OS Question Bank Solution

1. Write a program to implement FCFS (with arrival time=0 for all) Calculate waiting time, turnaround time for each process. Calculate avg. waiting time, avg turnaround time

FCFS in C:-

```
#include <stdio.h>

// Function to find the waiting time for all processes

int waitingtime(int proc[], int n,

int burst_time[], int wait_time[]) {

   // waiting time for first process is 0

   wait_time[0] = 0;

   // calculating waiting time

   for (int i = 1; i < n ; i++ )

   wait_time[i] = burst_time[i-1] + wait_time[i-1] ;

   return 0;

}

// Function to calculate turn around time

int turnaroundtime( int proc[], int n,

int burst_time[], int wait_time[], int tat[]) {

   // calculating turnaround time by adding

   // burst_time[i] + wait_time[i]8

   int i;

   for ( i = 0; i < n ; i++)

   tat[i] = burst_time[i] + wait_time[i];

   return 0;
```

```
}
//Function to calculate average time
int avgtime( int proc[], int n, int burst_time[]) {
   int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
   int i;
   //Function to find waiting time of all processes
   waitingtime(proc, n, burst_time, wait_time);
   //Function to find turn around time for all processes
   turnaroundtime(proc, n, burst_time, wait_time, tat);
   //Display processes along with all details
   printf("Processes  Burst   Waiting Turn around \n");
   // Calculate total waiting time and total turn
   // around time
   for ( i=0; i<n; i++) {
      total_wt = total_wt + wait_time[i];
      total_tat = total_tat + tat[i];
      printf(" %d\t   %d\t\t %d  \t%d\n", i+1, burst_time[i],
wait_time[i], tat[i]);
   }
   printf("Average  waiting  time  =  %f\n",  (float)total_wt  /
(float)n);
   printf("Average turn around time = %f\n", (float)total_tat /
(float)n);
   return 0;
}
```

```
// main function

int main() {

   //process id's

   int proc[] = { 1, 2, 3};

   int n = sizeof proc / sizeof proc[0];

   //Burst time of all processes

   int burst_time[] = {5, 8, 12};

   avgtime(proc, n, burst_time);

   return 0;

}
```

FCFS in Python:-

```
print("FIRST COME FIRST SERVE SCHEDULING")
n= int(input("Enter number of processes : "))
d = dict()

for i in range(n):
   key = "P"+str(i+1)
   a = int(input("Enter arrival time of process"+str(i+1)+": "))
   b = int(input("Enter burst time of process"+str(i+1)+": "))
   l = []
   l.append(a)
   l.append(b)
   d[key] = l

d = sorted(d.items(), key=lambda item: item[1][0])

ET = []
```

```
for i in range(len(d)):
    # first process
    if(i==0):
        ET.append(d[i][1][1])

    # get prevET + newBT
    else:
        ET.append(ET[i-1] + d[i][1][1])

TAT = []
for i in range(len(d)):
    TAT.append(ET[i] - d[i][1][0])

WT = []
for i in range(len(d)):
    WT.append(TAT[i] - d[i][1][1])

avg_WT = 0
for i in WT:
    avg_WT +=i
avg_WT = (avg_WT/n)

avg_TAT = 0
for i in TAT:
    avg_TAT += i
avg_TAT = (avg_TAT/n)

print("Process | Arrival | Burst | Exit | Turn Around | Wait |")
for i in range(n):
```

print(" ",d[i][0]," | ",d[i][1][0]," | ",d[i][1][1]," | ",ET[i]," | ",TAT[i]," | ",WT[i]," | ")

print("Average Waiting Time: ",avg_WT)

print("Average Turn Around Time: ",avg_TAT)

2. Write a program to implement SJF (with arrival time=0 for all) Calculate waiting time, turnaround time for each process. Calculate avg. waiting time, avg turnaround time

SJF in C:-

```c
#include<stdio.h>

int main()

{

   int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,pos,temp;

   float avg_wt,avg_tat;

   printf("Enter number of process:");

   scanf("%d",&n);

    printf("\nEnter Burst Time:\n");

   for(i=0;i<n;i++)

   {

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

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

       p[i]=i+1;

   }

   //sorting of burst times

   for(i=0;i<n;i++)

   {
```

```
    pos=i;
    for(j=i+1;j<n;j++)
    {
        if(bt[j]<bt[pos])
            pos=j;
    }
     temp=bt[i];
    bt[i]=bt[pos];
    bt[pos]=temp;
     temp=p[i];
    p[i]=p[pos];
    p[pos]=temp;
}

wt[0]=0;

for(i=1;i<n;i++)
{
    wt[i]=0;
    for(j=0;j<i;j++)
        wt[i]+=bt[j];
     total+=wt[i];
}
 avg_wt=(float)total/n;
```

```
    total=0;

     printf("\nProcesst    Burst Time    \tWaiting Time\tTurnaround Time");

    for(i=0;i<n;i++)

    {

        tat[i]=bt[i]+wt[i];

        total+=tat[i];

        printf("\np%d\t\t  %d\t\t    %d\t\t\t\t%d",p[i],bt[i],wt[i],tat[i]);

    }

     avg_tat=(float)total/n;

    printf("\n\nAverage Waiting Time=%f",avg_wt);

    printf("\nAverage Turnaround Time=%f\n",avg_tat);

}
```

SJF in Python:-

```
    # Function to find the waiting time
# for all processes
def findWaitingTime(processes, n, wt):
        rt = [0] * n


        # Copy the burst time into rt[]
        for i in range(n):
                rt[i] = processes[i][1]
        complete = 0
        t = 0
```

```
minm = 999999999
short = 0
check = False

# Process until all processes gets
# completed
while (complete != n):

    # Find process with minimum remaining
    # time among the processes that
    # arrives till the current time`
    for j in range(n):
        if ((processes[j][2] <= t) and
            (rt[j] < minm) and rt[j] > 0):
            minm = rt[j]
            short = j
            check = True
    if (check == False):
        t += 1
        continue

    # Reduce remaining time by one
    rt[short] -= 1
```

```
# Update minimum
minm = rt[short]
if (minm == 0):
        minm = 999999999

# If a process gets completely
# executed
if (rt[short] == 0):

        # Increment complete
        complete += 1
        check = False

        # Find finish time of current
        # process
        fint = t + 1

        # Calculate waiting time
        wt[short] = (fint - proc[short][1] - proc[short][2])

        if (wt[short] < 0):
                wt[short] = 0
```

```
            # Increment time
            t += 1

# Function to calculate turn around time
def findTurnAroundTime(processes, n, wt, tat):

    # Calculating turnaround time
    for i in range(n):
        tat[i] = processes[i][1] + wt[i]

# Function to calculate average waiting
# and turn-around times.
def findavgTime(processes, n):
    wt = [0] * n
    tat = [0] * n

    # Function to find waiting time
    # of all processes
    findWaitingTime(processes, n, wt)

    # Function to find turn around time
    # for all processes
```

```
    findTurnAroundTime(processes, n, wt, tat)

    # Display processes along with all details
    print("Processes 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)

# Driver code
if __name__ =="__main__":

    # Process id's
    proc = [[1, 6, 1], [2, 8, 1],
```

[3, 7, 2], [4, 3, 3]]

n = 4

findavgTime(proc, n)

3. Write a program to implement Non preemptive Priority scheduling. Calculate waiting time, turnaround time for each process. Calculate avg. waiting time, avg. turnaround time

```
#include<iostream>

using namespace std;
int main()
{
    int a[10],b[10],x[10],pr[10]={0};
    int waiting[10],turnaround[10],completion[10];
    int i,j,smallest,count=0,time,n;
    double avg=0,tt=0,end;
   cout<<"\nEnter the number of Processes: ";
    cin>>n;
    for(i=0;i<n;i++)
    {
      cout<<"\nEnter arrival time of process: ";
      cin>>a[i];
    }
    for(i=0;i<n;i++)
```

```
{
  cout<<"\nEnter burst time of process: ";
  cin>>b[i];
}
for(i=0;i<n;i++)
{
  cout<<"\nEnter priority of process: ";
  cin>>pr[i];
}
for(i=0;i<n;i++)
    x[i]=b[i];

pr[9]=-1;
for(time=0;count!=n;time++)
{
    smallest=9;
    for(i=0;i<n;i++)
    {
        if(a[i]<=time && pr[i]>pr[smallest] && b[i]>0 )
            smallest=i;
    }
    time+=b[smallest]-1;
    b[smallest]=-1;
```

```
        count++;

        end=time+1;

        completion[smallest] = end;

        waiting[smallest] = end - a[smallest] - x[smallest];

        turnaround[smallest] = end - a[smallest];

    }

    cout<<"Process"<<"\t"<<        "burst-time"<<"\t"<<"arrival-time"
<<"\t"<<"waiting-time" <<"\t"<<"turnaround-time"<< "\t"<<"completion-
time"<<"\t"<<"Priority"<<endl;

    for(i=0;i<n;i++)

    {


cout<<"p"<<i+1<<"\t\t"<<x[i]<<"\t\t"<<a[i]<<"\t\t"<<waiting[i]<<"\t\t
"<<turnaround[i]<<"\t\t"<<completion[i]<<"\t\t"<<pr[i]<<endl;

        avg = avg + waiting[i];

        tt = tt + turnaround[i];

    }

    cout<<"\n\nAverage waiting time ="<<avg/n;

    cout<<"  Average Turnaround time ="<<tt/n<<endl;

}
```

Python:-

```
# Python3 implementation for Priority Scheduling with
# Different Arrival Time priority scheduling
"""1. sort the processes according to arrival time
2. if arrival time is same the acc to priority
3. apply fcfs """

totalprocess = int(input("Enter the number of processes: "))
#totalprocess = 5
proc = []
for i in range(totalprocess):
 l = []
 for j in range(totalprocess-1):
            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] * totalprocess
```

```
 # 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] * totalprocess
tat = [0] * totalprocess

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] * totalprocess
ctime = [0] * totalprocess
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)

# Driver code
if __name__ =="__main__":
 arrivaltime = []
```

```
bursttime = []

priority = []


for i in range(totalprocess):

        arrivaltime.append(0)

        bursttime.append(int(input("Enter the Burst time for P"+str(i)+": ")))

        priority.append(int(input("Enter the priority for P"+str(i)+": ")))


        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)


# finding Gantt Chart

findgc()
```

4. Write a program to implement Round Robin. Calculate waiting time, turnaround time for each process. Calculate avg. waiting time, avg turnaround time

A)

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

```
int main()

{

      int i, limit, total = 0, x, counter = 0, time_quantum;

      int  wait_time  =  0,  turnaround_time  =  0,  arrival_time[10],
burst_time[10], temp[10];

      float average_wait_time, average_turnaround_time;

      printf("\nEnter Total Number of Processes:\t");

      scanf("%d", &limit);

      x = limit;

      for(i = 0; i < limit; i++)

      {

            printf("\nEnter Details of Process[%d]\n", i + 1);



            printf("Arrival Time:\t");



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



            printf("Burst Time:\t");



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



           Up temp[i] = burst_time[i];

      }
```

```
      printf("nEnter Time Quantum:\t");

      scanf("%d", &time_quantum);

      printf("\nProcess ID\t\tBurst Time\t Turnaround Time\t Waiting
Time\n");

      for(total = 0, i = 0; x != 0;)

      {

            if(temp[i] <= time_quantum && temp[i] > 0)

            {

                  total = total + temp[i];

                  temp[i] = 0;

                  counter = 1;

            }

            else if(temp[i] > 0)

            {

                  temp[i] = temp[i] - time_quantum;

                  total = total + time_quantum;

            }

            if(temp[i] == 0 && counter == 1)

            {

                  x--;

                  printf("\nProcess[%d]\t\t%d\t\t %d\t\t\t %d", i + 1,
burst_time[i], total - arrival_time[i], total - arrival_time[i] -
burst_time[i]);

                  wait_time = wait_time + total - arrival_time[i] -
burst_time[i];
```

```
                turnaround_time = turnaround_time + total -
arrival_time[i];
                counter = 0;
            }
            if(i == limit - 1)
            {
                i = 0;
            }
            else if(arrival_time[i + 1] <= total)
            {
                i++;
            }
            else
            {
                i = 0;
            }
        }

        average_wait_time = wait_time * 1.0 / limit;
        average_turnaround_time = turnaround_time * 1.0 / limit;
        printf("\n\nAverage Waiting Time:\t%f", average_wait_time);
        printf("\nAvg Turnaround Time:\t%f\n", average_turnaround_time);
        return 0;
}
```

Python:-

```
# Function to find the waiting time
# for all processes
def findWaitingTime(processes, n, bt,wt, quantum):
    rem_bt = [0] * n

    # Copy the burst time into rt[]
    for i in range(n):
            rem_bt[i] = bt[i]
    t = 0 # Current time

    # Keep traversing processes in round
    # robin manner until all of them are
    # not done.
    while(1):
            done = True

            # Traverse all processes one by
            # one repeatedly
            for i in range(n):

                    # If burst time of a process is greater
                    # than 0 then only need to process further
```

```
if (rem_bt[i] > 0) :
        done = False # There is a pending process

        if (rem_bt[i] > quantum) :

                # Increase the value of t i.e. shows
                # how much time a process has been processed
                t += quantum

                # Decrease the burst_time of current
                # process by quantum
                rem_bt[i] -= quantum

        # If burst time is smaller than or equal
        # to quantum. Last cycle for this process
        else:

                # Increase the value of t i.e. shows
                # how much time a process has been processed
                t = t + rem_bt[i]

                # Waiting time is current time minus
                # time used by this process
```

```
                wt[i] = t - bt[i]

                # As the process gets fully executed
                # make its remaining burst time = 0
                rem_bt[i] = 0

        # If all processes are done
        if (done == True):
            break

# Function to calculate turn around time
def findTurnAroundTime(processes, n, bt, wt, tat):

    # Calculating turnaround time
    for i in range(n):
        tat[i] = bt[i] + wt[i]


# Function to calculate average waiting
# and turn-around times.
def findavgTime(processes, n, bt, quantum):
    wt = [0] * n
    tat = [0] * n
```

```
# Function to find waiting time
# of all processes
findWaitingTime(processes, n, bt,wt, quantum)

# Function to find turn around time
# for all processes
findTurnAroundTime(processes, n, bt,wt, tat)

# Display processes along with all details
print("Processes 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(" ", i + 1, "\t\t", bt[i],
                "\t\t", wt[i], "\t\t", tat[i])

print("\nAverage waiting time = %.5f "%(total_wt /n) )
print("Average turn around time = %.5f "% (total_tat / n))
```

```
# Driver code
if __name__ =="__main__":

    # Process id's
    proc = []
    burst_time = []
    n = int(input("Enter number of processes: "))

    for i in range(n):
    # Burst time of all processes
            burst_time.append(int(input("Enter burst time: ")))
            proc.append(i)

    # Time quantum
    quantum = int(input("Enter time quantum for RR: "));
    findavgTime(proc, n, burst_time, quantum)
```

5. Write a program to simulate the First Fit Memory Allocation Technique.

A) First Fit in C++

```
#include<bits/stdc++.h>

using namespace std;

// Function to allocate memory to

// blocks as per First fit algorithm
```

```
void First_Fit(int block_size[], int total_blocks, int process_size[], int
total_process) {

   int allocation[total_process];

   memset(allocation, -1, sizeof(allocation));

   //this for loop wll pick eact process and allocate a first fit block to
it

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

      for (int j = 0; j < total_blocks; j++) {

         if (block_size[j] >= process_size[i]) {

            allocation[i] = j;

            block_size[j] -= process_size[i];

            break;

         }

      }

   }8

   cout << "\nProcess No.\tProcess Size\tBlock no.\n";

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

      cout << " " << i+1 << "\t\t" << process_size[i] << "\t\t";

      if (allocation[i] != -1)

         cout << allocation[i] + 1;

      else

         cout << "Not Allocated";

         cout << endl;

   }
```

```
}

int main() {

   //create array to store block sizes

   int block_size[] = {300, 50, 200, 350, 70};

    //create array to store process sizes

   int process_size[] = {200, 47, 212, 426, 10};

    //variable total_blocks that contain total number of blocks

   int total_blocks = sizeof(block_size) / sizeof(block_size[0]);

    //variable total_process that contain total number of blocks

   int total_process = sizeof(process_size) / sizeof(process_size[0]);

    //calling the function First_fit

   First_Fit(block_size, total_blocks, process_size, total_process);

   return 0 ;

}
```

First Fit in C:-

#include<stdio.h>

Int main()

{

int bsize[10], psize[10], bno, pno, flags[10], allocation[10], i, j;

for(i = 0; i < 10; i++)

{

```
        flags[i] = 0;
        allocation[i] = -1
}
printf("Enter no. of blocks: ");
scanf("%d", &bno);
printf("\nEnter size of each block: ");
for(i = 0; i < bno; i++)
        scanf("%d", &bsize[i]);
printf("\nEnter no. of processes: ");
scanf("%d", &pno);
printf("\nEnter size of each process: ");
for(i = 0; i < pno; i++)
        scanf("%d", &psize[i]);
for(i = 0; i < pno; i++)        //allocation as per first fit
        for(j = 0; j < bno; j++)
                if(flags[j] == 0 && bsize[j] >= psize[i])
                {
                        allocation[j] = i;
                        flags[j] = 1;
                        break;
                }
//display allocation details
printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");
```

```
    for(i = 0; i < bno; i++)
    {
            printf("\n%d\t\t%d\t\t", i+1, bsize[i]);
            if(flags[i] == 1)
                    printf("%d\t\t\t%d",allocation[i]+1,psize[allocation[i]]);
            else
                    printf("Not allocated");
    }
Return 0;
}
```

First Fit in Python:-

```
# Function to allocate memory to
# blocks as per First fit algorithm
def firstFit(blockSize, m, processSize, n):

    # Stores block id of the
    # block allocated to a process
    allocation = [-1] * n

    # Initially no block is assigned to any process

    # pick each process and find suitable blocks
    # according to its size ad assign to it
```

```
    for i in range(n):
        for j in range(m):
            if blockSize[j] >= processSize[i]:

                # allocate block j to p[i] process
                allocation[i] = j

                # Reduce available memory in this block.
                blockSize[j] -= processSize[i]

                break

    print(" Process No. Process Size      Block no.")
    for i in range(n):
        print(" ", i + 1, "                 ", processSize[i],
                                            "                    ", end = " ")
        if allocation[i] != -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")

# Driver code
if __name__ == '__main__':
```

```
m=int(input("Enter number of Blocks: "))
n=int(input("Enter number of Processes: "))
blockSize = []
processSize = []

for i in range (m):
        blockSize.append(int(input("Enter the size of Block B"+str(i)+": ")))

for j in range (n):
        processSize.append(int(input("Enter the size of Process P"+str(j)+": ")))

firstFit(blockSize, m, processSize, n)
```

6. Write programs to simulate the Best Fit Memory Allocation Technique.

Best Fit in C:-

```c
#include<stdio.h>

int main()

{

   int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;

   static int barray[20],parray[20];

   printf("Enter the number of blocks:-\n");

   scanf("%d",&nb);

   printf("Enter the number of processes:-\n");

   scanf("%d",&np);

   printf("\nEnter the size of the blocks:-\n");
```

```
for(i=1;i<=nb;i++)
{
    printf("Block no.%d:",i);
    scanf("%d",&b[i]);
}
printf("\nEnter the size of the processes :-\n");
for(i=1;i<=np;i++)
{
    printf("Process no.%d:",i);
    scanf("%d",&p[i]);
}
for(i=1;i<=np;i++)
{
    for(j=1;j<=nb;j++)
    {
        if(barray[j]!=1)
        {
            temp=b[j]-p[i];
            if(temp>=0)
                if(lowest>temp)
                {
                    parray[i]=j;
                    lowest=temp;
                }
```

```
            }

        }

        fragment[i]=lowest;

        barray[parray[i]]=1;

        lowest=10000;

    }

    printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment\n");

    for(i=1;i<=np && parray[i]!=0;i++)


printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragment[i]);



    printf("\n");

}
```

Best Fit in Python:-

```
# Python3 implementation of Best - Fit algorithm


# Function to allocate memory to blocks

# as per Best fit algorithm

def bestFit(blockSize, m, processSize, n):


    # Stores block id of the block

    # allocated to a process

    allocation = [-1] * n
```

```
# pick each process and find suitable
# blocks according to its size ad
# assign to it
for i in range(n):

    # Find the best fit block for
    # current process
    bestIdx = -1
    for j in range(m):
        if blockSize[j] >= processSize[i]:
            if bestIdx == -1:
                bestIdx = j
            elif blockSize[bestIdx] > blockSize[j]:
                bestIdx = j

    # If we could find a block for
    # current process
    if bestIdx != -1:

        # allocate block j to p[i] process
        allocation[i] = bestIdx
```

```
                # Reduce available memory in this block.
                blockSize[bestIdx] -= processSize[i]

    print("Process No. Process Size       Block no.")
    for i in range(n):
        print(i + 1, "               ", processSize[i],
                                                    end = "                    ")
        if allocation[i] != -1:
            print(allocation[i] + 1)
        else:
            print("Not Allocated")

# Driver code
if __name__ == '__main__':
    m=int(input("Enter number of Blocks: "))
    n=int(input("Enter number of Processes: "))
    blockSize = []
    processSize = []

    for i in range (m):
        blockSize.append(int(input("Enter the size of Block B"+str(i)+": ")))

    for j in range (n):
```

```
            processSize.append(int(input("Enter the size of Process P"+str(j)+": ")))


        bestFit(blockSize, m, processSize, n)
```

7. Write programs to simulate the Worst Fit Memory Allocation Technique.

Worst Fit in C:-

```
#include <stdio.h>

void implimentWorstFit(int blockSize[], int blocks, int processSize[], int processes)

{

   // This will store the block id of the allocated block to a process

   int allocation[processes];

   int occupied[blocks];

   // initially assigning -1 to all allocation indexes

   // means nothing is allocated currently

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

      allocation[i] = -1;

   }

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

      occupied[i] = 0;

   }

   // pick each process and find suitable blocks

   // according to its size an  d assign to it


for (int i=0; i < processes; i++)
```

```
{
    int indexPlaced = -1;
    for(int j = 0; j < blocks; j++)
    {
       // if not occupied and block size is large enough
       if(blockSize[j] >= processSize[i] && !occupied[j])
    {
       // place it at the first block fit to accomodate process
       if (indexPlaced == -1)
          indexPlaced = j;

       // if any future block is larger than the current block where
       // process is placed, change the block and thus indexPlaced
       else if (blockSize[indexPlaced] < blockSize[j])
          indexPlaced = j;
    }
  }
  // If we were successfully able to find block for the process
  if (indexPlaced != -1)
  {
    // allocate this block j to process p[i]
    allocation[i] = indexPlaced;
    // make the status of the block as occupied
```

```
            occupied[indexPlaced] = 1;
            // Reduce available memory for the block
            blockSize[indexPlaced] -= processSize[i];
        }
    }
    printf("\nProcess No.\tProcess Size\tBlock no.\n");
    for (int i = 0; i < processes; i++)
    {
        printf("%d \t\t\t %d \t\t\t", i+1, processSize[i]);
        if (allocation[i] != -1)
            printf("%d\n",allocation[i] + 1);
        else
            printf("Not Allocated\n");
    }
}
// Driver code
int main()
{
    int blockSize[] = {100, 50, 30, 120, 35};
    int processSize[] = {40, 10, 30, 60};
    int blocks = sizeof(blockSize)/sizeof(blockSize[0]);
    int processes = sizeof(processSize)/sizeof(processSize[0]);
```

```
    implimentWorstFit(blockSize, blocks, processSize, processes);
    return 0;
}
```

Worst Fit in Python:-

```
# Python3 implementation of worst - Fit algorithm


# Function to allocate memory to blocks as
# per worst fit algorithm
def worstFit(blockSize, m, processSize, n):

        # Stores block id of the block
        # allocated to a process

        # Initially no block is assigned
        # to any process
        allocation = [-1] * n

        # pick each process and find suitable blocks
        # according to its size ad assign to it
        for i in range(n):

                # Find the best fit block for
                # current process
```

```
            wstIdx = -1
            for j in range(m):
                if blockSize[j] >= processSize[i]:
                    if wstIdx == -1:
                        wstIdx = j
                    elif blockSize[wstIdx] < blockSize[j]:
                        wstIdx = j

            # If we could find a block for
            # current process
            if wstIdx != -1:

                # allocate block j to p[i] process
                allocation[i] = wstIdx

                # Reduce available memory in this block.
                blockSize[wstIdx] -= processSize[i]

print("Process No. Process Size Block no.")
for i in range(n):
            print(i + 1, "               ",
                processSize[i], end = "           ")
            if allocation[i] != -1:
```

```
                print(allocation[i] + 1)
        else:
                print("Not Allocated")


# Driver code
if __name__ == '__main__':
    m=int(input("Enter number of Blocks: "))
    n=int(input("Enter number of Processes: "))
    blockSize = []
    processSize = []

    for i in range (m):
        blockSize.append(int(input("Enter the size of Block B"+str(i)+": ")))

    for j in range (n):
        processSize.append(int(input("Enter the size of Process P"+str(j)+": ")))

    worstFit(blockSize, m, processSize, n)
```

8. Write a program to implement FIFO policy and calculate Hit ratio and Miss ratio

FIFO in C:-

```
#include <stdio.h>
int main()
{
```

```
int referenceString[10], pageFaults = 0, m, n, s, pages, frames;
printf("\nEnter the number of Pages:\t");
scanf("%d", &pages);
printf("\nEnter reference string values:\n");
for( m = 0; m < pages; m++)
{
   printf("Value No. [%d]:\t", m + 1);
   scanf("%d", &referenceString[m]);
}
printf("\n What are the total number of frames:\t");
{
   scanf("%d", &frames);
}
int temp[frames];
for(m = 0; m < frames; m++)
{
  temp[m] = -1;
}
for(m = 0; m < pages; m++)
{
  s = 0;
  for(n = 0; n < frames; n++)
   {
```

```
    if(referenceString[m] == temp[n])
      {
        s++;
        pageFaults--;
      }
  }
  pageFaults++;
  if((pageFaults <= frames) && (s == 0))
    {
      temp[m] = referenceString[m];
    }
  else if(s == 0)
    {
      temp[(pageFaults - 1) % frames] = referenceString[m];
    }
    printf("\n");
    for(n = 0; n < frames; n++)
     {
       printf("%d\t", temp[n]);
     }
}
printf("\nTotal Page Faults:\t%d\n", pageFaults);
return 0;
```

}

```
#include<stdio.h>

int main()

{

int i,j,n,a[50],frame[10],no,k,avail,count=0;

            printf("\n ENTER THE NUMBER OF PAGES:\n");

scanf("%d",&n);

            printf("\n ENTER THE PAGE NUMBER :\n");

            for(i=1;i<=n;i++)

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

            printf("\n ENTER THE NUMBER OF FRAMES :");

            scanf("%d",&no);

for(i=0;i<no;i++)

            frame[i]= -1;

                        j=0;

                        printf("\tref string\t page frames\n");

for(i=1;i<=n;i++)

                        {

                                    printf("%d\t\t",a[i]);

                                    avail=0;

                                    for(k=0;k<no;k++)
```

```
if(frame[k]==a[i])
                                                    avail=1;
                                            if (avail==0)
                                            {
                                                    frame[j]=a[i];
                                                    j=(j+1)%no;
                                                    count++;
                                                    for(k=0;k<no;k++)

printf("%d\t",frame[k]);
}
                                            printf("\n");
}
                                printf("Page Fault Is %d",count);
                                return 0;
}
```

FIFO in Python:- (Imported Queue in Program)

#from queue import Queue

Function to find page faults using FIFO

#def pageFaults(incomingStream, n, frames):

```
#    print("Incoming \t pages")
#    # Using Hashset to quickly check if a given
#    # incoming stream item in set or not
#    s = set()

#    # Queue created to store pages in FIFO manner
#    # since set will not store order or entry
#    # we will use queue to note order of entry of incoming page
#    queue = Queue()

#    page_faults = 0
#    for i in range(n):

#        # if set has lesser item than frames
#        # i.e. set can hold more items
#        if len(s) < frames:

#            # If incoming item is not present, add to set
#            if incomingStream[i] not in s:
#                s.add(incomingStream[i])

#                # increment page fault
#                page_faults += 1
```

```
#             # Push the incoming page into the queue
#             queue.put(incomingStream[i])

#       # If the set is full then we need to do page replacement
#       # in FIFO manner that is remove first item from both
#       # set and queue then insert incoming page
#       else:

#           # If incoming item is not present
#           if incomingStream[i] not in s:
#               # remove the first page from the queue
#               val = queue.queue[0]

#               queue.get()

#               # Remove from set
#               s.remove(val)

#               # insert incoming page to set
#               s.add(incomingStream[i])

#               # push incoming page to queue
```

```
#                queue.put(incomingStream[i])


#                # Increment page faults
#                page_faults += 1


#        print(incomingStream[i], end="\t\t")
#        for q_item in queue.queue:
#            print(q_item, end="\t")


#        print()
#    return page_faults



## Driver code
#incomingStream = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1]
#n = len(incomingStream)
#frames = 3
#page_faults = pageFaults(incomingStream, n, frames)
#hits = n - page_faults

#print("\nPage Faults: " + str(page_faults))
#print("Hit: " + str(hits))
```

```
from queue import Queue



# Function to find page faults using FIFO
def pageFaults(incomingStream, n, frames):
    print("Incoming \t pages")
    # Using Hashset to quickly check if a given
    # incoming stream item in set or not
    s = set()

    # Queue created to store pages in FIFO manner
    # since set will not store order or entry
    # we will use queue to note order of entry of incoming page
    queue = Queue()

    page_faults = 0
    for i in range(n):

        # if set has lesser item than frames
        # i.e. set can hold more items
        if len(s) < frames:
```

```
    # If incoming item is not present, add to set
    if incomingStream[i] not in s:
        s.add(incomingStream[i])

        # increment page fault
        page_faults += 1

        # Push the incoming page into the queue
        queue.put(incomingStream[i])

# If the set is full then we need to do page replacement
# in FIFO manner that is remove first item from both
# set and queue then insert incoming page
else:

    # If incoming item is not present
    if incomingStream[i] not in s:
        # remove the first page from the queue
        val = queue.queue[0]

        queue.get()

        # Remove from set
```

```
            s.remove(val)

            # insert incoming page to set
            s.add(incomingStream[i])

            # push incoming page to queue
            queue.put(incomingStream[i])

            # Increment page faults
            page_faults += 1

        print(incomingStream[i], end="\t\t")
        for q_item in queue.queue:
            print(q_item, end="\t")

        print()
    return page_faults


# Driver code
incomingStream = []
n= int(input("Enter length of page sequence: "))
for i in range(n):
```

incomingStream.append(int(input()))

frames = 3

page_faults = pageFaults(incomingStream, n, frames)

hits = n - page_faults

print("\nPage Faults: " + str(page_faults))

print("Hit: " + str(hits))

print("Hit Ratio: " + str((hits/n)*100))

print("Fault Ratio: " + str((page_faults/n)*100))

9. Write a program to implement LRU policy and calculate Hit ratio and Miss ratio

LRU in C:-

```c
#include<stdio.h>

int findLRU(int time[], int n){

int i, minimum = time[0], pos = 0;

for(i = 1; i < n; ++i){

if(time[i] < minimum){

minimum = time[i];

pos = i;

}

}

return pos;

}

int main()

{
```

```
   int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10],
flag1, flag2, i, j, pos, faults = 0;

printf("Enter number of frames: ");

scanf("%d", &no_of_frames);

printf("Enter number of pages: ");

scanf("%d", &no_of_pages);

printf("Enter reference string: ");

   for(i = 0; i < no_of_pages; ++i){

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

   }



for(i = 0; i < no_of_frames; ++i){

    frames[i] = -1;

   }



   for(i = 0; i < no_of_pages; ++i){

    flag1 = flag2 = 0;



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

    if(frames[j] == pages[i]){

    counter++;

    time[j] = counter;

  flag1 = flag2 = 1;

  break;
```

```
  }

    }



    if(flag1 == 0){

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

    if(frames[j] == -1){

    counter++;

    faults++;

    frames[j] = pages[i];

    time[j] = counter;

    flag2 = 1;

    break;

    }

    }

    }



    if(flag2 == 0){

    pos = findLRU(time, no_of_frames);

    counter++;

    faults++;

    frames[pos] = pages[i];

    time[pos] = counter;

    }
```

```
    printf("\n");



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

    printf("%d\t", frames[j]);

    }

}

printf("\n\nTotal Page Faults = %d\n", faults);



   return 0;

}
```

LRU in Python:-

```
ref_string = list(map(int,input("Enter Reference String numbers :").split()))

frames = int(input("Enter number of frames :"))

page = 0

frame_status = []

count = 0

for i in ref_string:

    if i in frame_status:

        continue

    if len(frame_status) == frames:

        page+=1

        frame_status[count] = i
```

```
        count+=1
        if count == frames:
            count=0
    else:
        frame_status.append(i)
    print(f'Frame Status : {frame_status}')
hit=len(ref_string)- page-frames
miss = page+frames
print(f'Page Fault is {miss}')
print(f'Page Miss is {miss}')
print(f'Page Hit is {hit}')
print("Page Hit Ratio: "+str((hit/len(ref_string)*100)))
print("Page Miss Ratio: "+str((miss/len(ref_string)*100)))
```

10. Write a program to implement Optimal policy and calculate Hit ratio and Miss ratio

Optimal Replacement Policy in C:-

```
#include<stdio.h>
int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], temp[10], flag1, flag2, flag3, i, j, k, pos, max, faults = 0;
    printf("Enter number of frames: ");
    scanf("%d", &no_of_frames);
```

```
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);

printf("Enter page reference string: ");

for(i = 0; i < no_of_pages; ++i){
   scanf("%d", &pages[i]);
}

for(i = 0; i < no_of_frames; ++i){
   frames[i] = -1;
}

for(i = 0; i < no_of_pages; ++i){
   flag1 = flag2 = 0;

   for(j = 0; j < no_of_frames; ++j){
      if(frames[j] == pages[i]){
            flag1 = flag2 = 1;
            break;
         }
   }
```

```
if(flag1 == 0){
   for(j = 0; j < no_of_frames; ++j){
      if(frames[j] == -1){
         faults++;
         frames[j] = pages[i];
         flag2 = 1;
         break;
      }
   }
}

if(flag2 == 0){
 flag3 =0;

   for(j = 0; j < no_of_frames; ++j){
    temp[j] = -1;

    for(k = i + 1; k < no_of_pages; ++k){
    if(frames[j] == pages[k]){
    temp[j] = k;
    break;
    }
    }
```

```
        }

        for(j = 0; j < no_of_frames; ++j){
         if(temp[j] == -1){
         pos = j;
         flag3 = 1;
         break;
         }
        }

        if(flag3 ==0){
         max = temp[0];
         pos = 0;

         for(j = 1; j < no_of_frames; ++j){
         if(temp[j] > max){
         max = temp[j];
         pos = j;
         }
         }
        }
frames[pos] = pages[i];
faults++;
```

```
        }

        printf("\n");

        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }

    printf("\n\nTotal Page Faults = %d", faults);

    return 0;
}
```

11. Write a program to simulate MVT

MVT in C:-

```
#include<stdio.h>
#include<conio.h>
void main()
{
int m=0,m1=0,m2=0,p,count=0,i;
clrscr();
printf("enter the memory capacity:");
scanf("%d",&m);
```

```
printf("enter the no of processes:");
scanf("%d",&p);
for(i=0;i<p;i++)
{
printf("\nenter memory req for process%d: ",i+1);
scanf("%d",&m1);
count=count+m1;
if(m1<=m)
{
if(count==m)
printf("there is no further memory remaining:");
printf("the memory allocated for process%d is: %d ",i+1,m);
m2=m-m1;
printf("\nremaining memory is: %d",m2);
m=m2;
}
}
else
{
printf("memory is not allocated for process%d",i+1);
}
printf("\nexternal fragmentation for this process is:%d",m2);
}
```

```
getch();

}
```

12. Write a program to simulate MFT

MFT in C:-

```
#include<stdio.h>

#include<conio.h>

int main()

{

int m,p,s,p1;

int m1[4],i,f,f1=0,f2=0,fra1,fra2,s1,pos;

clrscr();

printf("Enter the memory size:");

scanf("%d",&m);

printf("Enter the no of partitions:");

scanf("%d",&p);

s=m/p;

printf("Each partn size is:%d",s);

printf("\nEnter the no of processes:");

scanf("%d",&p1);

pos=m;

for(i=0;i<p1;i++)

{

if(pos<s)
```

```
{
printf("\nThere is no further memory for process%d",i+1);
m1[i]=0;
break;
}
else
{
printf("\nEnter the memory req for process%d:",i+1);
scanf("%d",&m1[i]);
if(m1[i]<=s)
{
printf("\nProcess is allocated in partition%d",i+1);
fra1=s-m1[i];
printf("\nInternal fragmentation for process is:%d",fra1);
f1=f1+fra1;
pos=pos-s;
}
else
{
printf("\nProcess not allocated in partition%d",i+1);
s1=m1[i];

while(s1>s)
{
```

```
s1=s1-s;

pos=pos-s;

}

pos=pos-s;

fra2=s-s1;

f2=f2+fra2;

printf("\nExternal Fragmentation for this process is:%d",fra2);

}

}

}

printf("\nProcess\tallocatedmemory");

for(i=0;i<p1;i++)

printf("\n%5d\t%5d",i+1,m1[i]);

f=f1+f2;

printf("\nThe tot no of fragmentation is:%d",f);

getch();

return 0;


}
```

13. Write a program to simulate Paging technique

Paging in C:-

```c
#include<stdio.h>

int main()

{
```

```
int ms, ps, nop, np, rempages, i, j, x, y, pa, offset;

int s[10], fno[10][20];



printf("\nEnter the memory size -- ");

scanf("%d",&ms);



printf("\nEnter the page size -- ");

scanf("%d",&ps);



nop = ms/ps;

printf("\nThe no. of pages available in memory are -- %d ",nop);



printf("\nEnter number of processes -- ");

scanf("%d",&np);

rempages = nop;

for(i=1;i<=np;i++)



{



printf("\nEnter no. of pages required for p[%d]-- ",i);

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



if(s[i] >rempages)

{
```

```
printf("\nMemory is Full");

break;

}

rempages = rempages - s[i];



printf("\nEnter pagetable for p[%d] --- ",i);

for(j=0;j<s[i];j++)

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

}



printf("\nEnter Logical Address to find Physical Address ");

printf("\nEnter process no. and pagenumber and offset -- ");



scanf("%d %d %d",&x,&y, &offset);







if(x>np || y>=s[i] || offset>=ps)

printf("\nInvalid Process or Page Number or offset");



else

{ pa=fno[x][y]*ps+offset;

printf("\nThe Physical Address is -- %d\n",pa);
```

```
}

return 0;

}
```

Paging in Python:-

```
import math

process = int(input("Size of process : "))

page_size = int(input("Size of page : "))

memory_size = int(input("Size of memory : "))

total_frames = memory_size*1024*1024 / page_size

total_frames = math.log(total_frames,2)

entries_page_table = process*1024/page_size

physical_memory = memory_size*1024*1024

def calc(n):
    expo = 0
    while(n%2 == 0):
        n/=2
        expo+=1
    return expo

phy = calc(physical_memory)

logical_bits = calc(process*1024)

offset = calc(page_size)

print(f'Total Number of bits in memory are: {phy}')
```

```
print(f'Page Table Entries: {entries_page_table}')
print(f'Total frames in memory: {total_frames}')
print(f'No of bits in logical address: {logical_bits}')
print(f'Total frames in memory: {total_frames}')
print(f'Offset Bits: {offset}')
page_ent = int(entries_page_table)
page_no = []
frame_no = []
valid1 = []
for i in range(page_ent):
    page_tab = int(input('Page no: '))
    frame_tab = int(input('Frame no (-1 for empty): '))
    if frame_tab != -1:
        valid = 1
        frame_no.append(frame_tab)
    else:
        valid = 0
        frame_no.append('...')
    page_no.append(page_tab)
    valid1.append(valid)
for i in range(1):
    print("Page no \t Frame no \t valid")
    for j in range(page_ent):
```

```
        print(f'{ page_no[j]} \t \t {frame_no[j]} \t \t {valid1[j]}')
for i in range(int(input('no. address to be checked: '))):
    c = []
    add = list(map(int, input('Enter address: ').split()))
    add1 = list(map(int, input('Enter offset: ').split()))
    add.reverse()
    sum = 0
    n=0
    for i in add:
        sum+=pow(2,n)*i
        n+=1
    for i in range(page_ent):
        if sum == page_no[i]:
            if valid1[i] == 1:
                print('Page Hit')
                c.append('1')
                break
    if not c:
        print('Page Miss')
```

14. Write a program to simulate Indexed File Allocation Technique

Indexed Allocation in C:-

```c
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
int files[50], indexBlock[50], indBlock, n;
void recurse1();
void recurse2();
void recurse1(){
   printf("Enter the index block: ");
   scanf("%d", &indBlock);
   if (files[indBlock] != 1){
       printf("Enter the number of blocks and the number of files needed
for the index %d on the disk: ", indBlock);
       scanf("%d", &n);
   }
   else{
       printf("%d is already allocated\n", indBlock);
       recurse1();
   }
   recurse2();
}
void recurse2(){
   int ch;
   int flag = 0;
   for (int i=0; i<n; i++){
       scanf("%d", &indexBlock[i]);
       if (files[indexBlock[i]] == 0)
           flag++;
   }
   if (flag == n){
       for (int j=0; j<n; j++){
           files[indexBlock[j]] = 1;
       }
       printf("Allocated\n");
       printf("File Indexed\n");
       for (int k=0; k<n; k++){
           printf("%d ------> %d : %d\n", indBlock, indexBlock[k],
files[indexBlock[k]]);
       }
```

```
    }
    else{
        printf("File in the index is already allocated\n");
        printf("Enter another indexed file\n");
        recurse2();
    }
    printf("Do you want to enter more files?\n");
    printf("Enter 1 for Yes, Enter 0 for No: ");
    scanf("%d", &ch);
    if (ch == 1)
        recurse1();
    else
        exit(0);
    return;
}
int main()
{
    for(int i=0;i<50;i++)
        files[i]=0;
    recurse1();
    return 0;
}
```

15. Write a program to simulate Contiguous File Allocation Technique

Contiguous File Allocation in C:-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void recurse(int files[]){
   int flag = 0, startBlock, len, j, k, ch;
   printf("Enter the starting block and the length of the files: ");
   scanf("%d%d", &startBlock, &len);
   for (j=startBlock; j<(startBlock+len); j++){
       if (files[j] == 0)
           flag++;
   }
   if(len == flag){
       for (int k=startBlock; k<(startBlock+len); k++){
```

```
            if (files[k] == 0){
                files[k] = 1;
                printf("%d\t%d\n", k, files[k]);
            }
        }
        if (k != (startBlock+len-1))
            printf("The file is allocated to the disk\n");
    }
    else
        printf("The file is not allocated to the disk\n");
    printf("Do you want to enter more files?\n");
    printf("Press 1 for YES, 0 for NO: ");
    scanf("%d", &ch);
    if (ch == 1)
        recurse(files);
    else
        exit(0);
    return;
}
int main()
{
int files[50];
for(int i=0;i<50;i++)
files[i]=0;
printf("Files Allocated are :\n");
recurse(files);
getch();
return 0;
}
```

16. Write program to simulate Linked File Allocation Technique

Linked Allocation in C:-

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
void recursivePart(int pages[]){
    int st, len, k, c, j;
    printf("Enter the index of the starting block and its length: ");
    scanf("%d%d", &st, &len);
    k = len;
    if (pages[st] == 0){
        for (j = st; j < (st + k); j++){
            if (pages[j] == 0){
```

```
                pages[j] = 1;
                printf("%d------>%d\n", j, pages[j]);
            }
            else {
                printf("The block %d is already allocated \n", j);
                k++;
            }
        }
    }
    else
        printf("The block %d is already allocated \n", st);
    printf("Do you want to enter more files? \n");
    printf("Enter 1 for Yes, Enter 0 for No: ");
    scanf("%d", &c);
    if (c==1)
        recursivePart(pages);
    else
        exit(0);
    return;
}
int main(){
    int pages[50], p, a;
    for (int i = 0; i < 50; i++)
        pages[i] = 0;
    printf("Enter the number of blocks already allocated: ");
    scanf("%d", &p);
    printf("Enter the blocks already allocated: ");
    for (int i = 0; i < p; i++){
        scanf("%d", &a);
        pages[a] = 1;
    }
    recursivePart(pages);
    getch();
    return 0;
}
```

17. Write a program to calculate safe sequence using Banker's algorithm.

```c
#include <stdio.h>

int current[5][5], maximum_claim[5][5], available[5];

int allocation[5] = {0, 0, 0, 0, 0};

int maxres[5], running[5], safe = 0;

int counter = 0, i, j, exec, resources, processes, k = 1;

int main()

{

printf("\nEnter number of processes: ");

    scanf("%d", &processes);

    for (i = 0; i < processes; i++)

{

        running[i] = 1;

        counter++;

    }

    printf("\nEnter number of resources: ");

    scanf("%d", &resources);

    printf("\nEnter Claim Vector:");

    for (i = 0; i < resources; i++)

{

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

    }

  printf("\nEnter Allocated Resource Table:\n");

    for (i = 0; i < processes; i++)

{
```

```
       for(j = 0; j < resources; j++)
{
  scanf("%d", &current[i][j]);
        }
    }
    printf("\nEnter Maximum Claim Table:\n");
    for (i = 0; i < processes; i++)
{
        for(j = 0; j < resources; j++)
{
            scanf("%d", &maximum_claim[i][j]);
        }
    }
printf("\nThe Claim Vector is: ");
    for (i = 0; i < resources; i++)
{
       printf("\t%d", maxres[i]);
}
    printf("\nThe Allocated Resource Table:\n");
    for (i = 0; i < processes; i++)
{
       for (j = 0; j < resources; j++)
{
            printf("\t%d", current[i][j]);
```

```
        }
printf("\n");
    }
    printf("\nThe Maximum Claim Table:\n");
    for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
{
       printf("\t%d", maximum_claim[i][j]);
        }
        printf("\n");
    }
    for (i = 0; i < processes; i++)
{
        for (j = 0; j < resources; j++)
{
            allocation[j] += current[i][j];
        }
    }
    printf("\nAllocated resources:");
    for (i = 0; i < resources; i++)
{
        printf("\t%d", allocation[i]);
    }
```

```
    for (i = 0; i < resources; i++)
{
       available[i] = maxres[i] - allocation[i];
}
    printf("\nAvailable resources:");
    for (i = 0; i < resources; i++)
{
        printf("\t%d", available[i]);
    }
    printf("\n");
    while (counter != 0)
{
        safe = 0;
        for (i = 0; i < processes; i++)
{
            if (running[i])
{
                exec = 1;
                for (j = 0; j < resources; j++)
{
                    if (maximum_claim[i][j] - current[i][j] > available[j])
{
                        exec = 0;
                        break;
```

```
                    }

                }

                if (exec)

{

                    printf("\nProcess%d is executing\n", i + 1);

                    running[i] = 0;

                    counter--;

                    safe = 1;

                    for (j = 0; j < resources; j++)

{

                        available[j] += current[i][j];

                    }

               break;

                }

            }

        }

        if (!safe)

{

            printf("\nThe processes are in unsafe state.\n");

            break;

        }

else

{

            printf("\nThe process is in safe state");
```

```
            printf("\nAvailable vector:");

            for (i = 0; i < resources; i++)

{

                printf("\t%d", available[i]);

            }

       printf("\n");

         }

    }

}
```

Bankers in Python:-

#def main():

 # track = []

 # processes = int(input("number of processes : "))

 # resources = int(input("number of resources : "))

 # max_resources = [int(i) for i in input("maximum resources : ").split()]

 # print("\n-- allocated resources for each process --")

 # currently_allocated = [[int(i) for i in input(f"process {j + 1} : ").split()] for j in range(processes)]

 # print("\n-- maximum resources for each process --")

 # max_need = [[int(i) for i in input(f"process {j + 1} : ").split()] for j in range(processes)]

 # allocated = [0] * resources

 # for i in range(processes):

 # for j in range(resources):

```
#         allocated[j] += currently_allocated[i][j]
#   print(f"\ntotal allocated resources : {allocated}")
#   available = [max_resources[i] - allocated[i] for i in range(resources)]
#   print(f"total available resources : {available}\n")
#   running = [True] * processes
#   count = processes
#   while count != 0:
#     safe = False
#     for i in range(processes):
#       if running[i]:
#         executing = True
#         for j in range(resources):
#           if max_need[i][j] - currently_allocated[i][j] > available[j]:
#             executing = False
#             break
#         if executing:
#           print(f"process {i + 1} is executing")
#         c = i+1
#         track.append(c)
#         running[i] = False
#         count -= 1
#         safe = True
#         for j in range(resources):
```

```
#                 available[j] += currently_allocated[i][j]
#             break
#     if not safe:
#         print("the processes are in an unsafe state.")
#         break
#   for i in track:
#     print(i, end="-->")
#if __name__ == '__main__':
#   main()
```

```
def main():
    processes = int(input("number of processes : "))
    resources = int(input("number of resources : "))
    max_resources = [int(i) for i in input("maximum resources : ").split()]

    print("\n-- allocated resources for each process --")
    currently_allocated = [[int(i) for i in input(f"process {j + 1} : ").split()] for j in
range(processes)]
```

```
print("\n-- maximum resources for each process --")
max_need = [[int(i) for i in input(f"process {j + 1} : ").split()] for j in range(processes)]

allocated = [0] * resources
for i in range(processes):
    for j in range(resources):
        allocated[j] += currently_allocated[i][j]
print(f"\ntotal allocated resources : {allocated}")

available = [max_resources[i] - allocated[i] for i in range(resources)]
print(f"total available resources : {available}\n")

running = [True] * processes
count = processes
while count != 0:
    safe = False
    for i in range(processes):
        if running[i]:
            executing = True
            for j in range(resources):
                if max_need[i][j] - currently_allocated[i][j] > available[j]:
                    executing = False
```

```
                break
            if executing:
                print(f"process {i + 1} is executing")
                running[i] = False
                count -= 1
                safe = True
                for j in range(resources):
                    available[j] += currently_allocated[i][j]
                break
        if not safe:
            print("the processes are in an unsafe state.")
            break

        print(f"the process is in a safe state.\navailable resources : {available}\n")


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

18. Write a program to implement the FCFS Disk Scheduling Policy

FCFS in C:-

```
#include<conio.h>
```

```
#include<stdio.h>
int main()
{
int i,j,sum=0,n;
int ar[20],tm[20];
int disk
clrscr();
printf("enter number of location\t");
scanf("%d",&n);
printf("enter position of head\t");
scanf("%d",&disk);
printf("enter elements of disk queue\n");
for(i=0;i<n;i++)
{
scanf("%d",&ar[i]);
tm[i]=disk-ar[i];

if(tm[i]<0)
{
tm[i]=ar[i]-disk;
}
disk=ar[i];
sum=sum+tm[i];
```

```
}
/*for(i=0;i<n;i++)
{
printf("\n%d",tm[i]);
} */
printf("\n movement of total cylinders %d",sum);
getch();
return 0;
}
```

FCFS in Python:

```
# Python program to demonstrate
# FCFS Disk Scheduling algorithm


def FCFS(arr, head):

    seek_count = 0
    distance, cur_track = 0, 0

    for i in range(size):
            cur_track = arr[i];

            # calculate absolute distance
```

```
            distance = abs(cur_track - head);

            # increase the total count
            seek_count += distance;

            # accessed track is now new head
            head = cur_track;

    print("Total number of seek operations = ",
                                                    seek_count);

    # Seek sequence would be the same
    # as request array sequence
    print("Seek Sequence is");

    for i in range(size):
            print(arr[i]);

# Driver code
if __name__ == '__main__':
    #size = 8
    ## request array
    #arr = [ 176, 79, 34, 60,
```

```
#                  92, 11, 41, 114 ];

#head = 50;

arr = []

size = int(input("Enter number of elements: "))


# iterating till the range

print("Enter the elements one by one:\n")

for i in range(0, size):

        arr.append(int(input())) # adding the element


print('the processes are: ', arr)

head = int(input("Initial position of head:"))


FCFS(arr, head)
```

19. Write a program to implement the following SSTF Disk Scheduling Policy

SSTF in C:-

```c
#include<stdio.h>

#include<stdlib.h>

int main()

{

   int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;

   printf("Enter the number of Requests\n");

   scanf("%d",&n);

   printf("Enter the Requests sequence\n");
```

```
for(i=0;i<n;i++)
 scanf("%d",&RQ[i]);
printf("Enter initial head position\n");
scanf("%d",&initial);

while(count!=n)
{
    int min=1000,d,index;
    for(i=0;i<n;i++)
    {
       d=abs(RQ[i]-initial);
       if(min>d)
       {
           min=d;
           index=i;
       }

    }
    TotalHeadMoment=TotalHeadMoment+min;
    initial=RQ[index];
    RQ[index]=1000;
    count++;
}
```

```
    printf("Total head movement is %d\n",TotalHeadMoment);

    return 0;

}
```

SSTF in Python:-

```
# Python3 program for implementation of
# SSTF disk scheduling


# Calculates difference of each
# track number with the head position
def calculateDifference(queue, head, diff):
        for i in range(len(diff)):
                diff[i][0] = abs(queue[i] - head)


# find unaccessed track which is
# at minimum distance from head
def findMin(diff):


        index = -1
        minimum = 999999999


        for i in range(len(diff)):
                if (not diff[i][1] and
                                minimum > diff[i][0]):
```

```
                minimum = diff[i][0]
                index = i
        return index

def shortestSeekTimeFirst(request, head):
        if (len(request) == 0):
                return

        l = len(request)
        diff = [0] * l

        # initialize array
        for i in range(l):
                diff[i] = [0, 0]

        # count total number of seek operation
        seek_count = 0

        # stores sequence in which disk
        # access is done
        seek_sequence = [0] * (l + 1)

        for i in range(l):
```

```
            seek_sequence[i] = head
            calculateDifference(request, head, diff)
            index = findMin(diff)

            diff[index][1] = True

            # increase the total count
            seek_count += diff[index][0]

            # accessed track is now new head
            head = request[index]

    # for last accessed track
    seek_sequence[len(seek_sequence) - 1] = head

    print("Total number of seek operations =",
                                                    seek_count)

    print("Seek Sequence is")

    # print the sequence
    for i in range(l + 1):
            print(seek_sequence[i])
```

```
# Driver code
if __name__ =="__main__":

    # request array
    proc = []
    size = int(input("Enter number of elements: "))
    head = int(input("Initial position of head:"))

    # iterating till the range
    print("Enter the elements one by one:\n")
    for i in range(0, size):
        proc.append(int(input())) # adding the element

    shortestSeekTimeFirst(proc, head)
```

20. Write a program to implement the SCAN Disk Scheduling Policy

SCAN in C:-

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
```

```
printf("Enter the number of Requests:-\n");

scanf("%d",&n);

printf("Enter the Requests sequence:-\n");

for(i=0;i<n;i++)

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

printf("Enter initial head position:-\n");

scanf("%d",&initial);

printf("Enter total disk size:-\n");

scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0:-\n");

scanf("%d",&move);



for(i=0;i<n;i++)

{

    for(j=0;j<n-i-1;j++)

    {

        if(RQ[j]>RQ[j+1])

        {

            int temp;

            temp=RQ[j];

            RQ[j]=RQ[j+1];

            RQ[j+1]=temp;

        }
```

```
    }
}


int index;
for(i=0;i<n;i++)
{
    if(initial<RQ[i])
    {
        index=i;
        break;
    }
}


if(move==1)
{
    for(i=index;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }
    TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
    initial = size-1;
    for(i=index-1;i>=0;i--)
    {
```

```
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];


        }
    }
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
        initial =0;
        for(i=index;i<n;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
        }
    }


    printf("Total head movement is %d\n",TotalHeadMoment);
    return 0;
}
```

SCAN in Python:-

```
size = int(input('Enter no of tracks: '))
head_pos = int(input('Enter Head Position: '))
scan = list(map(int,input('Enter Req Array: ').split()))
dire = int(input('Enter Direction 1 for left 0 for right: '))
head1 = []
dis =0
#print(scan)
scan.sort()
for s in range(len(scan)):
    if scan[s] > head_pos:
        mid = s
        break
if dire == 1:
    l1 = scan[:mid]
    l2 = scan[mid:]
    #print(l1)
    l1.reverse()
    l1.append(0)
    #print(l2)
    dis = head_pos + l2[-1]
    l3 = l1+l2
```

```
    print(f'The Tracs are : {l3}')
    print(dis)
else:
    l1 = scan[:mid]
    l2 = scan[mid:]
    l1.reverse()
    l2.append(199)
    l3 = l2 + l1
    dis = head_pos + (199-head_pos) + (199-l1[-1])
    print(f'The tracs are {l3}')
    print(dis)
```

21. Write a program to implement the following LOOK Disk Scheduling Policy

LOOK in C:-

```c
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
   printf("Enter the number of Requests:-\n");
   scanf("%d",&n);
   printf("Enter the Requests sequence:-\n");
   for(i=0;i<n;i++)
    scanf("%d",&RQ[i]);
```

```
printf("Enter initial head position:-\n");

scanf("%d",&initial);

printf("Enter total disk size:-\n");

scanf("%d",&size);

printf("Enter the head movement direction for high 1 and for low 0:-\n");

scanf("%d",&move);



for(i=0;i<n;i++)

{

    for( j=0;j<n-i-1;j++)

    {

        if(RQ[j]>RQ[j+1])

        {

            int temp;

            temp=RQ[j];

            RQ[j]=RQ[j+1];

            RQ[j+1]=temp;

        }



    }

}



int index;

for(i=0;i<n;i++)
```

```
{
    if(initial<RQ[i])
    {
        index=i;
        break;
    }
}

if(move==1)
{
    for(i=index;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }

    for( i=0;i<index;i++)
    {
         TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
         initial=RQ[i];

    }
}
else
```

```
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];

        }
    }

    printf("Total head movement is %d\n",TotalHeadMoment);
    return 0;
}
```

LOOK in Python:-

size = int(input('Enter no of tracks: '))

head_pos = int(input('Enter Head Position: '))

scan = list(map(int,input('Enter Req Array: ').split()))

dire = int(input('Enter Direction 1 for left 0 for right: '))

```
head1 = []
dis =0
#print(scan)
scan.sort()
for s in range(len(scan)):
    if scan[s] > head_pos:
        mid = s
        break
if dire == 1:
    l1 = scan[:mid]
    l2 = scan[mid:]
    #print(l1)
    l1.reverse()
    l1.append(0)
    #print(l2)
    dis = head_pos + l2[-1]
    l3 = l1+l2
    print(f'The Tracs are : {l3}')
    print(dis)
else:
    l1 = scan[:mid]
    l2 = scan[mid:]
    l1.reverse()
```

l2.append(199)

l3 = l2 + l1

dis = head_pos + (199-head_pos) + (199-l1[-1])

print(f'The tracs are {l3}')

print(dis)

22. Write a program to implement the C-SCAN Disk Scheduling Policy

C-Scan in C:-

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

   int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;

   printf("Enter the number of Requests\n");

   scanf("%d",&n);

   printf("Enter the Requests sequence\n");

   for(i=0;i<n;i++)

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

   printf("Enter initial head position\n");

   scanf("%d",&initial);

   printf("Enter total disk size\n");

   scanf("%d",&size);

   printf("Enter the head movement direction for high 1 and for low
0\n");

   scanf("%d",&move);
```

```
// logic for C-Scan disk scheduling


    /*logic for sort the request array */
for(i=0;i<n;i++)
{
    for( j=0;j<n-i-1;j++)
    {
        if(RQ[j]>RQ[j+1])
        {
            int temp;
            temp=RQ[j];
            RQ[j]=RQ[j+1];
            RQ[j+1]=temp;
        }
```

```
    }
}
```

```
int index;
for(i=0;i<n;i++)
{
    if(initial<RQ[i])
    {
        index=i;
        break;
```

```
    }
}


// if movement is towards high value
if(move==1)
{
    for(i=index;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }
    //  last movement for max size
    TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
    /*movement max to min disk */
    TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
    initial=0;
    for( i=0;i<index;i++)
    {
         TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
         initial=RQ[i];


    }
}
// if movement is towards low value
else
```

```
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        //  last movement for min size
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
        /*movement min to max disk */
        TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
        initial =size-1;
        for(i=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];

        }
    }

    printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

C-Scan in Python:-

 # Python3 program to demonstrate

```
# C-SCAN Disk Scheduling algorithm
    # before reversing the direction
    left.append(0)
    right.append(disk_size - 1)

    # Tracks on the left of the
    # head will be serviced when
    # once the head comes back
    # to the beggining (left end).
    for i in range(size):
            if (arr[i] < head):
                    left.append(arr[i])
            if (arr[i] > head):
                    right.append(arr[i])

    # Sorting left and right vectors
    left.sort()
    right.sort()

    # First service the requests
    # on the right side of the
    # head.
    for i in range(len(right)):
```

```
            cur_track = right[i]
def CSCAN(arr, head):

    seek_count = 0
    distance = 0
    cur_track = 0
    left = []
    right = []
    seek_sequence = []

    # Appending end values
    # which has to be visited

            # Appending current track
            # to seek sequence
            seek_sequence.append(cur_track)

            # Calculate absolute distance
            distance = abs(cur_track - head)

            # Increase the total count
            seek_count += distance
```

```
        # Accessed track is now new head
        head = cur_track

# Once reached the right end
# jump to the beggining.
head = 0

# adding seek count for head returning from 199 to 0
seek_count += (disk_size - 1)

# Now service the requests again
# which are left.
for i in range(len(left)):
        cur_track = left[i]

        # Appending current track
        # to seek sequence
        seek_sequence.append(cur_track)

        # Calculate absolute distance
        distance = abs(cur_track - head)

        # Increase the total count
```

```
            seek_count += distance

            # Accessed track is now the new head
            head = cur_track

    print("Total number of seek operations =",
            seek_count)
    print("Seek Sequence is")
    print(*seek_sequence, sep="\n")

# Driver code
# request array
arr =[]
size = int(input("Enter the number of requests: "))
disk_size = int(input("Enter the track size: "))
print("Enter the request series one by one:")
for i in range(size):
    arr.append(int(input()))
head = int(input("Enter the initial head position: "))
print("Initial position of head:", head)
CSCAN(arr, head)
```

23. Write a program to implement the following C-LOOK Disk Scheduling Policy

C-Look in C:-

```c
#include<stdio.h>

#include<stdlib.h>

int main()

{

   int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;

   printf("Enter the number of Requests\n");

   scanf("%d",&n);

   printf("Enter the Requests sequence\n");

   for(i=0;i<n;i++)

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

   printf("Enter initial head position\n");

   scanf("%d",&initial);

   printf("Enter total disk size\n");

   scanf("%d",&size);

   printf("Enter the head movement direction for high 1 and for low
0\n");

   scanf("%d",&move);



   // logic for C-look disk scheduling



       /*logic for sort the request array */

   for(i=0;i<n;i++)

   {

       for( j=0;j<n-i-1;j++)

       {

           if(RQ[j]>RQ[j+1])
```

```
            {
                int temp;
                temp=RQ[j];
                RQ[j]=RQ[j+1];
                RQ[j+1]=temp;
            }
```

```
        }
    }
```

```
    int index;
    for(i=0;i<n;i++)
    {
        if(initial<RQ[i])
        {
            index=i;
            break;
        }
    }

    // if movement is towards high value
    if(move==1)
    {
        for(i=index;i<n;i++)
        {
```

```
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for( i=0;i<index;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];

        }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=n-1;i>=index;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
             initial=RQ[i];
```

```
        }
    }


    printf("Total head movement is %d",TotalHeadMoment);
    return 0;
}
```

C-LOOK in Python:-

```
# Function to perform C-LOOK on the request
# array starting from the given head
def CLOOK(x, head):

    seek_count = 0
    distance = 0
    cur_track = 0

    left = []
    right = []

    seek_sequence = []

    # Tracks on the left of the
    # head will be serviced when
    # once the head comes back
```

```
# to the beginning (left end)
for i in range(N):
        if (x[i] < head):
                left.append(x[i])
        if (x[i] > head):
                right.append(x[i])

# Sorting left and right vectors
left.sort()
right.sort()

# First service the requests
# on the right side of the
# head
for i in range(len(right)):
        cur_track = right[i]

        # Appending current track
        # seek sequence
        seek_sequence.append(cur_track)

        # Calculate absolute distance
        distance = abs(cur_track - head)
```

```
        # Increase the total count
        seek_count += distance

        # Accessed track is now new head
        head = cur_track

# Once reached the right end
# jump to the last track that
# is needed to be serviced in
# left direction
seek_count += abs(head - left[0])
head = left[0]

# Now service the requests again
# which are left
for i in range(len(left)):
        cur_track = left[i]

        # Appending current track to
        # seek sequence
        seek_sequence.append(cur_track)
```

```
            # Calculate absolute distance
            distance = abs(cur_track - head)

            # Increase the total count
            seek_count += distance

            # Accessed track is now the new head
            head = cur_track

    print("Total number of seek operations =",
            seek_count)
    print("Seek Sequence is")

    for i in range(len(seek_sequence)):
            print(seek_sequence[i])

# Driver code

n = []

# number of elements as input
N = int(input("Enter number of elements: "))
disk_size = int(input("Enter disk size: "))
```

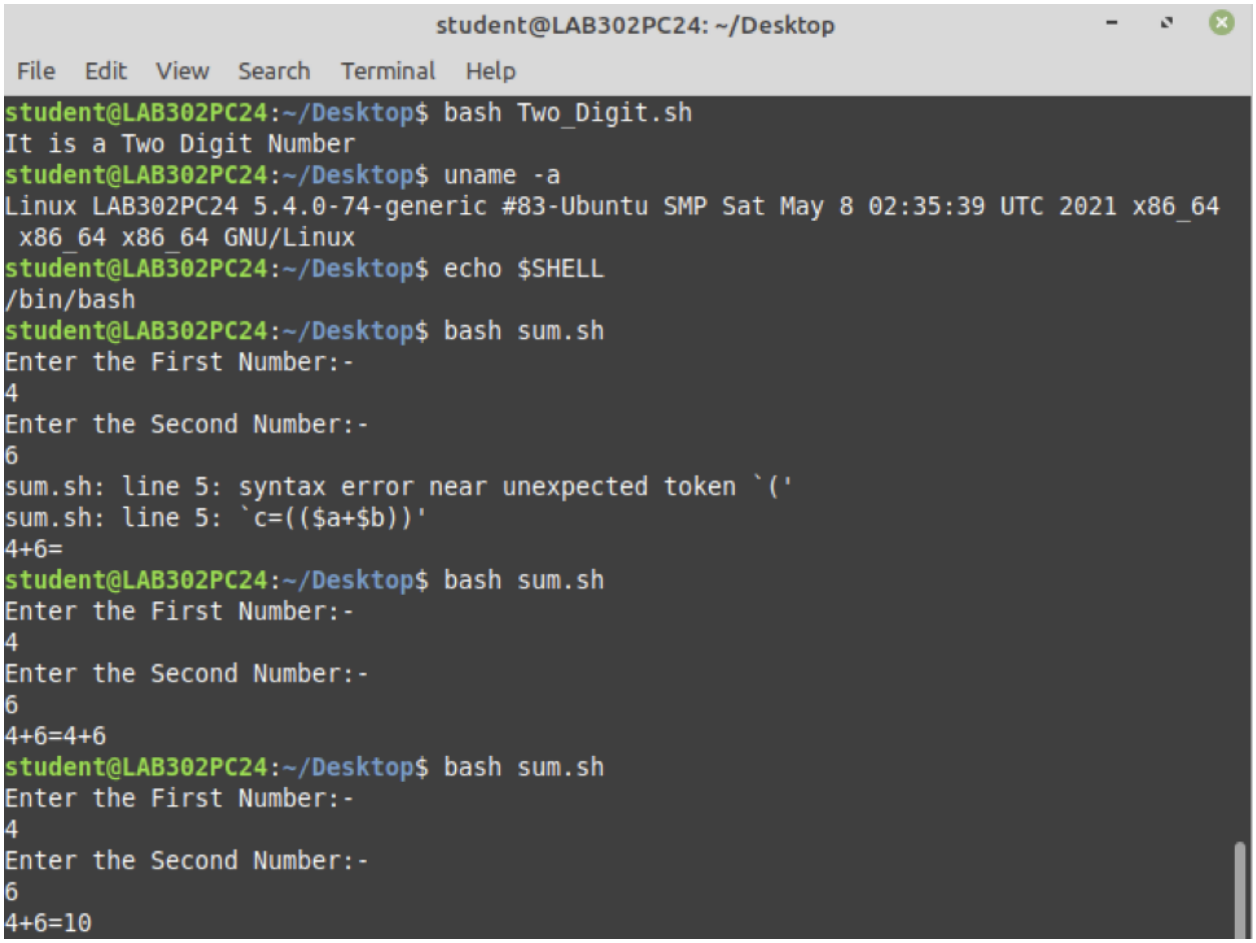
```
# iterating till the range

print("Enter the elements one by one:\n")

for i in range(0, N):

    x = int(input())


    n.append(x) # adding the element


print('the processes are: ', n)

head = int(input("Initial position of head:"))

CLOOK(n, head)
```

24. Shell Programming

Program 1:-

Addition of Two Numbers:-

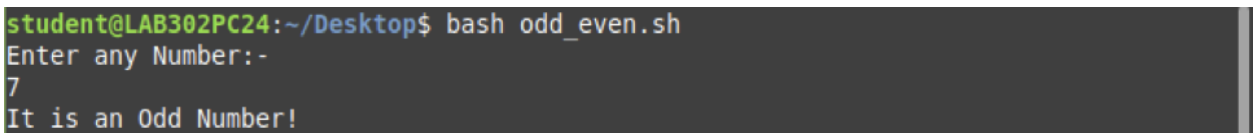
```
echo "Enter the First Number:-"
read a
echo "Enter the Second Number:-"
read b
c=$(($a+$b))
echo $a+$b=$c
```



Program 2:-

Odd Even:-

```
echo "Enter any Number:-"
read n
if [[ ($n%2 -eq 0) ]];
then
   echo "It is an Even Number!"
else
   echo "It is an Odd Number!"
fi
```



Program 3:-

Sum of N Numbers:-

```
a=1
echo "Enter the Number of elements:-"
read n
s=0
while[$a -le $n]
do
   s=$((s+a))
   a=$((a+1))
done
echo "The Sum is “$s
```

```
student@LAB302PC24:~/Desktop$ bash sum_n.sh
Enter the Number of elements:-
5
The Sum is 15
student@LAB302PC24:~/Desktop$
```

Program 4:-

Vote:-

```
echo "Enter you Age:-"
read a
if [[(a -ge 18)]];
then
   echo "Eligible to Vote"
else
   echo "Not Eligible to Vote"
fi
```

```
student@LAB302PC24:~/Desktop$ bash vote.sh
Enter you Age:-
27
Eligible to Vote
student@LAB302PC24:~/Desktop$
```

Program 5:-

Filenames with "A":- **(Make sure to create text files [.txt extension] on the desktop)**

```
for k in a*
do
echo "file name is $k"
cat $k
done
```

```
student@LAB302PC24:~/Desktop$ bash filename.sh
file name is a1.txt
hello its me here at TSEC
file name is a2.txt
Hello its me here again at TSEC
student@LAB302PC24:~/Desktop$
```

Program 6:-

Factorial:-

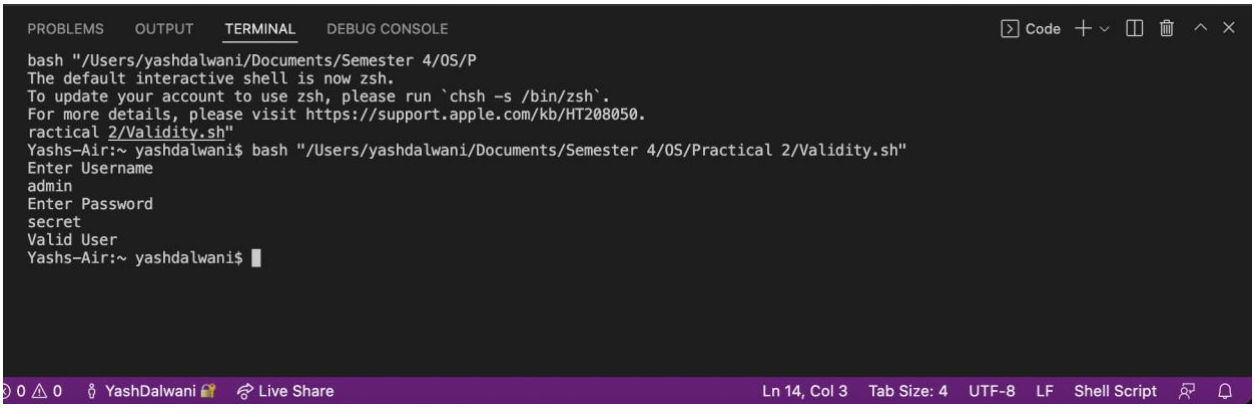
```
echo "Enter a Number:-"
Read number
f=1
for((i=1;i<=number;i++))
do
        f=$((f*i))
done
echo "Factorial is. "$f
```

```
student@LAB302PC24:~/Desktop$ bash factorial.sh
Enter the Number:-
5
Factorial is:- 120
student@LAB302PC24:~/Desktop$
```

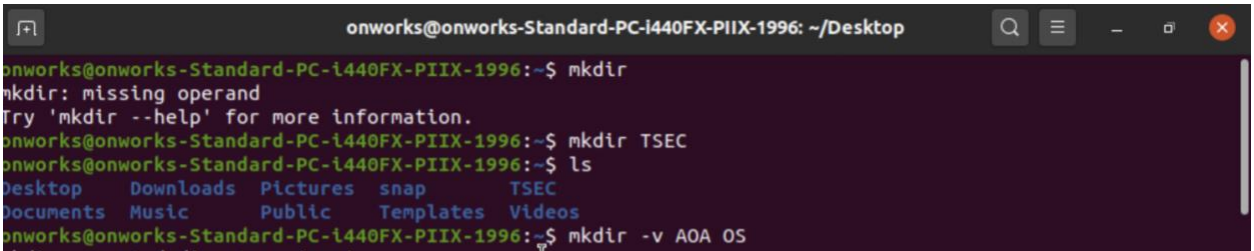
Program 7:-

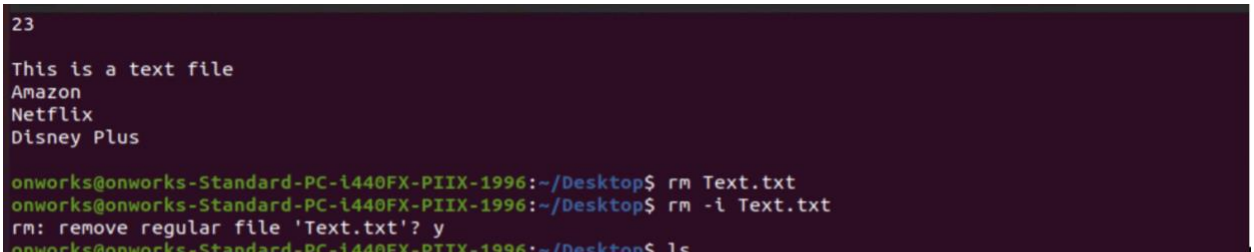
Function:-

```
function enter()
{
        echo "Enter Username"
        read u
        echo "Enter Password"
        read p
}
enter
if [[($u=="admin" && $p=="secret")]];
then
        echo "Valid User"
else
        echo "Invalid User"
fi
```

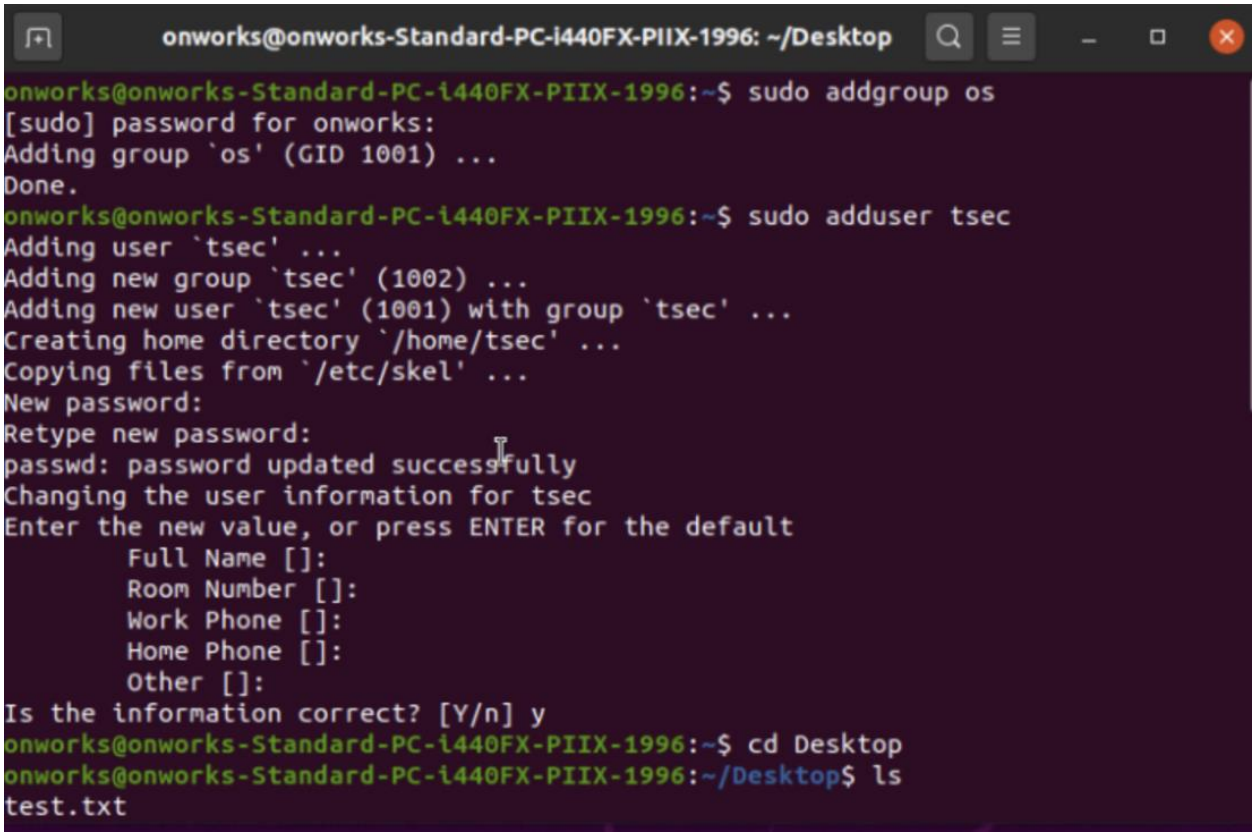


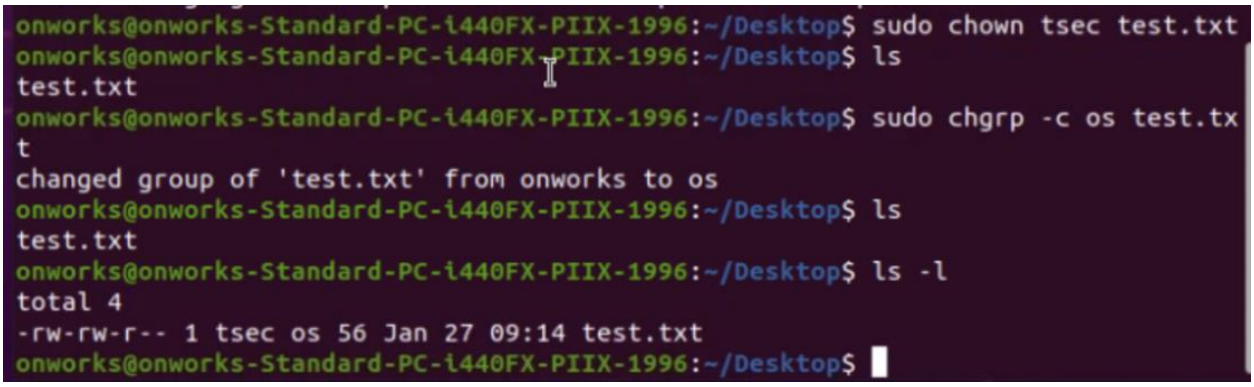
25. Linux Commands practiced in the first lab:-



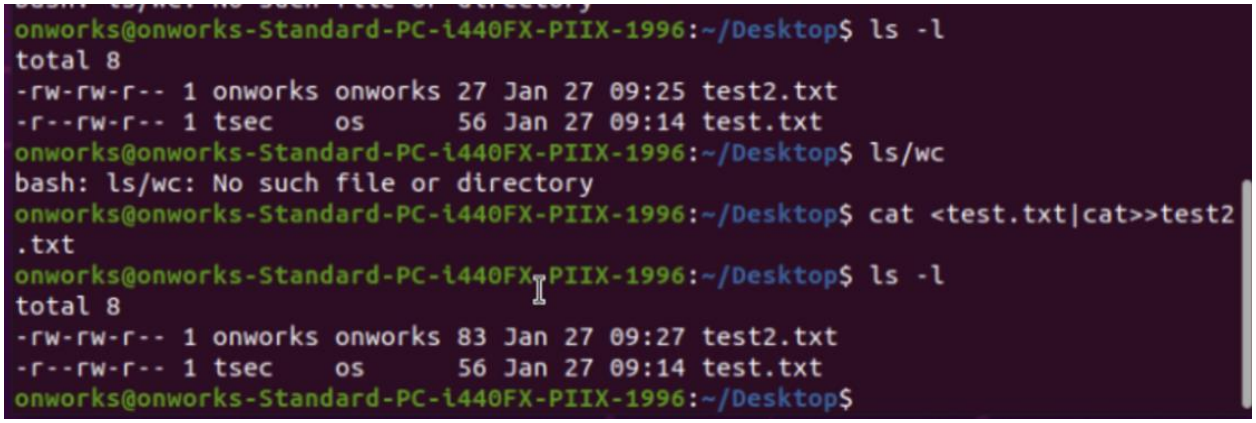

23
This is a text file
Amazon
Netflix
Disney Plus
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ rm Text.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ rm -i Text.txt
rm: remove regular file 'Text.txt'? y

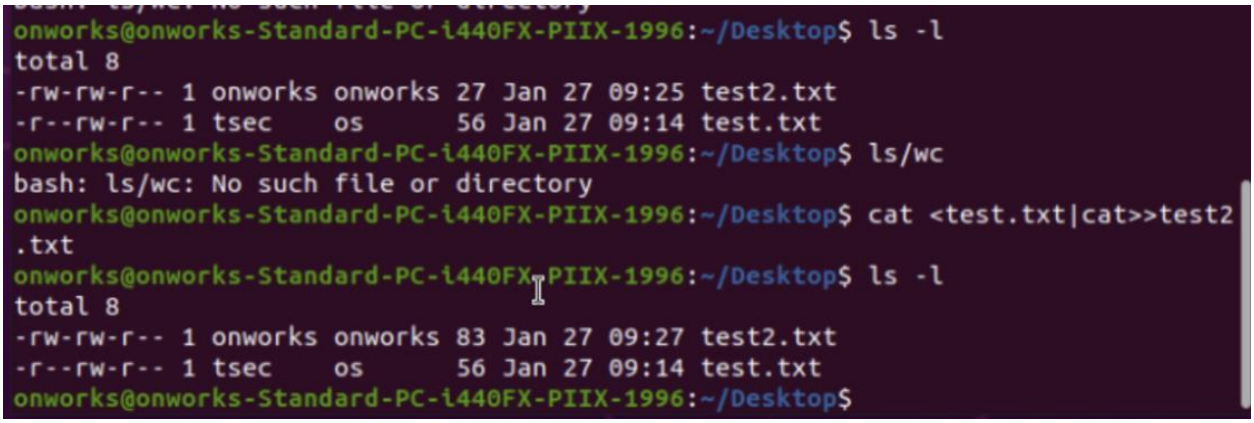
```
265
26
y
3256
3165731
trea
eg
ds
dhsfhtsdgb
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sort Stream.txt

Amazon
Disney Plus
Netflix
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sort Again.txt


26
26
265
3
3165731
3256
5
6
7
8
dhsfhtsdgb
ds
eg
trea
y
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ wc Stream.txt
 4  4 31 Stream.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ wc -m Stream.txt
29 Stream.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ `
```

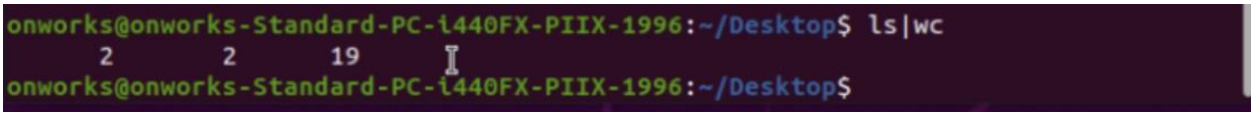

onworks@onworks-Standard-PC-i440FX-PIIX-1996: ~/Desktop
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo addgroup os
[sudo] password for onworks:
Adding group `os' (GID 1001) ...
Done.
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ sudo adduser tsec
Adding user `tsec' ...
Adding new group `tsec' (1002) ...
Adding new user `tsec' (1001) with group `tsec' ...
Creating home directory `/home/tsec' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for tsec
Enter the new value, or press ENTER for the default
Full Name []:
Room Number []:
Work Phone []:
Home Phone []:
Other []:
Is the information correct? [Y/n] y
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cd Desktop
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
test.txt

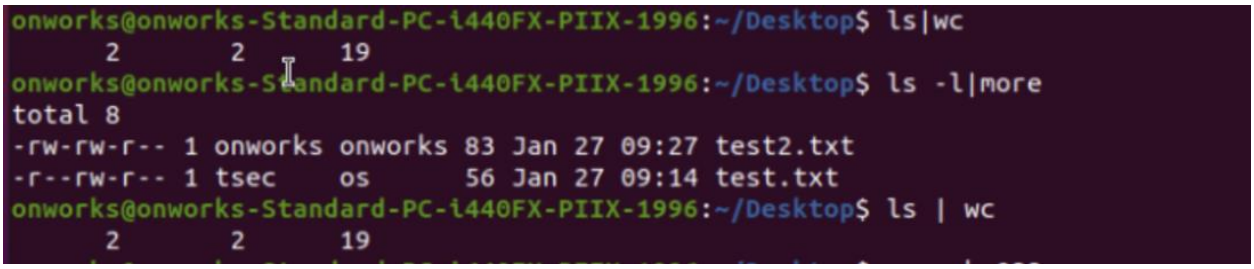

onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sudo chown tsec test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sudo chgrp -c os test.tx
t
changed group of 'test.txt' from onworks to os
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 4
-rw-rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$

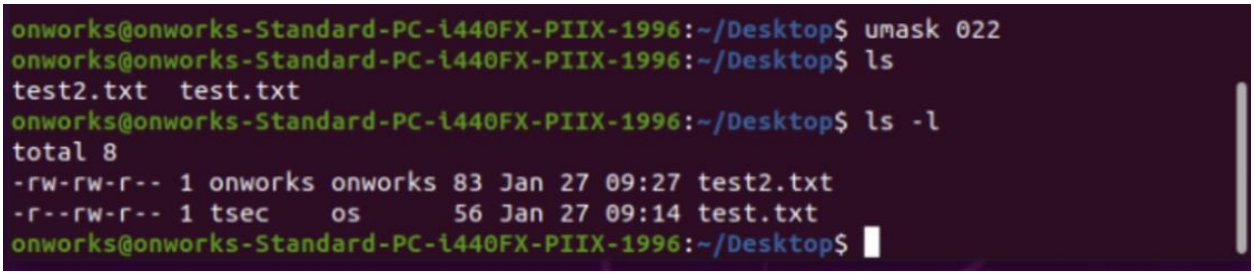

onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sudo chgrp -c os test.tx
t
changed group of 'test.txt' from onworks to os
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 4
-rw-rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ sudo chmod u=r test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 4
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$

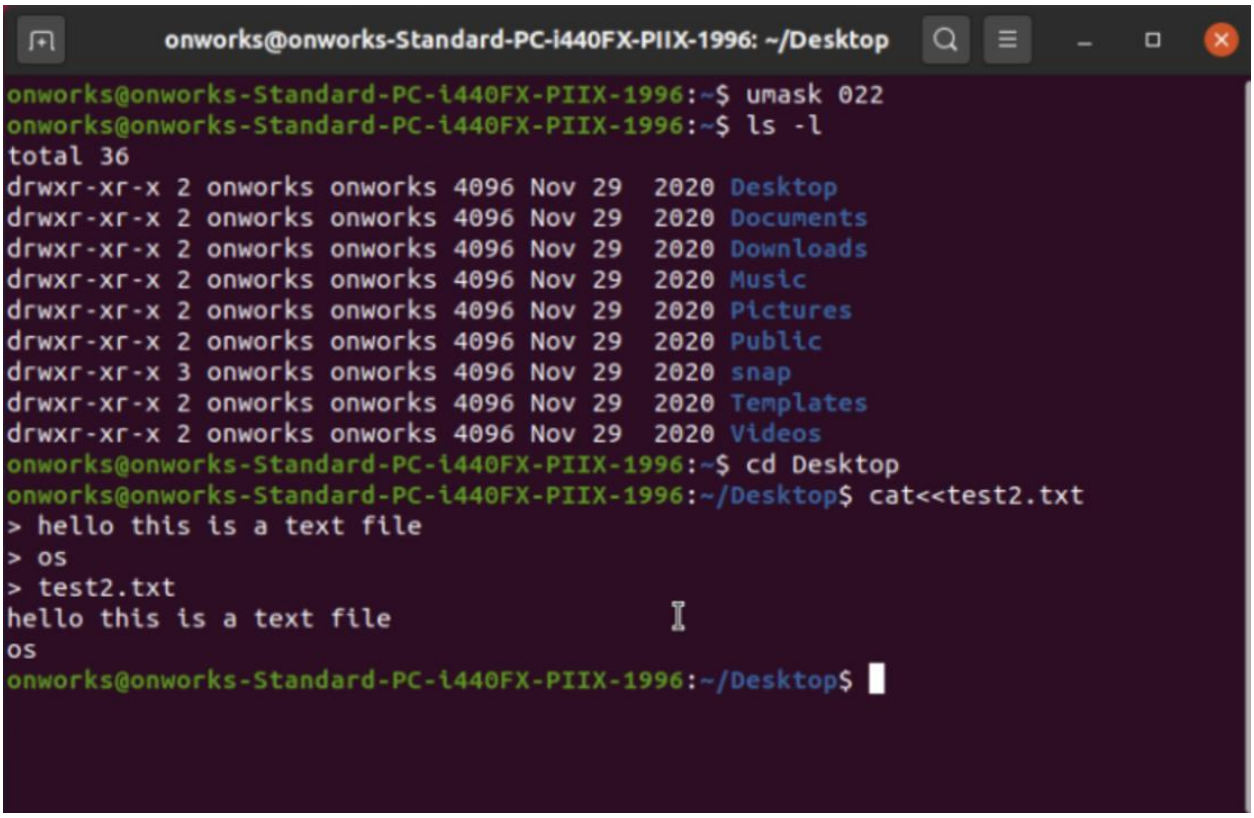

onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 27 Jan 27 09:25 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls/wc
bash: ls/wc: No such file or directory
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ cat <test.txt|cat>>test2
.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 83 Jan 27 09:27 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$

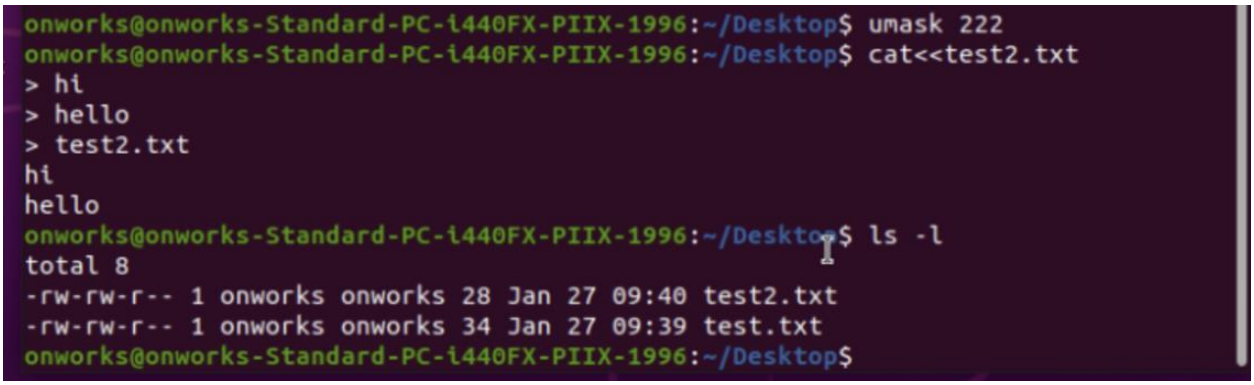

onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 27 Jan 27 09:25 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls/wc
bash: ls/wc: No such file or directory
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ cat <test.txt|cat>>test2
.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 83 Jan 27 09:27 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$

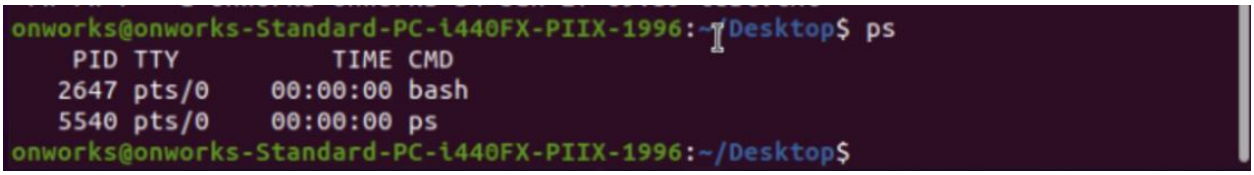

onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls|wc
2 2 19
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$


onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls|wc
2 2 19
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l|more
total 8
-rw-rw-r-- 1 onworks onworks 83 Jan 27 09:27 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls | wc
2 2 19


onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ umask 022
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls
test2.txt test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 83 Jan 27 09:27 test2.txt
-r--rw-r-- 1 tsec os 56 Jan 27 09:14 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$


onworks@onworks-Standard-PC-i440FX-PIIX-1996: ~/Desktop
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ umask 022
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ ls -l
total 36
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Desktop
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Documents
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Downloads
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Music
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Pictures
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Public
drwxr-xr-x 3 onworks onworks 4096 Nov 29 2020 snap
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Templates
drwxr-xr-x 2 onworks onworks 4096 Nov 29 2020 Videos
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ cd Desktop
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ cat<<test2.txt
> hello this is a text file
> os
> test2.txt
hello this is a text file
os
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$


onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ umask 222
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ cat<<test2.txt
> hi
> hello
> test2.txt
hi
hello
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ls -l
total 8
-rw-rw-r-- 1 onworks onworks 28 Jan 27 09:40 test2.txt
-rw-rw-r-- 1 onworks onworks 34 Jan 27 09:39 test.txt
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$


onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ps
PID TTY TIME CMD
2647 pts/0 00:00:00 bash
5540 pts/0 00:00:00 ps
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ps -e
    PID TTY          TIME CMD
      1 ?        00:00:07 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-kblockd
      8 ?        00:00:00 mm_percpu_wq
      9 ?        00:00:00 ksoftirqd/0
     10 ?        00:00:01 rcu_sched
     11 ?        00:00:00 migration/0
     12 ?        00:00:00 idle_inject/0
     13 ?        00:00:00 kworker/0:1-events
     14 ?        00:00:00 cpuhp/0
     15 ?        00:00:00 cpuhp/1
     16 ?        00:00:00 idle_inject/1
     17 ?        00:00:00 migration/1
     18 ?        00:00:00 ksoftirqd/1
     20 ?        00:00:00 kworker/1:0H-kblockd
     21 ?        00:00:00 kdevtmpfs
     22 ?        00:00:00 netns
     23 ?        00:00:00 rcu_tasks_kthre
     24 ?        00:00:00 kauditd
     25 ?        00:00:00 khungtaskd
     26 ?        00:00:00 oom_reaper
     27 ?        00:00:00 writeback
     28 ?        00:00:00 kcompactd0
     29 ?        00:00:00 ksmd
     30 ?        00:00:00 khugepaged
     35 ?        00:00:00 kworker/1:1-events
     77 ?        00:00:00 kintegrityd
     78 ?        00:00:00 kblockd
     79 ?        00:00:00 blkcg_punt_bio
     80 ?        00:00:00 tpm_dev_wq
     81 ?        00:00:00 ata_sff
     82 ?        00:00:00 md
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ps -o pid
    PID
   2647
   5577
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$ ps -e |head
    PID TTY          TIME CMD
      1 ?        00:00:08 systemd
      2 ?        00:00:00 kthreadd
      3 ?        00:00:00 rcu_gp
      4 ?        00:00:00 rcu_par_gp
      6 ?        00:00:00 kworker/0:0H-kblockd
      8 ?        00:00:00 mm_percpu_wq
      9 ?        00:00:00 ksoftirqd/0
     10 ?        00:00:01 rcu_sched
     11 ?        00:00:00 migration/0
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~/Desktop$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ whoami
onworks
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ echo "current path is $echo($PAT
H)"
current path is (/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/u
sr/games:/usr/local/games:/snap/bin)
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ echo "current home directory:$ec
ho($HOME)"
current home directory:(/home/onworks)
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ echo $SHELL
/bin/bash
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ uname -a
Linux onworks-Standard-PC-i440FX-PIIX-1996 5.4.0-54-generic #60-Ubuntu SMP Fri N
ov 6 10:37:59 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ uname -r
5.4.0-54-generic
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```

```
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$ ps -eo pid,ppid,%mem,%cpu --sort
=-%mem |head -n 10
    PID    PPID %MEM %CPU
   1052     626 10.4  1.6
   1259     626  6.4  0.8
   1405       1  2.4  0.0
    911     903  2.3  0.0
    912     903  2.0  0.1
   1202    1025  1.9  0.0
    915       1  1.8  0.0
   2019     626  1.8  0.0
    649     647  1.7  0.4
onworks@onworks-Standard-PC-i440FX-PIIX-1996:~$
```