands and display the output for each command

open, rmdir, mv, cp, cat, chmod, echo, nano, who

```
24ca041@server1:~$ mkdir -p os
24ca041@server1:~$ rmdir os
24ca041@server1:~$ pwd
/home/24ca041
```

```
24ca041@server1:~$ mv "Pictures"  "text files"
24ca041@server1:~$ pwd
/home/24ca041
```

```
24ca041@server1:~/text files$ cp 1.txt 2.txt
24ca041@server1:~/text files$
```

```
24ca041@server1:~/text files$ cat 1.txt 2.txt
txt 1
text 2
```

```
24ca041@server1:~/text files$ chmod +x h.sh
24ca041@server1:~/text files$ ./h.sh
Hello
```

```
24ca041@server1:~/text files$ echo "Hello"
Hello
```

```
24ca041@server1:~/Os Lab Evaluation$ xdg-open 1.txt
24ca041@server1:~/Os Lab Evaluation$
```

```
24ca041@server1:~/Os Lab Evaluation$ who -a
          system boot  2025-09-09 15:02
          run-level 5  2025-09-09 15:03
          pts/6        2025-09-10 14:07           157657 id=ts/6  term=0 exit=0
          pts/2        2025-09-12 14:05           805404 id=ts/2  term=0 exit=0
```

```
24ca041@server1:~$ nano 7.sh
24ca041@server1:~$ ls
 1.txt.save   Doc2        'OS Lab'              'text files'
 7.sh         Documents   'Os Lab Evaluation'    thinclient_drives
'C Lab'       Downloads    Pictures
'CPP Lab'     Music        Public
 Desktop      os5_1.c      Templates
```

Q.2 Write a bash script program for the following problem statements
a. To find Least Common Multiple (LCM) of two numbers, the program must take two natural numbers as input from the user and display the LCM of two numbers.

```
#!/bin/bash
gcd() {
    a=$1
    b=$2
    while [ $b -ne 0 ]; do
        temp=$b
        b=$((a % b))
        a=$temp
    done
    echo $a
}
lcm() {
    a=$1
    b=$2
    gcd_val=$(gcd $a $b)
    echo $(( (a * b) / gcd_val ))
}

read -p "Enter first number: " num1
read -p "Enter second number: " num2
lcm_result=$(lcm $num1 $num2)
echo "The LCM of $num1 and $num2 is: $lcm_result"
```

```
24ca041@server1:~/Os Lab Evaluation$ ./2.sh
Enter first number: 3
Enter second number: 5
The LCM of 3 and 5 is: 15
```

b. To append the text of one file to another file thrice. The program should take two file names as input and must append the contents of one file thrice to another file in the resultant third file. (Use of loop is expected)
For example
Input:
File1.txt: This is the first file of the project
File2.txt: This is the second file of the project.
Output:
File3.txt: This is the first file of the project, This is the first file of the project, This is the first file of the project. This is the second file of the project.

```
#!/bin/bash
  echo "Enter the source file name:"
  read src
  echo "Enter the destination file name:"
  read dest
  echo "Enter the result file name:"
  read result

  i=1
  while [ $i -le 3 ]; do
  cat "$src" >> "$result"
  i=$((i+1))
  done
  cat "$dest" >> "$result"
  echo "Done"
```

```
24ca041@server1:~/Os Lab Evaluation$ ./2_2.sh
Enter the source file name:
1.txt
Enter the destination file name:
2.txt
Enter the result file name:
3.txt
Done
```

Q.3. Write a program to implement preemptive shortest job first algorithm for CPU scheduling. The program must take the ready queue size and the number of process, arrival time and burst time as input and must display the waiting time for each process and the average waiting time.
For example:
Process Arrival Time BurstTime
P1 0 1
P2 1 6
P3 2 2
P4 9 5

```cpp
#include <bits/stdc++.h>
using namespace std;

int main() {
    int n;
    cout << "Enter number of processes: ";
    cin >> n;

    vector<int> AT(n), BT(n), rem(n), CT(n), TAT(n), WT(n);
    cout << "Enter Arrival Time and Burst Time for each process:\n";
    for (int i = 0; i < n; i++) {
        cout << "Process " << i + 1 << ": ";
        cin >> AT[i] >> BT[i];
        rem[i] = BT[i];
    }

    int complete = 0, time = 0;
    vector<int> timeline;

    while (complete < n) {
        int index = -1;
        int min_burst = INT_MAX;

        for (int i = 0; i < n; i++) {
            if (AT[i] <= time && rem[i] > 0 && rem[i] < min_burst) {
                min_burst = rem[i];
                index = i;
            }
        }

        if (index == -1) {
            timeline.push_back(0);
            time++;
        } else {
            timeline.push_back(index + 1);
            rem[index]--;
            time++;

            if (rem[index] == 0) {
                CT[index] = time;
                complete++;
            }
```

```
        }
    }

    float TAT_sum = 0, WT_sum = 0;

    for (int i = 0; i < n; i++) {
        TAT[i] = CT[i] - AT[i];
        WT[i] = TAT[i] - BT[i];
        TAT_sum += TAT[i];
        WT_sum += WT[i];
    }


    cout << "\nProcess\tAT\tBT\tCT\tTAT\tWT\n";
    for (int i = 0; i < n; i++) {
        cout << "P" << i + 1 << "\t" << AT[i] << "\t" << BT[i] << "\t"
             << CT[i] << "\t" << TAT[i] << "\t" << WT[i] << "\n";
    }

    cout << fixed << setprecision(2);
    cout << "\nAverage Turnaround Time = " << (TAT_sum / n) << "\n";
    cout << "Average Waiting Time = " << (WT_sum / n) << "\n";

    cout << "\nTimeline (Process at each time unit):\n";
    for (int i = 0; i < timeline.size(); i++) {
        if (timeline[i] == 0)
            cout << "Idle ";
        else
            cout << "P" << timeline[i] << " ";
    }
    cout << "\n";

    return 0;
}
```

```
24ca041@server1:~/Os Lab Evaluation$ g++ 3_2.cpp
24ca041@server1:~/Os Lab Evaluation$ ./a.out
Enter number of processes: 4
Enter Arrival Time and Burst Time for each process:
Process 1: 0 1
Process 2: 1 6
Process 3: 2 2
Process 4: 9 5

Process AT      BT      CT      TAT     WT
P1      0       1       1       1       0
P2      1       6       9       8       2
P3      2       2       4       2       0
P4      9       5       14      5       0

Average Turnaround Time = 4.00
Average Waiting Time = 0.50

Timeline (Process at each time unit):
P1 P2 P3 P3 P2 P2 P2 P2 P2 P4 P4 P4 P4 P4
```