CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

ISO Certified 9001:2015

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 44 years

DEPARTMENT OF COMPUTER SCIENCE

B.E – V SEMESTER

OPERATING SYSTEMS

LAB RECORD

COURSE CODE : 20CSC23

Academic Year

2022-23

CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. www.cbit.ac.in

ISO Certified 9001:2015

COMMITTED TO RESEARCH, INNOVATION AND EDUCATION 44 years

CERTIFICATE

Certified that this is the bonafide record of the practical work done during the academic year 2022-2023 by

Bheem Shashivardhan

Roll Number 160120748305 Section CSE-4 in the Laboratory of Operating System Lab of the Department of Computer Science Engineering.

Internal examiner *External examiner*

Head of the Department

Date :

INDEX

Week – 1

Shell Script Programs:

1) AIM : Write a shell script to display message.

PROGRAM :

```
echo "Hello World!"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Hello_Message.sh
[be19c4-65@cs1 ~]$ sh Hello_Message.sh
Hello World!
[be19c4-65@cs1 ~]$
```

2) AIM : Write a shell script to add two given numbers.

PROGRAM :

```
a=23

b=77

sum=$(($a+$b))

echo "Addition of $a and $b is : $sum"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi add.sh
[be19c4-65@cs1 ~]$ sh add.sh
Enter First number:
12
Enter Second number:
34
Sum :46
[be19c4-65@cs1 ~]$
```

3) AIM : Write a shell script to find given number is even or odd.

PROGRAM:

```
echo "Enter the number : "
read a
if [ $[$a % 2] -eq 0 ]
    then
    echo "$a is Even number!"
    else
    echo "$a is Odd number!"
fi
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi EvenOdd.sh
[be19c4-65@cs1 ~]$ sh EvenOdd.sh
Enter the number :
23
23 is Odd number!
[be19c4-65@cs1 ~]$
```

4) AIM : Write a shell script to find given year is leap year or not.

PROGRAM :

```
echo -n "Enter year (YYYY): "
read y
a = 'expr $y%4'
b = 'expr $y%100'
c = 'expr $y%400'
```

```
if[$a -eq 0 -a $b -ne - -o $c -eq 0]
then
echo "$y is leap year"
else
echo "$y is not a leap year"
fi
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi LeapYear.sh
[be19c4-65@cs1 ~]$ sh LeapYear.sh
Enter year (YYYY): 2020
```

5) AIM : Write a shell script to find factorial of a given number.

PROGRAM :

```
echo "Enter the number : "
read num
fact=1
for (( i=1; i<=$num; i++ ))
    do
    fact=$(($fact*$i))
    done
echo "Factorial of $num is : $fact"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi LeapYear.sh
[be19c4-65@cs1 ~]$ sh LeapYear.sh
Enter year (YYYY): 2020
```

6) AIM : Write a shell script to compare two numbers.

PROGRAM :

```
echo "Enter the Value of a : "
read a
echo "Enter the value of b : "
read b
if [ $a -eq $b ]
    then
    echo "$a and $b are Equal!"
elif [ $a -gt $b ]
    then
    echo "$a is greater than $b!"
else
    echo "$a is less than $b!"
fi
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Factorial.sh
[be19c4-65@cs1 ~]$ sh Factorial.sh
Enter the number :
6
Factorial of 6 is : 720
[be19c4-65@cs1 ~]$
```

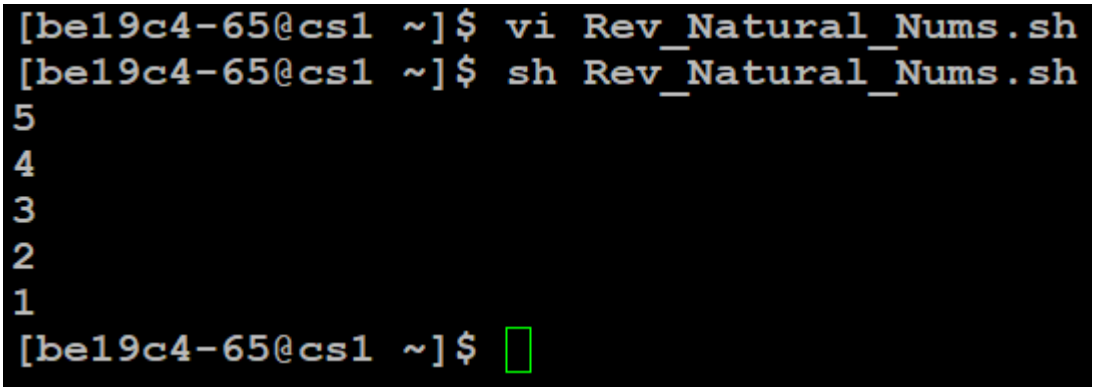
Week - 2

7) AIM : Write shell script to print nos as 5,4,3,2,1 using while loop.

PROGRAM :

```
num=5
while [ $num -ne 0 ]
    do
    echo "$num "
    num=$(($num-1))
    done
```

OUTPUT :



8) AIM : Write shell Script, using case statement to perform basic math operation as follows: + addition, - subtraction, x multiplication, / division.

PROGRAM :

```
echo "Enter the First Number : "
read a
echo "Enter the Second Number : "
```

```
read b
echo "Enter the option for operation (+ : Adddition, - : Subtraction, * : Multiplication, / : division, % : Modulus) : "
read choice
case $choice in
    "+")
    echo "addition of $a and $b is : $(($a+$b))"
    ;;
    "-")
    echo "Subtraction of $a and $b is : $(($a-$b))"
    ;;
    "*")
    echo "Multiplication of $a and $b is : $(($a*$b))"
    ;;
    "/")
    echo "Division of $a and $b is : $(($a/$b))"
    ;;
    "%")
    echo "Modulus of $a and $b is : $(($a%$b))"
    ;;
    *)
    echo "Invalid! Enter valid choice"
esac
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Switch_Case_Ex.sh
[be19c4-65@cs1 ~]$ sh Switch_Case_Ex.sh
Enter the First Number :
21
Enter the Second Number :
17
Enter the option for operation (+ : Adddition, - : Subtraction, * : Multiplication, / : division, % : Modulus) :
+
addition of 21 and 17 is : 38
[be19c4-65@cs1 ~]$
```

9) AIM : Write shell script to print given numbers sum of all digits, for eg. If no is 123 it's sum of all digit will be 1+2+3 = 6.

PROGRAM :

```
echo "Enter the Number : "
read num
sum=0
temp=$num
while [ $num -gt 0 ]
    do
    rem=$num%10
    sum=$(($sum+$rem))
    num=$(($num/10))
    done
echo "Sum of digits in $temp is : $sum"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Sum_Digits.sh
[be19c4-65@cs1 ~]$ sh Sum_Digits.sh
Enter the Number :
12356
Sum of digits in 12356 is : 17
[be19c4-65@cs1 ~]$
```

10) AIM : Write shell script to print given number in reverse order, eg. If no is 123 it must print as 321.

PROGRAM :

```
echo "Enter the Number : "
read num
rev=0
temp=num
while [ $num -gt 0 ]
    do
    rem=$num%10
    rev=$(($(($rev*10))+$rem))
    num=$(($num/10))
    done
echo "Reverse of $temp is : $rev"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Rev_of_Num.sh
[be19c4-65@cs1 ~]$ sh Rev_of_Num.sh
Enter the Number :
6213
Reverse of num is : 3126
[be19c4-65@cs1 ~]$
```

11) AIM : Write a shell script to check a number is prime (or) not.

PROGRAM :

```
echo "Enter the Number : "
read num
count=0
for (( i=1; i<=$num; i++ ))
    do
    if [ $[ $num%$i ] -eq 0 ]
        then
        count=$(($count+1))
    fi
    done
if [ $count -eq 2 ]
    then
    echo "$num is a Prime!"
else
    echo "$num is not a Prime!"
fi
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Prime_or_not.sh
[be19c4-65@cs1 ~]$ sh Prime_or_not.sh
Enter the Number :
23
23 is a Prime!
[be19c4-65@cs1 ~]$
```

12) AIM : Write a Shell script to print the first *n* fibonnaci numbers.

PROGRAM :

```
echo "Enter the Number : "
read num
a=0
b=1
count=0
echo "Fibonacci Series is : "
while [ $count -ne $num ]
    do
    echo $a
    res=$(($a+$b))
    a=$(($b))
    b=$(($res))
    count=$(($count+1))
    done
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Fibonnaci_Series.sh
[be19c4-65@cs1 ~]$ sh Fibonnaci_Series.sh
Enter the Number :
7
Fibonacci Series is :
0
1
1
2
3
5
8
[be19c4-65@cs1 ~]$
```

13) AIM : Write a Shell script to print the sum of *n* natural numbers.

PROGRAM :

```
echo "Enter the N value : "
read num
sum=0
for(( i=1; i<=num; i++ ))
    do
    sum=$(($sum+$i))
    done
echo "Sum of natural numbers upto $num is : $sum"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Sum_Of_Nums.sh
[be19c4-65@cs1 ~]$ sh Sum_Of_Nums.sh
Enter the N value :
15
Sum of natural numbers upto 15 is : 120
[be19c4-65@cs1 ~]$
```

14) **AIM :** Write a Shell script to find the GCD of two numbers.

PROGRAM :

```
echo "Enter the first Number: "
read a
echo "Enter the second Number: "
read b
m=$a
if [ $b -lt $m ]
then
m=$b
fi
while [ $m -ne 0 ]
do
x=`expr $a % $m`
y=`expr $b % $m`
if [ $x -eq 0 -a $y -eq 0 ]
then
echo "GCD of $a and $b is: $m"
break
fi
m=`expr $m - 1`
done
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi gcd.sh
[be19c4-65@cs1 ~]$ sh gcd.sh
Enter the first Number:
32
Enter the second Number:
4
GCD of 32 and 4 is: 4
[be19c4-65@cs1 ~]$
```

15) **AIM :** Write a Shell script to find the LCM of two numbers.

PROGRAM :

```
echo "Enter the first Number: "
read a
echo "Enter the second Number: "
read b
m=$a
if [ $b -lt $m ]
then
m=$b
fi
temp=$(($a*$b))
while [ $m -ne 0 ]
do
x=`expr $a % $m`
y=`expr $b % $m`
if [ $x -eq 0 -a $y -eq 0 ]
then
break
fi
m=`expr $m - 1`
done
lcm=$(($temp/$m))
echo "LCM of $a and $b is: $lcm"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi LCM_of_Num.sh
[be19c4-65@cs1 ~]$ sh LCM_of_Num.sh
Enter the first Number:
44
Enter the second Number:
11
LCM of 44 and 11 is: 44
[be19c4-65@cs1 ~]$
```

16) **AIM :** Write shell script to find the sum of the factors of a number.

PROGRAM :

```
echo "Enter the Number : "
read n
sum=0
for (( i=1; i<=n; i++ ))
do
    if [ $(($n%$i)) -eq 0 ]
    then
    sum=$(($sum+$i))
    fi
done
echo "Sum of factors of $n is : $sum"
```

OUTPUT :

```
[be19c4-65@cs1 ~]$ vi Sum_of_Factors.sh
[be19c4-65@cs1 ~]$ sh Sum_of_Factors.sh
Enter the Number :
46
Sum of factors of 46 is : 72
[be19c4-65@cs1 ~]$
```

Week – 3

CPU Scheduling Algorithms:

1) AIM : Implement FCFS CPU scheduling algorithms without arrival times.

PROGRAM :

```
def getWT(p, n, bt):
    wt = []
    wt.append(0)
    for i in range(1, n):
        # Waiting Time = Entering Time - Arrival Time
        wt.append(wt[i-1] + bt[i-1])
    return wt
def getTAT(p, n, wt, bt):
    tat = []
    for i in range(n):
        tat.append(bt[i]+wt[i]) # Turn around Time = Waiting Time + Burst Time
    return tat
def getAvgTime(p, n, bt):
    totalWT = totalTAT = 0
    wt = getWT(p, n, bt)
    tat = getTAT(p, n, wt, bt)
    print("First Come First Serve CPU Scheduling Alogirthm(without
AT):\n============================================================\n")
    print("PID\tBT\tWT\tTAT\n")
```

```
    for i in range(n):
        print("%d\t%d\t%d\t%d" % (p[i], bt[i], wt[i], tat[i]))
    print("Processes Execution Sequence : ")
    print('p'+" --> p".join([str(i) for i in p]))
    avgWT = sum(wt)/n
    avgTAT = sum(tat)/n
    print("\nAverage Waiting Time(WT): ", avgWT)
    print("Average Turn Around Time(TAT): ", avgTAT)
if __name__== "__main__":
    processes = [1, 2, 3, 4]
    burstTime = [2, 5, 3, 2]
    n = len(processes)
    getAvgTime(processes, n, burstTime)
```

OUTPUT :

```
First Come First Serve CPU Scheduling Alogirthm(without AT):
============================================================

PID     BT      WT      TAT

1       2       0       2
2       5       2       7
3       3       7       10
4       2       10      12

Processes Execution Sequence :
p1 --> p2 --> p3 --> p4

Average Waiting Time(WT):  4.75
Average Turn Around Time(TAT):  7.75
PS D:\BE SEM 5\Operating System(OS)>
```

2) AIM : Implement FCFS CPU scheduling algorithms with arrival times.

PROGRAM :

```
# 2. Implement FCFS CPU scheduling algorithms with arrival times
def getWT(p, n, at, bt):
    p2 = p[:]
    at2 = at[:]
    bt2 = bt[:]
    for i in range(n):
        for j in range(i+1, n):
            if at2[i] > at2[j]:
                p2[i], p2[j] = p2[j], p2[i]
                at2[i], at2[j] = at2[j], at2[i]
                bt2[i], bt2[j] = bt2[j], bt2[i]
    wt = []
    temp = [0]*(n+1)
    temp[0] = 0
    for i in range(n):
        temp[i+1] = temp[i]+bt2[i] # Calculating Entering Time
        # Waiting Time = Entering Time - Arrival Time
        wt.append([p2[i], temp[i]-at2[i]])
    return p2, at2, bt2, wt
def getTAT(p, n, wt, bt):
    tat = []
    for i in range(n):
```

```
        # Turn around Time = Waiting Time + Burst Time
        tat.append(bt[i]+wt[i][1])
    return tat
def getAvgTime(p, n, at, bt):
    totalWT = totalTAT = 0
    p2, at2, bt2, wt = getWT(p, n, at, bt)
    tat = getTAT(p, n, wt, bt2)
    wt = sorted(wt)
    print("First Come First Serve CPU Scheduling Alogirthm(with
At):\n=========================================================\n")
    print("PID\tAT\tBT\tWT\tTAT\n")
    for i in range(n):
        print("%d\t%d\t%d\t%d\t%d" % (p[i], at[i], bt[i], wt[i][1], tat[i]))
    print("\nProcesses execution sequence : ")
    print('p'+" --> p".join(list(map(str, p2))))
    avgWT = sum([i[1] for i in wt])/n
    avgTAT = sum(tat)/n
    print("\nAverage Waiting Time(WT): ", avgWT)
    print("Average Turn Around Time(TAT): ", avgTAT)
if __name__== "__main__":
    processes = [1, 2, 3, 4, 5]
    arrivalTime = [2, 1, 0, 3, 4]
    burstTime = [4, 3, 6, 5, 2]
    n = len(processes)
```

getAvgTime(processes, n, arrivalTime, burstTime)

OUTPUT :

```
First Come First Serve CPU Scheduling Alogirthm(with At):
=========================================================

PID     AT      BT      WT      TAT

1       2       4       7       6
2       1       3       5       8
3       0       6       0       11
4       3       5       10      15
5       4       2       14      16

Processes execution sequence :
p3 --> p2 --> p1 --> p4 --> p5

Average Waiting Time(WT):  7.2
Average Turn Around Time(TAT):  11.2
PS D:\BE SEM 5\Operating System(OS)>
```

3) AIM : Implement Non-preemptive SJF CPU scheduling algorithms without arrival times.

PROGRAM :

```
# 3. Implement Non-preemptive SJF CPU scheduling algorithms without arrival times

def getWT(p, n, bt):
    p2 = p[:]
    bt2 = bt[:]
    flag = False
    for i in range(n-1):
        for j in range(n-i-1):
            if (bt2[j] > bt2[j+1]):
                flag = True
                p2[j], p2[j+1] = p2[j+1], p2[j]
```

```
                bt2[j+1], bt2[j] = bt2[j], bt2[j+1]
            if flag == False:
                break
        wt = [[p2[0], 0]]
        for i in range(1, n):
            # Waiting Time = Entering Time - Arrival Time
            wt.append([p2[i], wt[i-1][1] + bt2[i-1]])
        return p2, bt2, wt
    def getTAT(n, wt, bt):
        tat = []
        for i in range(n):
            # Turn around Time = Waiting Time + Burst Time
            tat.append(wt[i][1]+bt[i])
        return tat
    def getAvgTime(p, n, bt):
        totalWT = totalTAT = 0
        p2, bt2, wt = getWT(p, n, bt)
        tat = getTAT(n, wt, bt2)
        wt = sorted(wt)
        print("Shorest Job First CPU Scheduling Alogirthm(without AT):
    \n=========================================================\n")
        print("PID\tBT\tWT\tTAT\n")
        for i in range(n):
            print("%d\t%d\t%d\t%d" % (p[i], bt[i], wt[i][1], tat[i]))
```

```
    print("\nProcesses execution sequence : ")
    print('p'+" --> p".join(list(map(str, p2))))
    avgWT = sum([i[1] for i in wt])/n
    avgTAT = sum(tat)/n
    print("\nAverage Waiting Time(WT): ", avgWT)
    print("Average Turn Around Time(TAT): ", avgTAT)
if __name__== "__main__":
    processes = [1, 2, 3, 4]
    burstTime = [7, 3, 1, 4]
    n = len(processes)
    getAvgTime(processes, n, burstTime)
```

OUTPUT :

```
Shorest Job First CPU Scheduling Alogirthm(without AT):
=======================================================

PID     BT      WT      TAT

1       7       8       1
2       3       1       4
3       1       0       8
4       4       4       15

Processes execution sequence :
p3 --> p2 --> p4 --> p1

Average Waiting Time(WT):  3.25
Average Turn Around Time(TAT):  7.0
PS D:\BE SEM 5\Operating System(OS)>
```

4) AIM : Implement Non-preemptive SJF CPU scheduling algorithms with arrival times

PROGRAM :

```
# 4. Implement Non-preemptive SJF CPU scheduling algorithms with arrival times.
def getWT(ts, n):
    wt = []
    temp = [0]*(n+1)
    temp[0] = 0
    for i in range(n):
        temp[i+1] = temp[i]+ts[i][2] # Calculating Entering Time
        # Waiting Time = Entering Time - Arrival Time
        wt.append(temp[i] - ts[i][1])
    return wt
def getTAT(ts, n, wt):
    tat = []
    for i in range(n):
        # Turn around Time = Waiting Time + Burst Time
        tat.append(ts[i][1]+wt[i])
    return tat
def getAvgTime(ts, n, pi):
    totalWT = totalTAT = 0
    wt = getWT(ts, n)
    tat = getTAT(ts, n, wt)
```

```
    print("Shorest Job First CPU Scheduling Alogirthm(without AT):
\n=====================================================\n")
    print("PID\tAT\tBT\tWT\tTAT\n")
    for i in range(n):
        print("%d\t%d\t%d\t%d\t%d" %
              (pi[i][0], pi[i][1], pi[i][2], wt[i], tat[i]))
    print("\nProcesses execution sequence : ")
    print('p'+" --> p".join([str(i[0]) for i in ts]))
    avgWT = sum(wt)/n
    avgTAT = sum(tat)/n
    print("\nAverage Waiting Time(WT): ", avgWT)
    print("Average Turn Around Time(TAT): ", avgTAT)
if __name__== "__main__":
    p1 = [1, 6, 7] # For each process : [Process ID, Arrival Time, Burst Time]
    p2 = [2, 4, 6]
    p3 = [3, 0, 2]
    p4 = [4, 2, 4]
    processes_with_BT = [p1, p2, p3, p4]
    printInfo = processes_with_BT
    timeStamps = sorted([[i[2], i[1], i[0]] for i in processes_with_BT])
    timeStamps = [[i[2], i[1], i[0]] for i in timeStamps]
    n = len(timeStamps)
    getAvgTime(timeStamps, n, printInfo)
```

OUTPUT :

```
Shorest Job First CPU Scheduling Alogirthm(without AT):
=======================================================

PID     AT      BT      WT      TAT

1       6       7       0       0
2       4       6       0       2
3       0       2       2       6
4       2       4       6       12

Processes execution sequence :
p3 --> p4 --> p2 --> p1

Average Waiting Time(WT):  2.0
Average Turn Around Time(TAT):  5.0
PS D:\BE SEM 5\Operating System(OS)\Programs>
```

5) AIM : Implement Preemptive SJF CPU scheduling algorithms with arrival times

PROGRAM :

```
# 5. Implement Preemptive SJF CPU scheduling algorithms with arrival times.

print("Preemptive Shorest Job First(SJF) CPU Scheduling Algorithm(with AT):\n=================================================================\n")

n = 5

bt = [0] * (n + 1)

at = [0] * (n + 1)

abt = [0] * (n + 1)

at = [0, 1, 4, 5, 6]

abt = [3, 2, 1, 5, 6]

at.append(0)

abt.append(0)

for i in range(n):
```

```
    bt[i] = [abt[i], at[i], i]
bt.pop(-1)
sumbt = 0
i = 0
ll = []
for i in range(0, sum(abt)):
    l = [j for j in bt if j[1] <= i]
    l.sort(key=lambda x: x[0])
    bt[bt.index(l[0])][0] -= 1
    for k in bt:
        if k[0] == 0:
            t = bt.pop(bt.index(k))
            ll.append([k, i + 1])
ct = [0] * (n + 1)
tat = [0] * (n + 1)
wt = [0] * (n + 1)
for i in ll:
    ct[i[0][2]] = i[1]
for i in range(len(ct)):
    tat[i] = ct[i] - at[i]
    wt[i] = tat[i] - abt[i]
ct.pop(-1)
wt.pop(-1)
tat.pop(-1)
```

```
abt.pop(-1)

at.pop(-1)

print()

print('BT\tAT\tCT\tTAT\tWT')

for i in range(len(ct)):

   print("{}\t{}\t{}\t{}\t{}\n".format(abt[i], at[i], ct[i], tat[i], wt[i]))

print('Average Waiting Time(WT) = ', sum(wt)/len(wt))

print('Average Turnaround Time(TAT) = ', round(sum(tat)/len(tat), 2))
```

OUTPUT :

```
Preemptive Shorest Job First(SJF) CPU Scheduling Algorithm(with AT):
================================================================


BT      AT      CT      TAT     WT
3       0       3       3       0

2       1       5       4       2

1       4       6       2       1

5       5       11      6       1

6       6       17      11      5

Average Waiting Time(WT) =  1.8
Average Turnaround Time(TAT) =  5.2
PS D:\BE SEM 5\Operating System(OS)>
```

6) AIM: Implement Priority CPU scheduling algorithms with arrival times.

PROGRAM:

```
totalprocess = 5
proc = []
for i in range(5):
   l = []
   for j in range(4):
      l.append(0)
    proc.append(l)
def get_wt_time( wt):
   service = [0] * 5
   service[0] = 0
   wt[0] = 0
   for i in range(1, totalprocess):
      service[i] = proc[i - 1][1] + service[i - 1]
      wt[i] = service[i] - proc[i][0] + 1
      if(wt[i] < 0) :
         wt[i] = 0
def get_tat_time(tat, wt):
    for i in range(totalprocess):
      tat[i] = proc[i][1] + wt[i]
def findgc():
   wt = [0] * 5
   tat = [0] * 5
   wavg = 0
   tavg = 0
```

```
    get_wt_time(wt)


    get_tat_time(tat, wt)
    stime = [0] * 5
    ctime = [0] * 5
    stime[0] = 1
    ctime[0] = stime[0] + tat[0]
    for i in range(1, totalprocess):
        stime[i] = ctime[i - 1]
        ctime[i] = stime[i] + tat[i] - wt[i]
    print("Process_no\tStart_time\tComplete_time",
            "\tTurn_Around_Time\tWaiting_Time")
    for i in range(totalprocess):
        wavg += wt[i]
        tavg += tat[i]
        print(proc[i][3], "\t\t", stime[i],
                    "\t\t", end = " ")
        print(ctime[i], "\t\t", tat[i], "\t\t\t", wt[i])
    # display the average waiting time
    # and average turn around time
    print("Average waiting time is : ", end = " ")
    print(wavg / totalprocess)
    print("average turnaround time : " , end = " ")
    print(tavg / totalprocess)
if __name__=="__main__":
    arrivaltime = [1, 2, 3, 4, 5]
```

```
    bursttime = [3, 5, 1, 7, 4]
    priority = [3, 4, 1, 7, 8]
    for i in range(totalprocess):

        proc[i][0] = arrivaltime[i]
        proc[i][1] = bursttime[i]
        proc[i][2] = priority[i]
        proc[i][3] = i + 1
    proc = sorted (proc, key = lambda x:x[2])
    proc = sorted (proc)
    findgc()
```

Output:

```
Process_no      Start_time      Complete_time   Turn_Around_Time        Waiting_Time
1               1               4               3                       0
2               4               9               7                       2
3               9               10              7                       6
4               10              17              13                      6
5               17              21              16                      12
Average waiting time is :  5.2
average turnaround time :  9.2
PS D:\SEMISTERS-SUB\PYTHON>
```

7) AIM: Implement Priority CPU scheduling algorithms without arrival times.

PROGRAM:

```
def findWaitingTime(processes, n, wt):
    wt[0] = 0
    for i in range(1, n):
        wt[i] = processes[i - 1][1] + wt[i - 1]
```

```
def findTurnAroundTime(processes, n, wt, tat):
    for i in range(n):
        tat[i] = processes[i][1] + wt[i]
def findavgTime(processes, n):
    wt = [0] * n


    tat = [0] * n
    findWaitingTime(processes, n, wt)
    findTurnAroundTime(processes, n, wt, tat)
    print("\nProcesses    Burst Time    Waiting",
          "Time    Turn-Around Time")
    total_wt = 0
    total_tat = 0
    for i in range(n):
        total_wt = total_wt + wt[i]
        total_tat = total_tat + tat[i]
        print(" ", processes[i][0], "\t\t",
                  processes[i][1], "\t\t",
                  wt[i], "\t\t", tat[i])
    print("\nAverage waiting time = %.5f "%(total_wt /n))
    print("Average turn around time = ", total_tat / n)
def priorityScheduling(proc, n):
    proc = sorted(proc, key = lambda proc:proc[2],
                              reverse = True);
    print("Order in which processes gets executed")
    for i in proc:
```

```
        print(i[0], end = " ")
    findavgTime(proc, n)
if __name__=="_main_":
    proc = [[1, 10, 0],
            [2, 5, 1],
            [3, 8, 2]]
    n = 3
    priorityScheduling(proc, n)
```

Output:

```
Order in which processes gets executed
3 2 1
Processes    Burst Time    Waiting Time    Turn-Around Time
  3             8              0              8
  2             5              8              13
  1             10             13             23

Average waiting time = 7.00000
Average turn around time =  14.666666666666666
PS D:\SEMISTERS-SUB\PYTHON>
```

8) AIM: Implement Preempitive priority CPU scheduling algorithms with arrival times.

PROGRAM:

```
totalprocess = 6
proc = []
for i in range(6):
    l = []
    for j in range(5):
        l.append(0)
     proc.append(l)
# Using FCFS Algorithm to find Waiting time
def get_wt_time( wt ):
```

```
    # declaring service array that stores
    # cumulative burst time
    service = [0] * 6
    # Initialising initial elements
    # of the arrays
    service[0] = 0
    wt[0] = 0
    for i in range(1, totalprocess):
        service[i] = proc[i - 1][1] + service[i - 1]

        wt[i] = service[i] - proc[i][0] + 1
        # If waiting time is negative,
        # change it o zero
        if(wt[i] < 0):
            wt[i] = 0
def get_tat_time(tat, wt):
    # Filling turnaroundtime array
    for i in range(totalprocess):
        tat[i] = proc[i][1] + wt[i]
def findgc():
    # Declare waiting time and
    # turnaround time array
    wt = [0] * 6
    tat = [0] * 6
    wavg = 0
    tavg = 0
```

```
# Function call to find waiting time array
get_wt_time(wt)
# Function call to find turnaround time
get_tat_time(tat, wt)
stime = [0] * 6
ctime = [0] * 6
stime[0] = 1
ctime[0] = stime[0] + tat[0]
# calculating starting and ending time
for i in range(1, totalprocess):
    stime[i] = ctime[i - 1]
    ctime[i] = stime[i] + tat[i] - wt[i]
print("Process_no\tStart_time\tComplete_time",
          "\tTurn_Around_Time\tWaiting_Time")
# display the process details
for i in range(totalprocess):
    wavg += wt[i]
    tavg += tat[i]
    print(proc[i][3], "\t\t", stime[i],
                  "\t\t", end = " ")
    print(ctime[i], "\t\t", tat[i], "\t\t\t", wt[i])
# display the average waiting time
# and average turn around time
print("Average waiting time is : ", end = " ")
print(wavg / totalprocess)
print("average turnaround time : " , end = " ")
```

```
    print(tavg / totalprocess)
if __name__=="_main_":
    arrivaltime = [0,1,2,3,4,5,6]
    bursttime = [1,7,3,6,5,15,8]
    priority = [2,6,3,5,4,10,9]
    for i in range(totalprocess):
        proc[i][0] = arrivaltime[i]
        proc[i][1] = bursttime[i]
        proc[i][2] = priority[i]
        proc[i][3] = i + 1
    # Using inbuilt sort function
    proc = sorted (proc, key = lambda x:x[2])
    proc = sorted (proc)
    # Calling function findgc for
    # finding Gantt Chart
    findgc()
```

Output:

```
Process_no      Start_time      Complete_time   Turn_Around_Time        Waiting_Time
1               1               2               1                       0
2               2               9               8                       1
3               9               12              10                      7
4               12              18              15                      9
5               18              23              19                      14
6               23              38              33                      18
Average waiting time is :  8.166666666666666
average turnaround time :  14.333333333333334
PS D:\SEMISTERS-SUB\PYTHON>
```

9) AIM: Implement Round Robin CPU scheduling algorithms with arrival times.

PROGRAM:

```
if __name__== '__main__':
    print("Enter Total Process Number: ")
    total_p_no = int(input())
    total_time = 0
    total_time_counted = 0
    # proc is process list
    proc = []
    wait_time = 0
    turnaround_time = 0
    for _ in range(total_p_no):
        print("Enter process arrival time and burst time")
        input_info = list(map(int, input().split(" ")))
        arrival, burst, remaining_time = input_info[0], input_info[1], input_info[1]
        proc.append([arrival, burst, remaining_time, 0])
        total_time += burst
    print("Enter time quantum")
    time_quantum = int(input())
    while total_time != 0:
        for i in range(len(proc)):
            if proc[i][2] <= time_quantum and proc[i][2] >= 0:
                total_time_counted += proc[i][2]
                total_time -= proc[i][2]
                proc[i][2] = 0
            elif proc[i][2] > 0:
```

```
            proc[i][2] -= time_quantum
            total_time -= time_quantum
            total_time_counted += time_quantum
        if proc[i][2] == 0 and proc[i][3] != 1:
            wait_time += total_time_counted - proc[i][0] - proc[i][1]
            turnaround_time += total_time_counted - proc[i][0]
            proc[i][3] = 1
    print("\nAvg Waiting Time is ", (wait_time * 1) / total_p_no)
    print("Avg Turnaround Time is ", (turnaround_time * 1) / total_p_no)
```

Output:

```
Enter Total Process Number:
5
Enter process arrival time and burst time
0 20
25 25
30 35
60 15
100 10
Enter time quantum
5

Avg Waiting Time is  14.0
Avg Turnaround Time is  35.0
PS D:\SEMISTERS-SUB\PYTHON>
```

Week - 5

File Handling Commands

COMMANDS:

a) Mkdir

```
[be19c4-29@cs1 Week-5]$ mkdir -m a=rwx m_option
[be19c4-29@cs1 Week-5]$ cd m_option
```

```
[be19c4-29@cs1 Week-5]$ mkdir -v v_option1 v_option2
mkdir: created directory `v_option1'
mkdir: created directory `v_option2'
[be19c4-29@cs1 Week-5]$
```

```
[be19c4-29@cs1 Week-5]$ mkdir -p -v first/second
mkdir: created directory `first'
mkdir: created directory `first/second'
[be19c4-29@cs1 Week-5]$
```

b) Rmdir

```
[be19c4-29@cs1 Week-5]$ ls
fisrt  m_option  v_option1  v_option2
[be19c4-29@cs1 Week-5]$ rmdir -p v_option1
[be19c4-29@cs1 Week-5]$ ls
fisrt  m_option  v_option2
[be19c4-29@cs1 Week-5]$
```

```
[be19c4-29@cs1 Week-5]$ rmdir -v v_option2
rmdir: removing directory, `v_option2'
[be19c4-29@cs1 Week-5]$
```

c) Unlink

```
[be19c4-29@cs1 Week-5]$ mkdir first
[be19c4-29@cs1 Week-5]$ cd first
[be19c4-29@cs1 first]$ vi sample.sh


[be19c4-29@cs1 first]$ unlink sample.sh
[be19c4-29@cs1 first]$ ls
[be19c4-29@cs1 first]$
```

d) Touch

```
[be19c4-29@cs1 first]$ touch -a sample.sh
[be19c4-29@cs1 first]$ ls
sample.sh
```

```
[be19c4-29@cs1 first]$ touch -m sample.sh
[be19c4-29@cs1 first]$
```

e) Cp

```
[be19c4-29@cs1 Week-5]$ vi a.txt
[be19c4-29@cs1 Week-5]$ cp a.txt b.txt
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  file1  first  fisrt  m_op
[be19c4-29@cs1 Week-5]$
```

f) Mv

```
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  file1
[be19c4-29@cs1 Week-5]$ mv a.txt c.txt
[be19c4-29@cs1 Week-5]$ ls
b.txt  c.txt  file1
[be19c4-29@cs1 Week-5]$
```

g) Cat

```
[be19c4-29@cs1 Week-5]$ vi a.txt
[be19c4-29@cs1 Week-5]$ cat a.txt
sup
[be19c4-29@cs1 Week-5]$
```

```
[be19c4-29@cs1 Week-5]$ cat > d.txt

[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt  file1
[be19c4-29@cs1 Week-5]$
```

```
[be19c4-29@cs1 Week-5]$ tac a.txt
```

h) Ln

```
[be19c4-29@cs1 Week-5]$ ln a.txt b.txt
ln: creating hard link `b.txt': File exists
[be19c4-29@cs1 Week-5]$
```

i) Rm

```
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt  file1
[be19c4-29@cs1 Week-5]$ rm file1
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt
[be19c4-29@cs1 Week-5]$
```

```
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt  sample
[be19c4-29@cs1 Week-5]$ rm -r sample
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt
[be19c4-29@cs1 Week-5]$
```

j) Umask

```
[be19c4-29@cs1 Week-5]$ umask
0022
[be19c4-29@cs1 Week-5]$
```

k) Chdir

- It is used to change the directory
- It is a function

l) Chmod

```
[be19c4-29@cs1 Week-5]$ chmod u=r a.txt
[be19c4-29@cs1 Week-5]$ vi a.txt
```

```
-- INSERT -- W10: Warning: Changing a readonly file
```

m) Ls

```
[be19c4-29@cs1 Week-5]$ ls
a.txt  b.txt  c.txt  d.txt
```

n) Chown

```
[be19c4-29@cs1 Week-5]$ chown be19c4-29 a.txt
[be19c4-29@cs1 Week-5]$
```

o) Chgrp

```
[be19c4-29@cs1 Week-5]$ sudo chgrp a.txt b.txt
[sudo] password for be19c4-29:
```

p) Cd

```
[be19c4-29@cs1 Week-5]$ mkdir sample
[be19c4-29@cs1 Week-5]$ cd sample
[be19c4-29@cs1 sample]$
```

Text Handling Commands

COMMANDS:

a) Head

```
[be19c4-29@cs1 Week-5]$ vi a.txt
[be19c4-29@cs1 Week-5]$ head a.txt
a
b
c
d
e
f
g
h
i
j
[be19c4-29@cs1 Week-5]$
```

b) Tail

```
[be19c4-29@cs1 Week-5]$ tail a.txt
f
g
h
i
j
k
l
m
n
o
[be19c4-29@cs1 Week-5]$
```

c) Diff

```
[be19c4-29@cs1 Week-5]$ vi a.txt
[be19c4-29@cs1 Week-5]$ vi b.txt
[be19c4-29@cs1 Week-5]$ diff a.txt b.txt
1,5c1,5
< a
< b
< c
< d
< e
---
> 1
> 2
> 3
> 4
> 5
```

e) Echo

```
[be19c4-29@cs1 Week-5]$ echo "Hello World"
Hello World
[be19c4-29@cs1 Week-5]$
```

Program 3:

i) Tar

```
[be19c4-29@cs1 Week-5]$ tar cvf file.tar *.txt
a.txt
b.txt
c.txt
d.txt
```

ii)cpio

```
[be19c4-29@cs1 Week-5]$ cpio -i < archive
1 block
```

```
[be19c4-29@cs1 Week-5]$ cpio -o < e.txt > archive
1 block
```

WEEK - 6

1. NETWORK COMMANDS

(a) **Who:** The who command is used to get information about currently logged in user on to system.

Syntax: $who [options] [filename]

(b) **Ftp:** The ftp command connects a computer system to a remote server using the FTP protocol. Once connected, it also lets users transfer files between the local machine and the remote system,and manage files and directories on the remote system.

Syntax: ftp [IP]

(c) **Telnet:** telnet command is used to create a remote connection with a system over a TCP/IPnetwork. It allows us to administrate other systems by the terminal.

Syntax: telnet hostname/IP address

sudo apt update

sudo apt install telnetd -y

(d) **rlogin:** rlogin starts a terminal session on the remote host host. The standard Berkeley "rhosts"authorization mechanism is used.

Syntax: *rlogin [-8EKLdx] [-e char] [-l username] host*

(e) **Finger:** Finger command is a user information lookup command which gives details of all the userslogged in. This tool is generally used by system administrators. It provides details like login name, user name, idle time, login time, and in some cases their email address even.

Syntax: finger [username]

Arp: arp command manipulates the System's ARP cache. It also allows a complete dump of the ARP cache. ARP stands for Address Resolution Protocol. The primary function of this protocol is to resolve the IP address of a system to its mac address, and hence it works between level 2(Data linklayer) and level 3(Network layer).

Syntax: *arp [-v] [-i if] [-H type] -a [hostname]*

(f) Mount: mount command is used to mount the filesystem found on a device to big tree *structure(Linux filesystem) rooted at '/'.*

Syntax: mount -t type device dir

(g) Umount: umount can be used to detach these devices from the Tree.

Syntax: umount [username]

(h) uname: The command 'uname' displays the information about the system.

Syntax: uname -a

(i) W: w command in Linux is used to show who is logged on and what they are doing. This commandshows the information about the users currently on the machine and their processes.

Syntax: w [options] user [...]

(j) Find: The find command in UNIX is a command line utility for walking a file hierarchy. It can be used to find files and directories and perform subsequent operations on them. It supports searchingby file, folder, name, creation date, modification date, owner and permissions. By using the '-exec' other UNIX commands can be executed on files or folders found.

Syntax: *$ find [where to start searching from]*

[expression determines what to find] [-options] [what to find]

W -h

W -u

W -s

2. PROCESS COMMANDS:

(a) Ps: ps command is used to list the currently running processes and their PIDs along with someother information depends on different options.

Syntax: ps [options]

(b) Ulimit: ulimit is admin access required Linux shell command which is used to see, set, or limit theresource usage of the current user.

Syntax: ulimit -a

(c) Kill: kill command in Linux (located in /bin/kill), is a built-in command which is used to terminateprocesses manually. kill command sends a signal to a process which terminates the process.

Syntax: $kill -l

3. OTHER COMMANDS:

(a) Cal: cal command is a calendar command in Linux which is used to see the calendar of a specificmonth or a whole year.

Syntax: cal [[month] year]

(b) Pwd: pwd stands for Print Working Directory. It prints the path of the working directory, startingfrom the root.

Syntax: pwd -L

Name: Bheem Shashivardhan **Section:** CSE – AI&ML **Roll No.:** 160120748305

Week – 7

Deadlock Avoidance Algorithms:

1) **AIM:** Implement Bankers algorithm for deadlock avoidance . Print the safe sequence if the system is in safe state, Otherwise inform that there no safe sequence. Also implement granting of resource request .

PROGRAM:

```
# Implementation of Resource Request Algorithm for Deadlock Avoidance

def getCompare(a, b):
    for i in range(len(a)):
        if a[i] > b[i]:
            return False
    return True

def getSum(a, b, sum=True):
    if sum:
        return [a[i] + b[i] for i in range(len(a))]
    return [a[i] - b[i] for i in range(len(a))]

def getSafeSeq(avail, max, alloc):
    procNo = len(max)
    resoNo = len(avail)
    need = [getSum(max[i], alloc[i], sum=False) for i in range(procNo)]
    finish = [False for i in range(procNo)]
    visited = [0 for i in range(procNo)]
    visitedInd = 0
    for k in range(procNo):
        for i in range(procNo):
            if finish[i] == False and getCompare(need[i], avail):
                finish[i] = True
```

```
            avail = getSum(avail, alloc[i])
            visited[visitedInd] = i
            visitedInd += 1
    if all(finish):
        return [True, visited]
    else:
        return [False]
def resourceRequest(req, avail, max, alloc):
    procNo = len(max)
    resNo = len(avail)
    reqProc = req[0]
    need = getSum(max[reqProc], alloc[reqProc], False)
    if getCompare(req[1], need):
        if getCompare(req[1], avail):
            avail = getSum(avail, req[1], False)
            alloc[reqProc] = getSum(alloc[reqProc], req[1])
            res = getSafeSeq(avail, max, alloc)
            if len(res) == 2:
                print("Request will be Granted!(Because, after granting resources system will be in Safe state)\nSafe Sequence: " +
                      " -->".join(["P" + str(i) for i in res[1]]))
            else:
                print(
                    "Request will not be granted!. Because, if the request is granted then system will be in unsafe state(Deadlock)")
        else:
            print("No available resources to grant for process - ",
```

```
            req[0], sep="")
    else:
        print("Resource Request is not legal!(i.e., request > need)")
if__name__== "_main_":
    available = [3, 3, 2]
    maximum = [[7, 5, 3], [3, 2, 2], [9, 0, 2], [2, 2, 2], [4, 3, 3]]
    allocated = [[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2]]
    # getSafeSeq(available, max, allocated)
    prono = 1
    # options : [4, [3, 3, 0]] or [0, [4, 2, 1]] or [2, [3, 1, 0]] or [0, [0, 2, 0]]
    request = [prono, [1, 0, 2]]
    resourceRequest(request, available, maximum, allocated)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs\Deadlock_Avoidance> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe "d:/BE SEM 5/Operati
ng System(OS)/Programs/Deadlock_Avoidance/Resource_Request_Algo(Deadlock_Avoidance) (1).py"
Request will be Granted!(Because, after granting resources system will be in Safe state)
Safe Sequence: P1 -->P3 -->P4 -->P0 -->P2
```

Memory Partitioning Algorithms:

2) AIM: Implement equal sized fixed partitioning algorithm and calculate total internal fragmentation.

PROGRAM :

```
a=int(input("enter memory partition size:"))
b=int(input("enter number of partition:")) c=a/b
print("each partition size are:",c) e=list(map(int,input("enter the each job
size").split())) s=0
for i in e: if
    i<=c:
        d=c-i
```

```
        s=s+d

        print("total internal fragmentation for job",i,"is",d) else:

        print("job size",i,"cannot be executed") print("total internal
 fragmentation for all jobs are:",s)
```

OUTPUT:

```
enter memory partition size:345
enter number of partition:23
each partition size are: 15.0
enter the each job size12
total internal fragmentation for job 12 is 3.0
total internal fragmentation for all jobs are: 3.0
```

3) AIM: Implement Un equal sized fixed partitioning algorithm and calculate total internal fragmentation.

PROGRAM :

```
#unequal partition sizes a=int(input("enter

memory size:"))

d=list(map(int,input("Enter the partition sizes:").split()))

c=list(map(int,input("Enter each job size:").split())) l1=[]

s=0

for i in d: s=s+i

    if s<=a:

        l1.append(i)

l1.append(a-sum(l1))

print("size of unequal partitions are:",l1) for j in

range(len(d)):

    if j>len(l1):

     break

    else:

        if c[j]<=l1[j]:

            print("total internal fragmentation for job",c[j],"is",l1[j]-c[j]) else:

            print("job of size",c[j],"cannot be fitted")
```

OUTPUT:

```
enter memory size:345
1Enter the partition sizes:3
Enter each job size:12
size of unequal partitions are: [3, 34
job of size 12 cannot be fitted
```

4) AIM: Implement dynamic partitioning algorithm and calculate total external fragmentation.

PROGRAM :

```
tm = int(input("Enter total memory size:")) n =

int(input("Enter no. of jobs:"))

summ = 0

for i in range(n):

    j_s = int(input("Enter job size:"))

    if j_s <= tm and summ <= tm-summ: print("Job",i+1,"allocated in main

        memory.") summ+=j_s

    else:

        print("Space unavailable.") print("External

fragmentation:",tm-summ)
```

OUTPUT:

```
Enter total memory size:345
Enter no. of jobs:13
Enter job size:1
Job 1 allocated in main memory.
Enter job size:23
Job 2 allocated in main memory.
Enter job size:23
Job 3 allocated in main memory.
Enter job size:234
Job 4 allocated in main memory.
Enter job size:12
Space unavailable.
```

WEEK – 8

1) AIM: Implement FIFO(First In First Out) page replacement algorithm.

PROGRAM:

```
# Implementing FIFO(First In First Out) page replacement Algorithm.

def getPageFaults(referStr, fr):
    mem = []
    ptr = pageFaults = 0
    for i in referStr:
        if i not in mem:
            if len(mem) < fr:
                mem.append(i)
            else:
                mem[ptr] = i
                ptr = (ptr+1) % fr
            pageFaults += 1
            print(mem)
    return pageFaults
if __name__== "__main__":
    # [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1]
    referenceString = [1, 2, 3, 4, 5, 3, 1, 6,
                       7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2]
    frames = 3
    print("\nReference String: ", referenceString, "\nNo. of page Faults is: ",
          getPageFaults(referenceString, frames))
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs> python -u "d:\BE SEM 5\Operating System(OS)\Programs\FIFO_Page_Replacement_Algo.py"
[1]
[1, 2]
[1, 2, 3]
[4, 2, 3]
[4, 5, 3]
[4, 5, 1]
[6, 5, 1]
[6, 7, 1]
[6, 7, 8]
[9, 7, 8]
[9, 5, 8]
[9, 5, 4]
[2, 5, 4]

Reference String:  [1, 2, 3, 4, 5, 3, 1, 6, 7, 8, 7, 8, 9, 7, 8, 9, 5, 4, 5, 4, 2]
No. of page Faults is:  13
```

2) AIM: Implement LRU(Least Recently Used) page replacement algorithm.

PROGRAM:

```
# Implementing OPTIMAL Page Replacement Algorithm.

def getPageFaults(referStr, fr):
    pageFaults = 0
    mem = []
    mem_swap = [0 for i in range(fr)]
    for i in range(len(referStr)):
        if referStr[i] not in mem:
            if len(mem) < fr:
                mem.append(referStr[i])
            else:
                for j in range(len(mem)):
                    if mem[j] not in referStr[i-1::-1]:
                        mem[j] = referStr[i]
                    else:
                        mem_swap[j] = referStr[i-1::-1].index(mem[j])
                mem[mem_swap.index(max(mem_swap))] = referStr[i]
            pageFaults += 1
```

```
            print(mem)
    return pageFaults
if __name__== "__main__":
    referenceString = [7, 0, 1, 2, 0, 3, 0,
                       4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1]
    frames = 3
    print("\nReference String: ", referenceString,
          "\nNo. of Page Faults: ", getPageFaults(referenceString, frames))
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs> python -u "d:\BE SEM 5\Operating System(OS)\Programs\LRU(new)_Page_Replacement_Algo.py"
[7]
[7, 0]
[7, 0, 1]
[2, 0, 1]
[2, 0, 3]
[4, 0, 3]
[4, 0, 2]
[4, 3, 2]
[0, 3, 2]
[1, 3, 2]
[1, 0, 2]
[1, 0, 7]

Reference String:  [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1]
No. of Page Faults:  12
```

3) AIM: Implement OPTIMAL page replacement algorithm.

PROGRAM:

```
# implementing LFU(Least Recently Used) Page Replacement Algorithm.
def getPageFaults(referStr, fr):
    pageFaults = 0
    mem = []
    mem_swap = [0 for i in range(len(mem))]
    for i in range(len(referStr)):
        if referStr[i] not in mem:
            if len(mem) < fr:
                mem.append(referStr[i])
```

```
        else:
            mem_swap = [0 for i in range(len(mem))]
            for k in range(len(mem)):
                if mem[k] not in referStr[i+1:]:
                    mem[k] = referStr[i]
                    break
                else:
                    mem_swap[k] = referStr[i+1:].index(mem[k])
            mem[mem_swap.index(max(mem_swap))] = referStr[i]
        pageFaults += 1
        print(mem)
    return pageFaults
if __name__== "__main__":
    referenceString = [7, 0, 1, 2, 0, 3, 0,
                       4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1]
    frames = 4
    print("\nReference String: ", referenceString,
          "\nNo. of Page Faults: ", getPageFaults(referenceString, frames))
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs> python -u "d:\BE SEM 5\Operating System(OS)\Programs\OPTIMAL(new)_Page_Replacement_Algo.py"
[7]
[7, 0]
[7, 0, 1]
[7, 0, 1, 2]
[3, 0, 1, 2]
[3, 0, 4, 2]
[1, 0, 4, 2]
[7, 0, 7, 2]
[1, 0, 7, 2]

Reference String:  [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1]
No. of Page Faults:  9
```

WEEK – 9

File Allocation Methods:

1) AIM: Implementation of Contiguous File Allocation method.

PROGRAM:

```
def getFreeBlocks(dsk):
    return [str(i + 1) for i in range(len(dsk)) if dsk[i] == 0]
if __name__== "_main_":
    blocks = int(input("Enter the no. of blocks in the disk: "))
    disk = [0] * blocks
    dirc = []
    ind = 0
    while 1:
        choice = int(input(
            "\n1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...\nEnter the Operation number you want to perform:"))
        if choice == 1:
            file = input("Enter the File name: ")
            noBlocks = int(input("Enter the no. of blocks: "))
            if ind + noBlocks > blocks:
                if len(getFreeBlocks(disk)) >= noBlocks:
                    con = int(input(
                        "\nThere is a possibiblity of compaction.\nIf you to do compaction (1 - Yes, 0 - No): "))
                    if con == 1:
                        dirc2 = []
                        ind2 = 0
```

```
                k = 0
                for i in dirc:
                    dirc2.append([i[0], ind2 + 1, i[2]])
                    for j in range(ind2, ind2 + i[2]):
                        disk[j] = i[0]
                        ind2 = j
                    ind2 += 1
                for i in range(ind2, len(disk)):
                    disk[i] = 0
                dirc = dirc2
                ind = ind2
            else:
                print(
                    "Can't insert file {}!. Since, file blocks > free blocks in disk".format(file))
                continue
        dirc.append([file, ind + 1, noBlocks])
        for i in range(ind, ind + noBlocks):
            disk[i] = file
            ind = i
        ind += 1
        print("Disk: ", disk, dirc)
    elif choice == 2:
        fileName = input("Enter the file Name: ")
        if fileName not in disk:
            print(f"File {fileName} not found in the disk")
            continue
```

```
        else:
            while fileName in disk:
                disk[disk.index(fileName)] = 0
        files = [i[0] for i in dirc]
        dirc.pop(files.index(fileName))
        print(disk, dirc)
    elif choice == 3:
        print("\nDirectory: \n ---------")
        print("Name\tStart\tNo. of Blocks\n")
        for i in range(len(dirc)):
            print("\t".join([str(i) for i in dirc[i]]))
    elif choice == 4:
        freeBlocks = getFreeBlocks(disk)
        if freeBlocks == 0:
            print("\nThere is no free block in disk!. Disk is full")
        else:
            print("\nFree Blocks in the disk: \n---------------------- ")
            print(" ".join(getFreeBlocks(disk)))
    else:
        break
```

OUTPUT:

```
Enter the Operation number you want to perform:3

Directory:
----------
Name    Start   No. of Blocks

f1      1       3
f3      8       4

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:4

Free Blocks in the disk:
------------------------
4  5  6  7  12  13  14

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:1
Enter the File name: f4
Enter the no. of blocks: 5

There is a possibiblity of compaction.
If you to do compaction (1 - Yes, 0 - No): 1
Disk:  ['f1', 'f1', 'f1', 'f3', 'f3', 'f3', 'f3', 'f4', 'f4', 'f4', 'f4', 'f4', 0, 0] [['f1', 1, 3], ['f3', 4, 4], ['f4', 8, 5]]

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:1
Enter the File name: f5
Enter the no. of blocks: 4
Can't insert file f5!. Since, file blocks > free blocks in disk

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:3

Directory:
----------
Name    Start   No. of Blocks

f1      1       3
f3      4       4
f4      8       5

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:4

Free Blocks in the disk:
------------------------
13  14

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:f4

PS D:\BE SEM 5\Operating System(OS)> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe "d:/BE SEM 5/Operating System(OS)
/Programs/File_Allocation/Contiguous_Disk_Allocation_Algo.py"
Enter the no. of blocks in the disk: 14

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:1
Enter the File name: f1
Enter the no. of blocks: 3
Disk:  ['f1', 'f1', 'f1', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [['f1', 1, 3]]

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:1
Enter the File name: f2
Enter the no. of blocks: 4
Disk:  ['f1', 'f1', 'f1', 'f2', 'f2', 'f2', 'f2', 0, 0, 0, 0, 0, 0, 0] [['f1', 1, 3], ['f2', 4, 4]]

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:1
Enter the File name: f3
Enter the no. of blocks: 4
Disk:  ['f1', 'f1', 'f1', 'f2', 'f2', 'f2', 'f2', 'f3', 'f3', 'f3', 'f3', 0, 0, 0] [['f1', 1, 3], ['f2', 4, 4], ['f3', 8, 4]]

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:3

Directory:
----------
Name    Start   No. of Blocks

f1      1       3
f2      4       4
f3      8       4

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:4

Free Blocks in the disk:
------------------------
12  13  14

1 - Insertion, 2 - Deletion, 3 - Display Directory, 4 - Display free blocks, Any key to Exit...
Enter the Operation number you want to perform:2
Enter the file Name: f2
['f1', 'f1', 'f1', 0, 0, 0, 0, 'f3', 'f3', 'f3', 'f3', 0, 0, 0] [['f1', 1, 3], ['f3', 8, 4]]
```

2) AIM: Implementation of Linked File Allocation method.

PROGRAM:

```
class Node:
    def __init_(self, data):
        self.data = data
        self.next = None
class LinkedList:
    def __init_(self):
        self.head = None
    def create(self, ele):
        new = Node(ele)
        if self.head == None:
            self.head = new
        else:
            temp = self.head
            while temp.next:
                temp = temp.next
            temp.next = new
    def getElements(self):
        eles = []
        temp = self.head
        while temp:
            eles.append(temp.data)
            temp = temp.next
        strg = "--> ".join([str(i) for i in eles])
        return strg + " " * (24 - len(strg))
```

```
def getFreeBlocks(dsk):
    return [i for i in range(len(dsk)) if dsk[i] == 0]
if __name__== "_main_":
    noBlocks = int(input("Enter no. of blocks in the disk: "))
    disk = [0 for i in range(noBlocks)]
    dirc = []
    while 1:
        choice = input("\n1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...\nEnter the choice: ")
        if choice == '1':
            print("\nCreating a File")
            fName = input("Enter the file name: ")
            nBlocks = int(input("Enter no. of blocks: "))
            freeBlocks = getFreeBlocks(disk)
            allocBlocks = []
            if nBlocks <= len(freeBlocks):
                ldlst = LinkedList()
                for i in range(nBlocks):
                    freeBlock = getFreeBlocks(disk)[0]
                    ldlst.create(freeBlock)
                    allocBlocks.append(freeBlock + 1)
                    disk[freeBlock] = fName
                dirc.append([fName, ldlst, allocBlocks])
            else:
                print("No free available blocks to allocate! i.e., free blocks in disk < blocks of file")
            print(disk, dirc)
```

```
    elif choice == '2':
        print("\nDeleting a File")
        fileName = input("Enter the file Name: ")
        fileIdx = [i[0] for i in dirc].index(fileName)
        for i in dirc[fileIdx][2]:
            disk[i - 1] = 0
        head = dirc[fileIdx][1]
        head = None
        dirc.pop(fileIdx)
        print(disk, dirc)
    elif choice == '3':
        print("\nExtending a File")
        fileName = input("Enter the file name which you want to Extend: ")
        nBlocks = int(input("Enter the no. of new blocks: "))
        fileIdx = [i[0] for i in dirc].index(fileName)
        freeBlocks = getFreeBlocks(disk)
        print(nBlocks, freeBlocks, len(freeBlocks), (nBlocks <= len(freeBlocks)))
        if nBlocks <= len(freeBlocks):
            ldlst = dirc[fileIdx][1]
            for i in range(nBlocks):
                freeBlock = getFreeBlocks(disk)[0]
                ldlst.create(freeBlock)
                dirc[fileIdx][2].append(freeBlock + 1)
                disk[freeBlock] = fileName
        else:
```

```
            print(f"No free available blocks to extend file {fileName}! i.e., free blocks in disk < blocks of file")
        print(disk, dirc)
    elif choice == '4':
        print("\nFile Directory\n ---------- ")
        print("File Name\tBlocks Numbers Allocated\tlength")
        for i in range(len(dirc)):
            print("{}\t\t{}\t{}".format(dirc[i][0], dirc[i][1].getElements(),len(dirc[i][2])))
    else:
        break
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe "d:/BE SEM 5/Operating System(OS)/Programs/File_Allocation/LinkedFileAllocation_Algo.py"
Enter no. of blocks in the disk: 17

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f1
Enter no. of blocks: 4
['f1', 'f1', 'f1', 'f1', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [['f1', <__main__.LinkedList object at 0x000002792F4FBFD0>, [1, 2, 3, 4]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f2
Enter no. of blocks: 3
['f1', 'f1', 'f1', 'f1', 'f2', 'f2', 'f2', 0, 0, 0, 0, 0, 0, 0, 0, 0] [['f1', <__main__.LinkedList object at 0x000002792F4FBFD0>, [1, 2, 3, 4]], ['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f3
Enter no. of blocks: 5
['f1', 'f1', 'f1', 'f1', 'f2', 'f2', 'f2', 'f3', 'f3', 'f3', 'f3', 'f3', 0, 0, 0, 0] [['f1', <__main__.LinkedList object at 0x000002792F4FBFD0>, [1, 2, 3, 4]], ['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f3', <__main__.LinkedList object at 0x000002792F4FBAC0>, [8, 9, 10, 11, 12]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f4
Enter no. of blocks: 3
['f1', 'f1', 'f1', 'f1', 'f2', 'f2', 'f2', 'f3', 'f3', 'f3', 'f3', 'f3', 'f4', 'f4', 'f4', 0, 0] [['f1', <__main__.LinkedList object at 0x000002792F4FBFD0>, [1, 2, 3, 4]], ['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f3', <__main__.LinkedList object at 0x000002792F4FBAC0>, [8, 9, 10, 11, 12]], ['f4', <__main__.LinkedList object at 0x000002792F4FB880>, [13, 14, 15]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 4

File Directory
---------------
File Name       Blocks Numbers Allocated        length
f1              0--> 1--> 2--> 3                4
f2              4--> 5--> 6                     3
f3              7--> 8--> 9--> 10--> 11         5
f4              12--> 13--> 14                  3
```

```
1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 2

Deleting a File
Enter the file Name: f1
[0, 0, 0, 0, 'f2', 'f2', 'f2', 'f3', 'f3', 'f3', 'f3', 'f3', 'f4', 'f4', 'f4', 0, 0] [['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f3', <__main__.LinkedList object at 0x000002792F4FBAC0>, [8, 9, 10, 11, 12]], ['f4', <__main__.LinkedList object at 0x000002792F4FB880>, [13, 14, 15]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 2

Deleting a File
Enter the file Name: f3
[0, 0, 0, 0, 'f2', 'f2', 'f2', 0, 0, 0, 0, 'f4', 'f4', 'f4', 0, 0] [['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f4', <__main__.LinkedList object at 0x000002792F4FB880>, [13, 14, 15]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 3

Extending a File
Enter the file name which you want to Extend: f4
Enter the no. of new blocks: 5
5 [0, 1, 2, 3, 7, 8, 9, 10, 11, 15, 16] 11 True
['f4', 'f4', 'f4', 'f4', 'f2', 'f2', 'f2', 'f4', 0, 0, 0, 0, 'f4', 'f4', 'f4', 0, 0] [['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f4', <__main__.LinkedList object at 0x000002792F4FB880>, [13, 14, 15, 1, 2, 3, 4, 8]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f6
Enter no. of blocks: 5
['f4', 'f4', 'f4', 'f4', 'f2', 'f2', 'f2', 'f4', 'f6', 'f6', 'f6', 'f6', 'f4', 'f4', 'f4', 'f6', 0] [['f2', <__main__.LinkedList object at 0x000002792F4FBC40>, [5, 6, 7]], ['f4', <__main__.LinkedList object at 0x000002792F4FB880>, [13, 14, 15, 1, 2, 3, 4, 8]], ['f6', <__main__.LinkedList object at 0x000002792F4FB8E0>, [9, 10, 11, 12, 16]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 4

File Directory
---------------
File Name       Blocks Numbers Allocated        length
f2              4--> 5--> 6                     3
f4              12--> 13--> 14--> 0--> 1--> 2--> 3--> 7 8
f6              8--> 9--> 10--> 11--> 15        5

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: exit
PS D:\BE SEM 5\Operating System(OS)>
```

3) AIM: Implementation of Indexed File Allocation method.

PROGRAM:

```
def getFreeBlocks(dsk):
    return [i for i in range(len(dsk)) if dsk[i] == 0]
if __name__== "_main_":
    noBlocks = int(input("Enter no. of blocks in the disk: "))
    disk = [0 for i in range(noBlocks)]
    dirc = []
    while 1:
        choice = input("\n1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...\nEnter the choice: ")
        if choice == '1':
            print("\nCreating a File")
            fileName = input("Enter the file name: ")
            nBlocks = int(input("Enter no. of blocks: "))
            freeBlocks = getFreeBlocks(disk)
            allocBlocks = []
            if nBlocks <= len(freeBlocks):
                fileIndex = freeBlocks[0]
                disk[fileIndex] = fileName + "_index"
                for i in range(nBlocks):
                    freeBlock = getFreeBlocks(disk)[0]
                    allocBlocks.append(freeBlock + 1)
                    disk[freeBlock] = fileName
                dirc.append([fileName, fileIndex + 1, allocBlocks])
```

```
        else:
            print("No free available blocks to allocate! i.e., free blocks in disk < blocks of file")
        print(disk, dirc)
    elif choice == '2':
        print("\nDeleting a File")
        fileName = input("Enter the file Name: ")
        fileIdx = [i[0] for i in dirc].index(fileName)
        for i in dirc[fileIdx][2]:
            disk[i - 1] = 0
        disk[dirc[fileIdx][1] - 1] = 0
        dirc.pop(fileIdx)
        print(disk, dirc)
    elif choice == '3':
        print("\nExtending a File")
        fileName = input("Enter the file name which you want to Extend: ")
        nBlocks = int(input("Enter the no. of new blocks: "))
        fileIdx = [i[0] for i in dirc].index(fileName)
        freeBlocks = getFreeBlocks(disk)
        if nBlocks <= len(freeBlocks):
            fileindex = dirc[fileIdx][1] - 1
            for i in range(nBlocks):
                freeBlock = getFreeBlocks(disk)[0]
                dirc[fileIdx][2].append(freeBlock + 1)
                disk[freeBlock] = fileName
        else:
```

```
            print(f"No free available blocks to extend file {fileName}! i.e., free blocks in disk < blocks of file")
        print(disk, dirc)
    elif choice == '4':
        print("\nFile Directory\n ---------- ")
        print("File Name\tIndex Block Number\tBlocks in Index(File)")
        for i in range(len(dirc)):
            print("{}\t\t{}{}\t{}".format(dirc[i][0], dirc[i][1], " " * 18, dirc[i][2]))
    else:
        break
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python310/python.exe "d:/BE SEM 5/Operating System(OS)/Programs/File_Allocation/IndexedFileAllocation_Algo.py"
Enter no. of blocks in the disk: 14

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f1
Enter no. of blocks: 3
['f1_index', 'f1', 'f1', 'f1', 0, 0, 0, 0, 0, 0, 0, 0, 0, 0] [['f1', 1, [2, 3, 4]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f2
Enter no. of blocks: 4
['f1_index', 'f1', 'f1', 'f1', 'f2_index', 'f2', 'f2', 'f2', 'f2', 0, 0, 0, 0, 0] [['f1', 1, [2, 3, 4]], ['f2', 5, [6, 7, 8, 9]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 1

Creating a File
Enter the file name: f3
Enter no. of blocks: 3
['f1_index', 'f1', 'f1', 'f1', 'f2_index', 'f2', 'f2', 'f2', 'f2', 'f3_index', 'f3', 'f3', 'f3', 0] [['f1', 1, [2, 3, 4]], ['f2', 5, [6, 7, 8, 9]], ['f3', 10, [11, 12, 13]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 4

File Directory
---------------
File Name       Index Block Number      Blocks in Index(File)
f1              1                       [2, 3, 4]
f2              5                       [6, 7, 8, 9]
f3              10                      [11, 12, 13]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 2

Deleting a File
Enter the file Name: f2
['f1_index', 'f1', 'f1', 'f1', 0, 0, 0, 0, 0, 'f3_index', 'f3', 'f3', 'f3', 0] [['f1', 1, [2, 3, 4]], ['f3', 10, [11, 12, 13]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 3

Extending a File
Enter the file name which you want to Extend: f3
Enter the no. of new blocks: 4
1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 3

Extending a File
Enter the file name which you want to Extend: f3
Enter the no. of new blocks: 4
['f1_index', 'f1', 'f1', 'f1', 'f3', 'f3', 'f3', 'f3', 0, 'f3_index', 'f3', 'f3', 'f3', 0] [['f1', 1, [2, 3, 4]], ['f3', 10, [11, 12, 13, 5, 6, 7, 8]]]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: 4

File Directory
---------------
File Name       Index Block Number      Blocks in Index(File)
f1              1                       [2, 3, 4]
f3              10                      [11, 12, 13, 5, 6, 7, 8]

1 - Creation, 2 - Deletion, 3 - Extension, 4 - Display Directory or press any key to Exit...
Enter the choice: exit
PS D:\BE SEM 5\Operating System(OS)>
```

Week – 10

Disk Scheduling Alogrithms:

1) AIM: Implementation of FCFS(First Come First Serve) Disk Scheduling Algorithm.

PROGRAM:

```
print("Enter the Queue:")
a = list(map(int, input().split()))
head = int(input("Enter the starting point:"))
q = []
q.append(head)
for i in a:
    q.append(i)
sum = 0
for i in range(len(q)-1):
    sum += abs(q[i+1]-q[i])
print("Total head movement:", sum)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs>
ograms/Disk_Scheduling/FCFS_Algo.py"
Enter the Queue:
98 183 37 122 14 24 65 67
Enter the starting point:53
Total head movement: 522
```

2) AIM: Implementation of SSTF(Shortest Seek Time First) Disk Scheduling Algorithm.

PROGRAM:

```
print("Enter the Queue:")
a = list(map(int, input().split()))
head = int(input("Enter the starting point:"))
```

```
q = []
q.append(head)
for i in a:
    q.append(i)
sum = 0
key = head
c = []
while len(c) < len(q):
    l = [0]
    m = 9999999999
    c.append(key)
    for j in range(len(q)):
        if q[j] not in c:
            m = min(m, abs(key-q[j]))
            if m == abs(key-q[j]):
                k = q[j]
    if m != 9999999999:
        sum += m
        key = k
print("Total head movement:", sum)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs>
ograms/Disk_Scheduling/SSTF_Algo.py"
Enter the Queue:
98 183 37 122 14 24 65 67
Enter the starting point:53
Total head movement: 236
```

3) AIM: Implementation of SCAN Disk Scheduling Algorithm.

PROGRAM:

```
print("Enter the Queue:")
q = list(map(int, input().split()))
head = int(input("Enter the starting point:"))
sum = 0
l1 = []
l2 = []
for i in q:
    if head > i:
        l1.append(i)
    else:
        l2.append(i)
l1.append(0)
l1.append(head)
l1.sort()
l1 = l1[::-1]
l2.sort()
l3 = l1+l2
for i in range(len(l3)-1):
    sum += abs(l3[i+1]-l3[i])
print("Total head movement:", sum)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs>
ograms/Disk_Scheduling/SCAN_Algo.py"
Enter the Queue:
98 183 37 122 14 24 65 67
Enter the starting point:53
[53, 37, 24, 14, 0, 65, 67, 98, 122, 183]
Total head movement: 236
```

4) AIM: Implementation of C-SCAN Disk Scheduling Algorithm.

PROGRAM:

```
print("Enter the Queue:")
q = list(map(int, input().split()))
head = int(input("Enter the starting point:"))
sum = 0
l1 = []
l2 = []
for i in q:
    if head > i:
        l1.append(i)
    else:
        l2.append(i)
l1.append(0)
l1.sort()
l2.append(head)
l2.append(199)
l2.sort()
l3 = l2+l1
print(l3)
for i in range(len(l3)-1):
    sum += abs(l3[i+1]-l3[i])
print("Total head movement:", sum)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs> &
ograms/Disk_Scheduling/C-SCAN_Algo.py"
Enter the Queue:
98 183 37 122 14 24 65 67
Enter the starting point:53
[53, 65, 67, 98, 122, 183, 199, 0, 14, 24, 37]
Total head movement: 382
```

5) AIM: Implementation of C-LOOK Disk Scheduling Algorithm.

PROGRAM:

```
print("Enter the Queue:")
q = list(map(int, input().split()))
head = int(input("Enter the starting point:"))
sum = 0
l1 = []
l2 = []
for i in q:
    if head > i:
        l1.append(i)
    else:
        l2.append(i)
l1.sort()
l2.append(head)
l2.sort()
l3 = l2+l1
print(l3)
for i in range(len(l3)-1):
    sum += abs(l3[i+1]-l3[i])
print("Total head movement:", sum)
```

OUTPUT:

```
PS D:\BE SEM 5\Operating System(OS)\Programs>
ograms/Disk_Scheduling/C-LOOK_Algo.py"
Enter the Queue:
98 183 37 122 14 24 65 67
Enter the starting point:53
[53, 65, 67, 98, 122, 183, 14, 24, 37]
Total head movement: 322
```