# Operating System (CS 402)

**Lab Report Submitted to**

## Indian Institute of Information Technology Surat

**for**

****

### Bachelor of Technology

**In**

### Computer Science and Engineering Department

**Submitted by**

### JITANSHU J. RAUT UI21CS27

**Course Faculty**

### Shraddha Patel

## Department of Computer Science and Engineering

## Indian Institute of Information Technology Surat

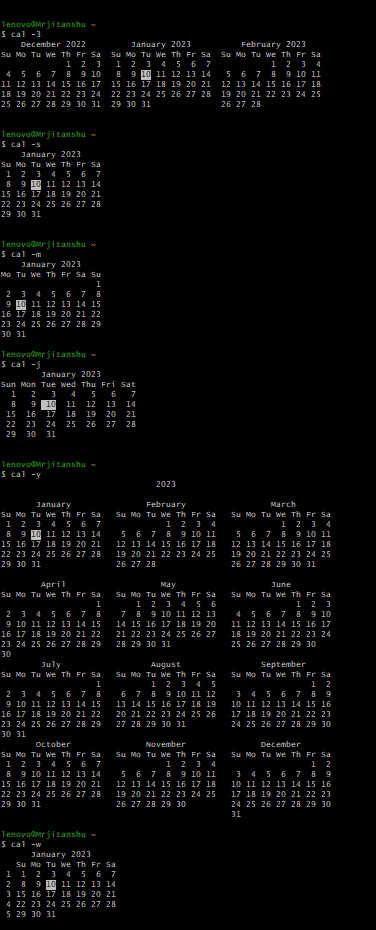
## Gujarat-394190, India

**2023**

# Table of Contents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Exp. No** | **Name of the Experiments** | **Page no** | **Date of Experiment** | **Date of Submission** |
| **1** | **Study of network devices in detail** | **3-14** | 9-1-2023 | 15-1-2023 |
| **2** |  |  |  |  |
| **3** |  |  |  |  |
| **4** |  |  |  |  |
| **5** |  |  |  |  |
| **6** |  |  |  |  |
| **7** |  |  |  |  |
| **8** |  |  |  |  |
| **9** |  |  |  |  |
| **10** |  |  |  |  |
| **11** |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

**Assignment 1**



**CAL**

Option Use

-1 Display single (current) month output. (Thisisthe default.)

-3 Display prev/current/next month output

-s Display Sunday as the first day of the week (This is thedefault.)

-m Display Monday as the first day of the week

-j Display Julian dates (days one-based, numbered fromJanuary 1)

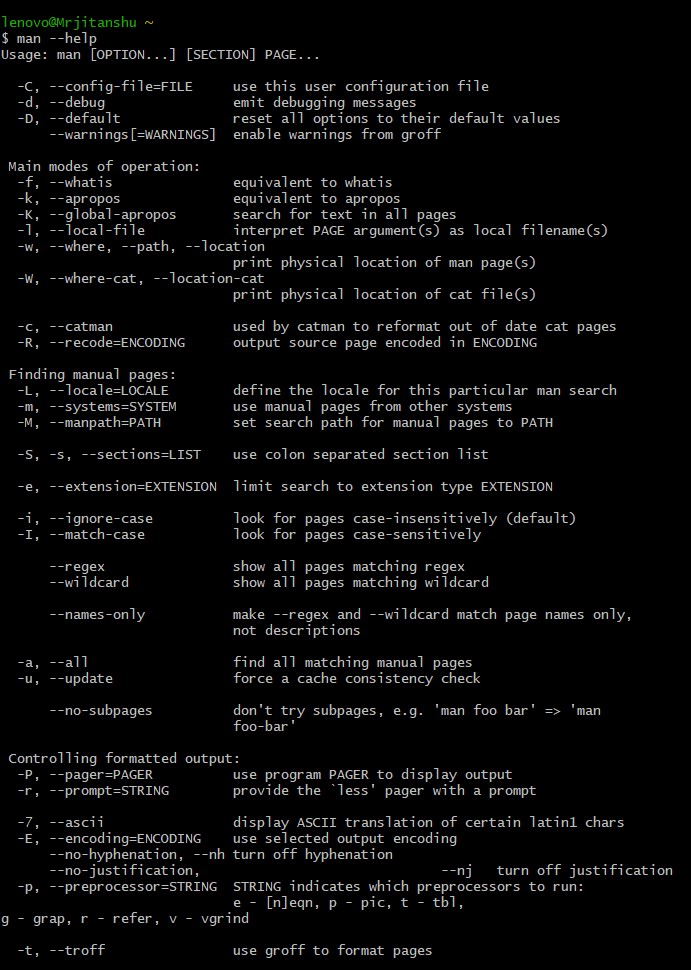
-y Display a calendar for the current year

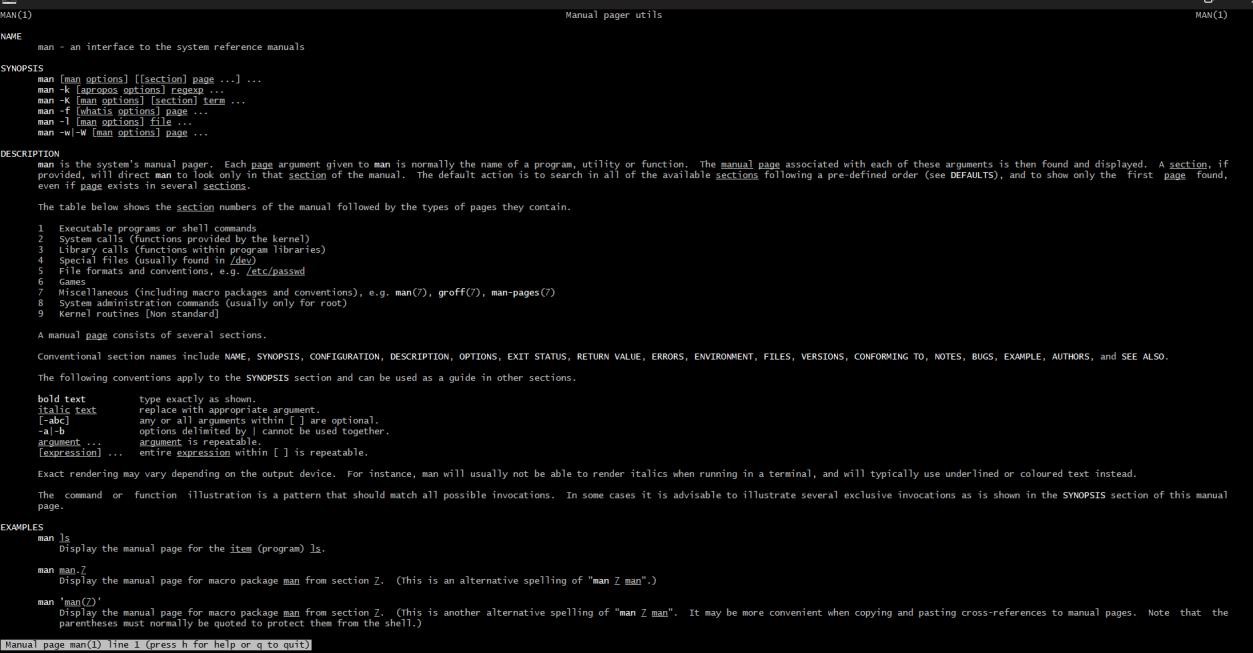
-w Print the number of the week under each week column

**Clear**

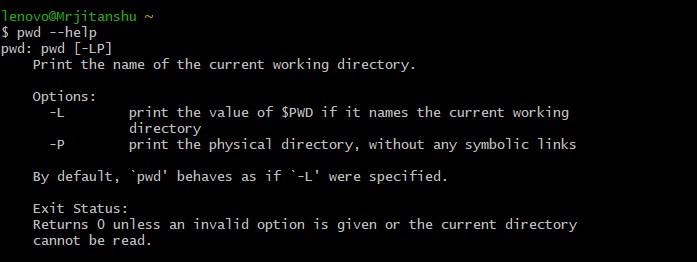


**Man**





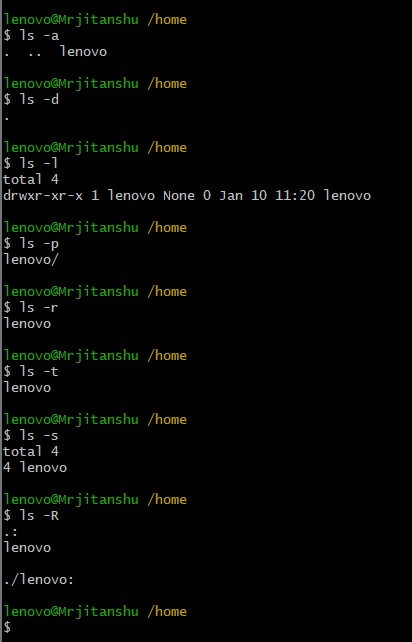
**Pwd**

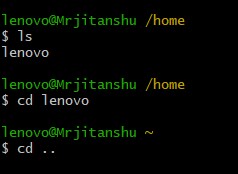


**Cd**



**Ls**



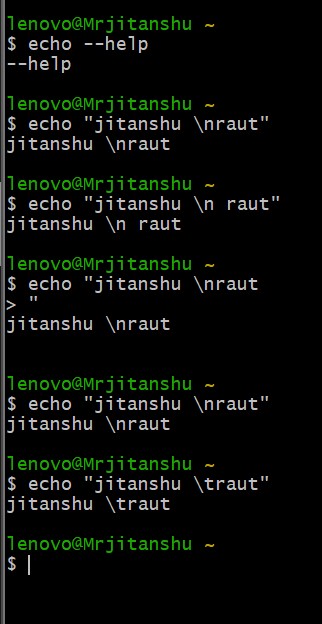
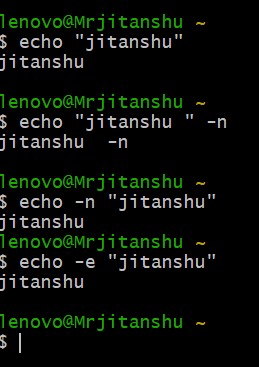


**Exit**

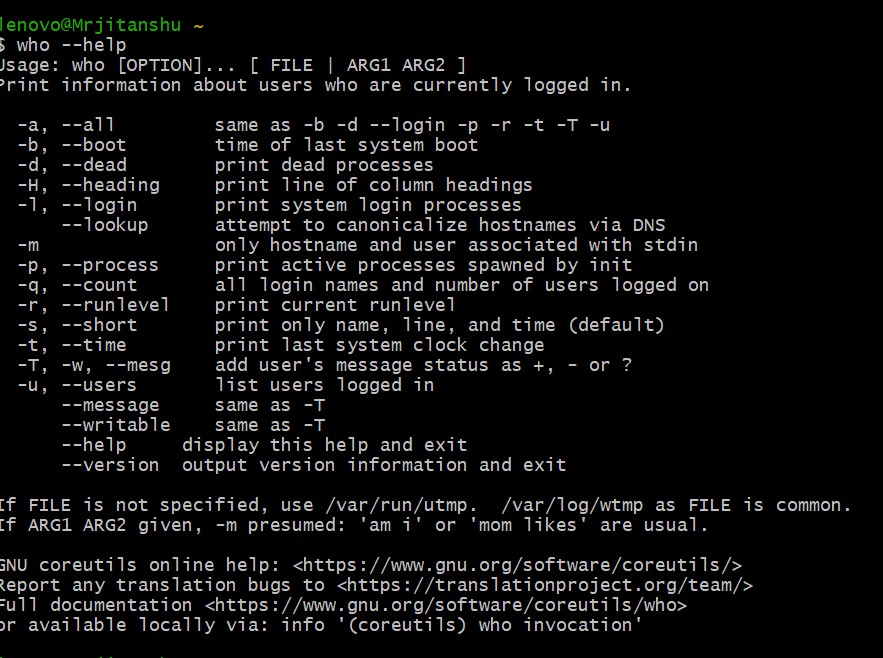
exit :- It is used to terminate a program, shell or log you out of anetwork normally.

Syntax: - exit

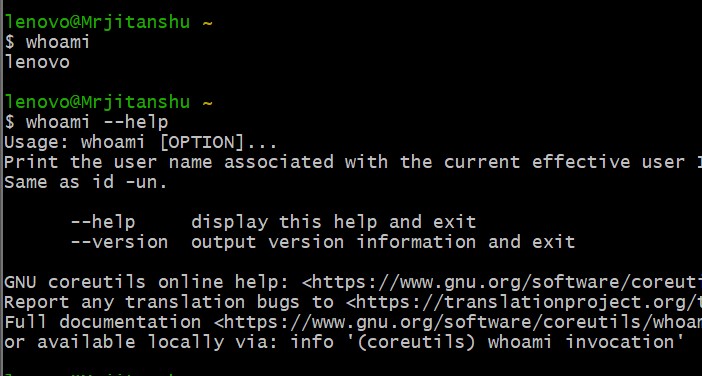
Echo



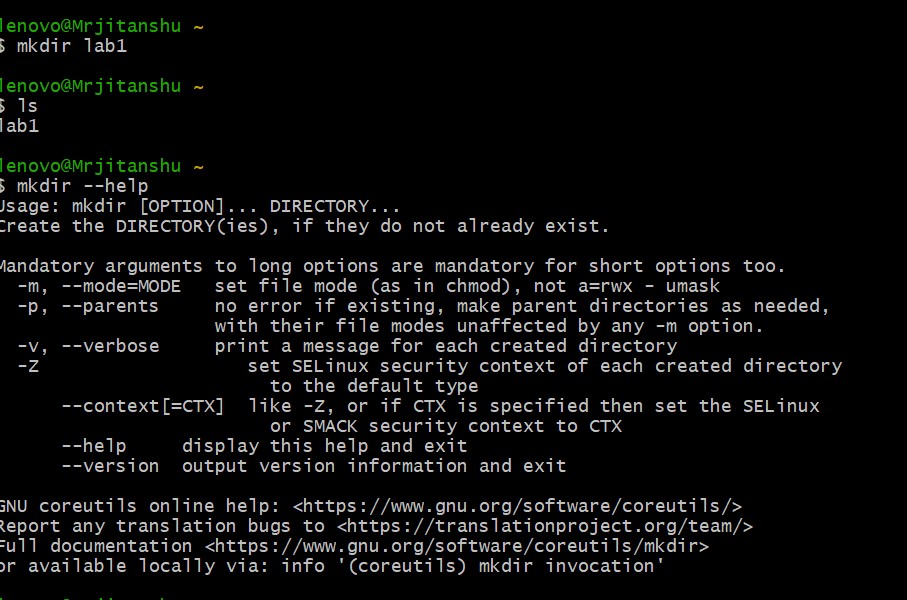
**Who**



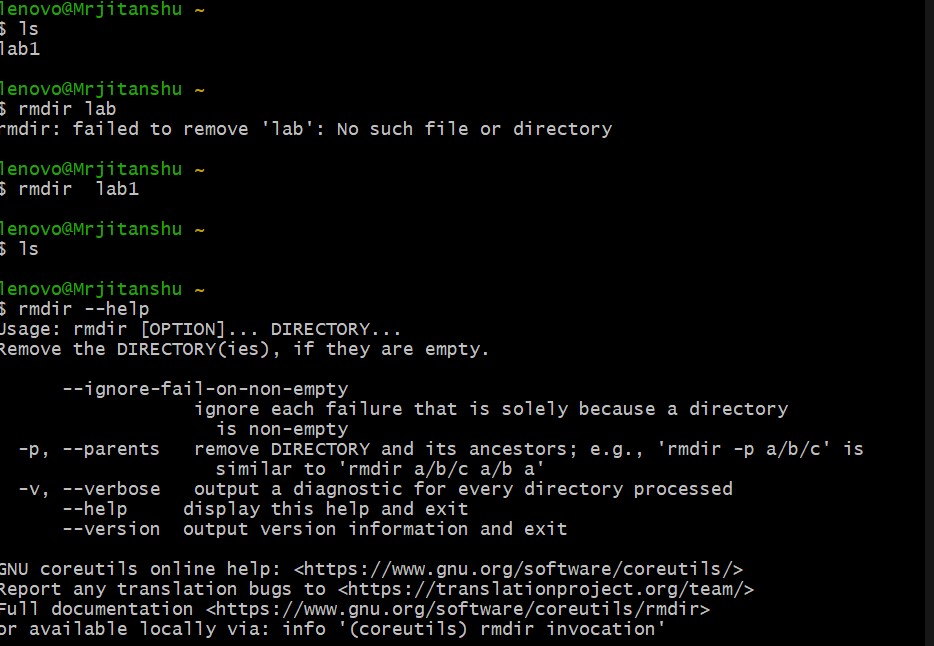
**Whoami**



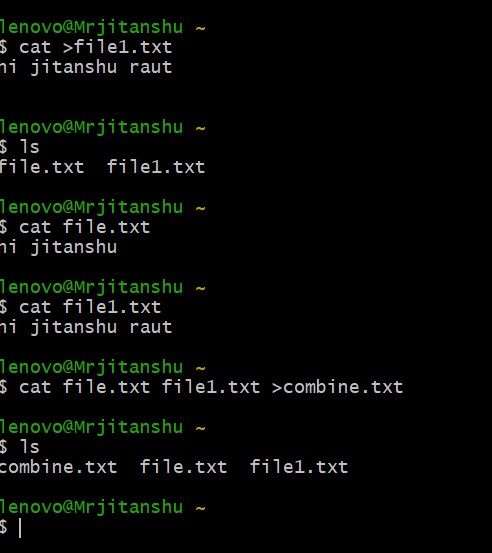
**Mkdir**



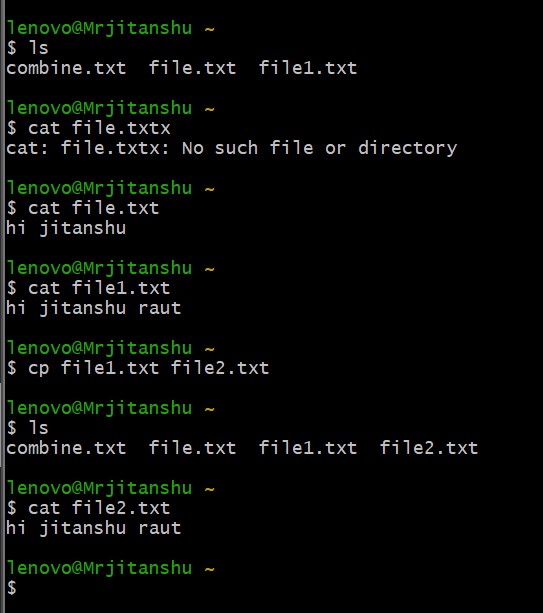
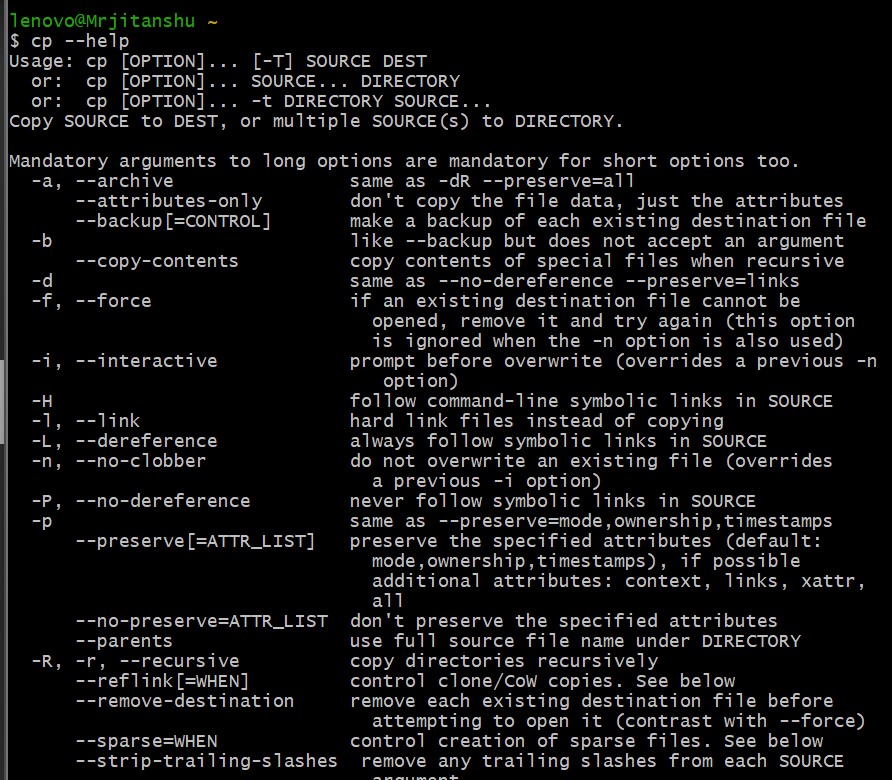
**Rmdir**



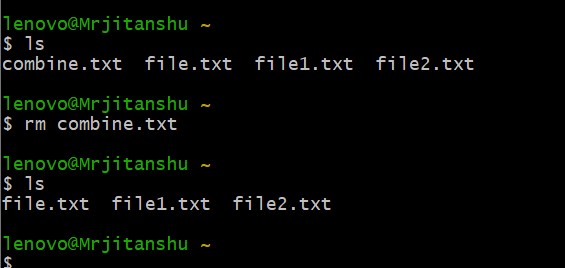
**Cat**



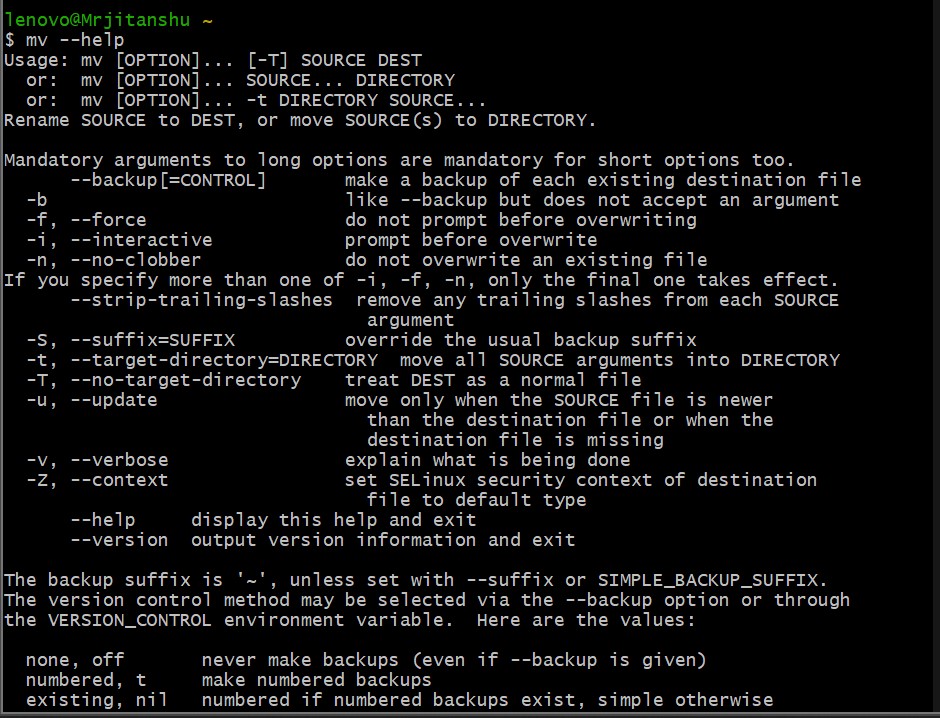
**Cp**



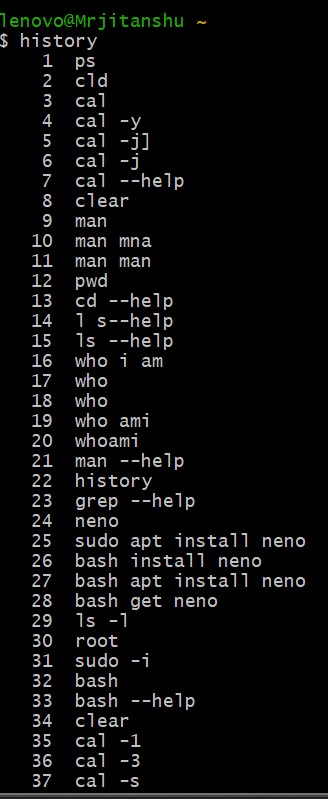
**Rm**



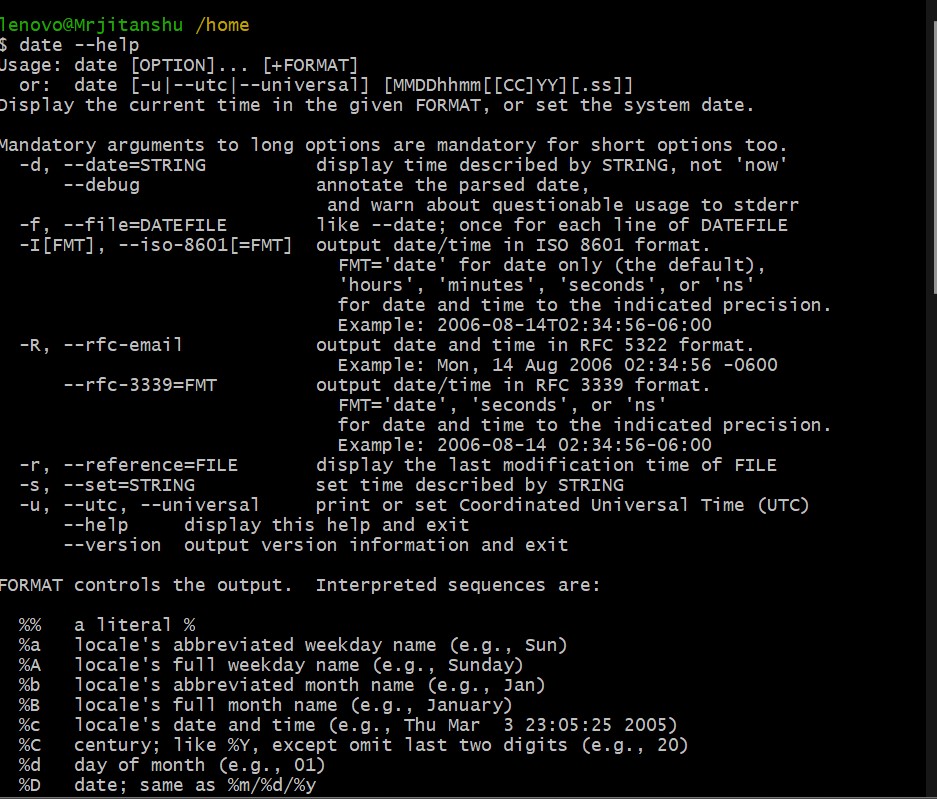
Mv



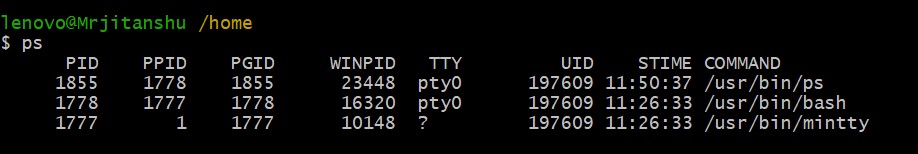
**History**



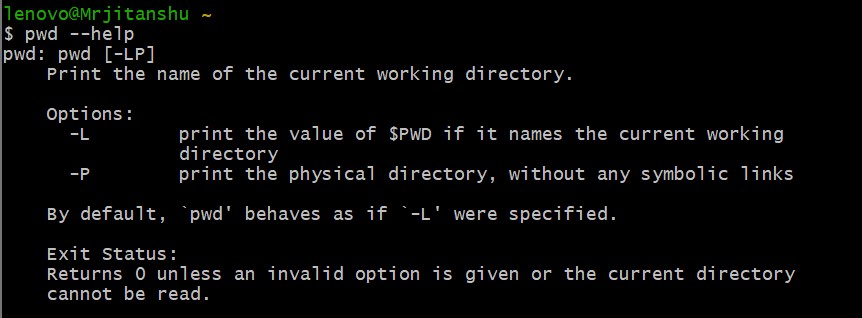
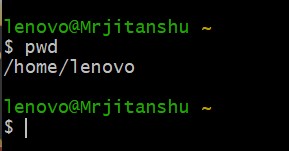
**Date**



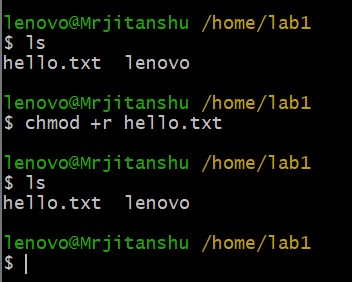
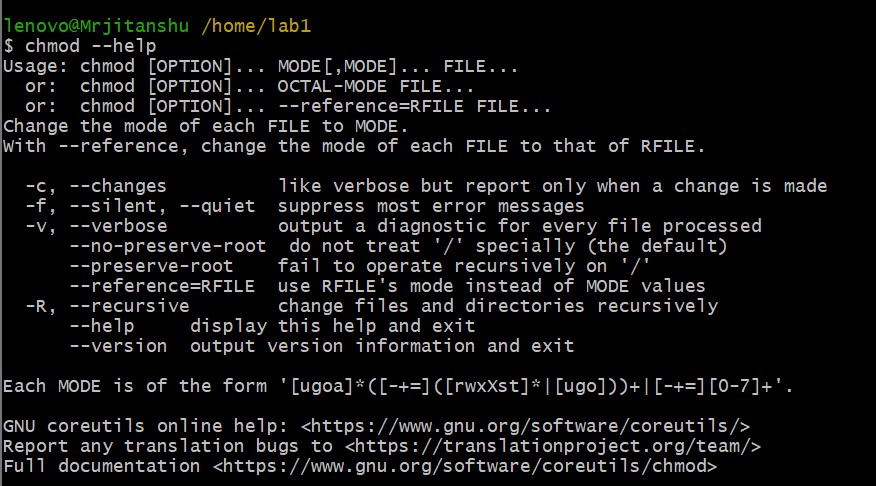
**Ps**



Pwd



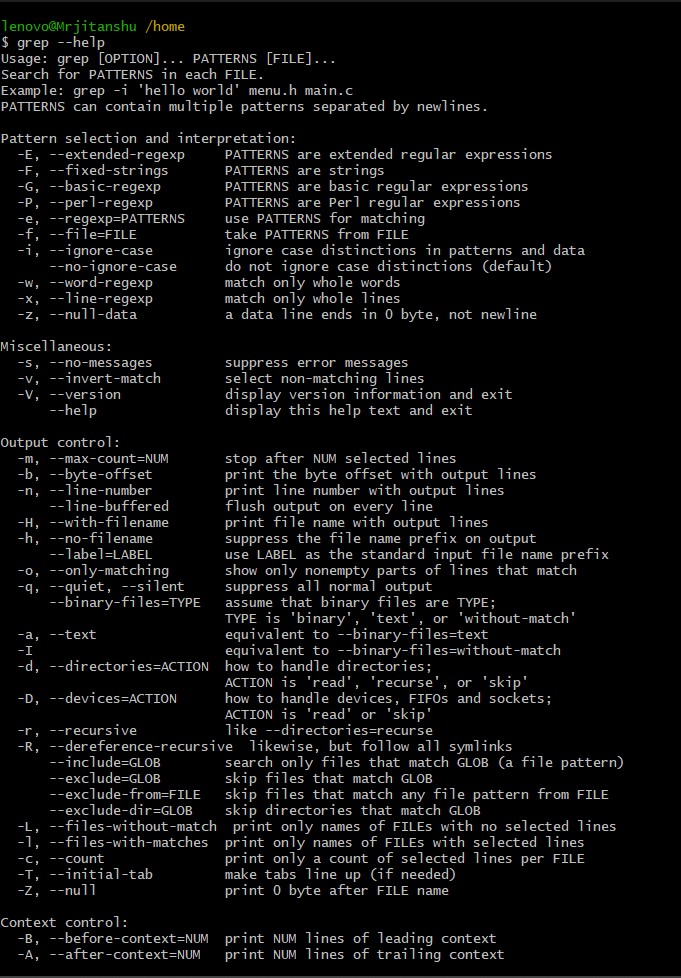
Chmod

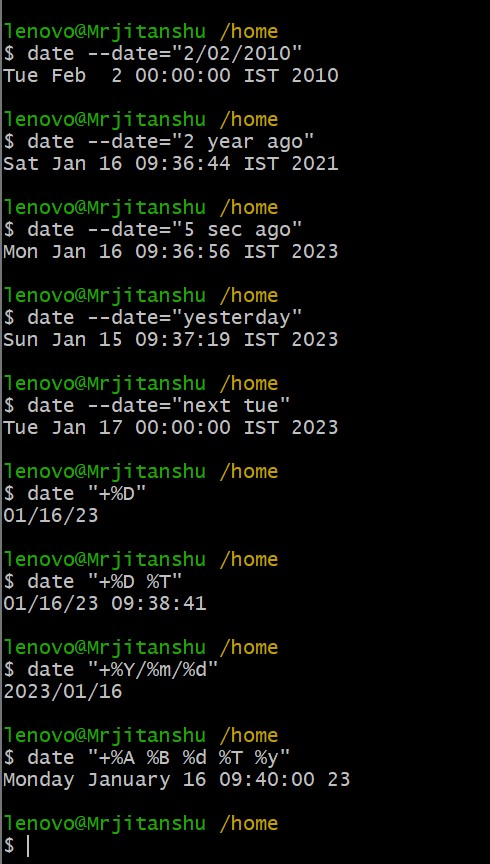
Chown



Grep



Date



**Assignment 2**

**Question-1**

┌──(jitanshu㉿kali)-[~/assigments]

└─$ mkdir COMP

┌──(jitanshu㉿kali)-[~/assigments]

└─$ ls

COMP

┌──(jitanshu㉿kali)-[~/assigments]

└─$ cd COMP

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ LS

LS: command not found

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ ls

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ mkdir Comp1 Comp2 Comp3

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ ls

Comp1 Comp2 Comp3

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ cd Comp1

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1]

└─$ mkdir CPU

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1]

└─$ cd CPU

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ mkdir Control ALU

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ cd Control

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ cat> Comp.c

hello

^C

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ cat>p1.txt

hi

^C

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ cat >p2.txt

hi hello

^C

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ ls

Comp.c p1.txt p2.txt

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ cd /home/jitanshu/assigments/COMP/Comp2

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ ls

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ mkdir In\_out

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ cd ..

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ cd Comp3

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3]

└─$ mkdir Memory

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3]

└─$ cd Memory

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ mkdir RAM ROM

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ cd RAM

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/RAM]

└─$ cat>play.wml

hi

^C

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/RAM]

└─$ cat>play2.wml

hello

^C

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/RAM]

└─$ cd ..

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ cd ROM

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/ROM]

└─$ ls

pict1.jpeg pict2.jpeg

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/ROM]

└─$ cd /home/jitanshu/assigments/COMP

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   └── Control

│   ├── Comp.c

│   ├── p1.txt

│   └── p2.txt

├── Comp2

│   └── In\_out

└── Comp3

└── Memory

├── RAM

│   ├── play2.wml

│   └── play.wml

└── ROM

├── pict1.jpeg

└── pict2.jpeg

10 directories, 7 files

**Question-2**

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2/In\_out]

└─$ cp /home/jitanshu/assigments/COMP/Comp1/CPU/Control/Comp.c /home/jitanshu/assigments/COMP/Comp1/CPU/Control/Computer.c

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   └── p2.txt

├── Comp2

│   └── In\_out

└── Comp3

└── Memory

├── RAM

│   ├── play2.wml

│   └── play.wml

└── ROM

├── pict1.jpeg

└── pict2.jpeg

10 directories, 8 files

**Question-3**

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ cp /home/jitanshu/assigments/COMP/Comp1/CPU/Control/p2.txt /home/jitanshu/assigments/COMP/Comp1/CPU/Control/p3.txt

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ tree

.

├── ALU

└── Control

├── Comp.c

├── Computer.c

├── p1.txt

├── p2.txt

└── p3.txt

2 directories, 5 files

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ cp /home/jitanshu/assigments/COMP/Comp1/CPU/Control/p2.txt /home/jitanshu/assigments/COMP/Comp1/CPU/ALU/p3.txt

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp1/CPU]

└─$ tree

.

├── ALU

│   └── p3.txt

└── Control

├── Comp.c

├── Computer.c

├── p1.txt

├── p2.txt

└── p3.txt

2 directories, 6 files

**Question-4**

─(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ cp /home/jitanshu/assigments/COMP/Comp1/CPU/Control/p1.txt /home/jitanshu/assigments/COMP/Comp3/Memory/ROM/p3.txt

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ tree

.

├── RAM

│   ├── play2.wml

│   └── play.wml

└── ROM

├── p3.txt

├── pict1.jpeg

└── pict2.jpeg

2 directories, 5 files

**Question-5**

(jitanshu㉿kali)-[~/assigments/COMP]

└─$ cp /home/jitanshu/assigments/COMP/Comp3/Memory/ROM/pict1.jpeg /home/jitanshu/assigments/COMP/Comp1/CPU/ALU/pict1.jpeg

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   │   ├── p3.txt

│   │   └── pict1.jpeg

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   ├── p2.txt

│   └── p3.txt

├── Comp2

│   └── In\_out

└── Comp3

└── Memory

├── RAM

│   ├── play2.wml

│   └── play.wml

└── ROM

├── p3.txt

├── pict1.jpeg

└── pict2.jpeg

**Question-6**

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ mv /home/jitanshu/assigments/COMP/Comp3/Memory/RAM/play.wml /home/jitanshu/assigments/COMP/Comp3/Memory/RAM/playlist.wml

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ tree

.

└── In\_out

1 directory, 0 files

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ cd ..

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   │   ├── p3.txt

│   │   └── pict1.jpeg

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   ├── p2.txt

│   └── p3.txt

├── Comp2

│   └── In\_out

└── Comp3

└── Memory

├── RAM

│   ├── play2.wml

│   └── playlist.wml

└── ROM

├── p3.txt

├── pict1.jpeg

└── pict2.jpeg

10 directories, 12 files

**Question-7**

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ find -iname '\*.txt'

./Comp3/Memory/RAM/p3.txt

./Comp1/CPU/Control/p1.txt

./Comp1/CPU/Control/p3.txt

./Comp1/CPU/Control/p2.txt

./Comp1/CPU/ALU/p3.txt

./Comp2/In\_out/p3.txt

**Question-8**

─(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/ALU]

└─$ cp /home/jitanshu/assigments/COMP/Comp1/CPU/ALU/\* /home/jitanshu/assigments/COMP/Comp2/In\_out/

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   │   ├── p3.txt

│   │   └── pict1.jpeg

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   ├── p2.txt

│   └── p3.txt

├── Comp2

│   └── In\_out

│   ├── p3.txt

│   └── pict1.jpeg

└── Comp3

└── Memory

├── RAM

│   ├── play2.wml

│   └── playlist.wml

└── ROM

├── p3.txt

├── pict1.jpeg

└── pict2.jpeg

10 directories, 14 files

**Question-9**

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2/In\_out]

└─$ mv /home/jitanshu/assigments/COMP/Comp3/Memory/ROM/\* /home/jitanshu/assigments/COMP/Comp3/Memory/RAM

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   │   ├── p3.txt

│   │   └── pict1.jpeg

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   ├── p2.txt

│   └── p3.txt

├── Comp2

│   └── In\_out

│   ├── p3.txt

│   └── pict1.jpeg

└── Comp3

└── Memory

├── RAM

│   ├── p3.txt

│   ├── pict1.jpeg

│   ├── pict2.jpeg

│   ├── play2.wml

│   └── playlist.wml

└── ROM

10 directories, 14 files

**Question-10**

┌──(jitanshu㉿kali)-[~/…/COMP/Comp3/Memory/RAM]

└─$ ls [a-z]\*[a-z]\*[a-z]\*[y]\*

play2.wml playlist.wml

**Question-11**

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2/In\_out]

└─$ cp [p]\* /home/jitanshu/assigments/COMP/Comp2/

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2/In\_out]

└─$ cd ..

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp2]

└─$ ls

In\_out p3.txt pict1.jpeg

**Question-12**

──(jitanshu㉿kali)-[~/assigments/COMP/Comp3]

└─$ cd Memory

┌──(jitanshu㉿kali)-[~/assigments/COMP/Comp3/Memory]

└─$ rm /home/jitanshu/assigments/COMP/Comp1/CPU/ALU/\*

zsh: sure you want to delete all 2 files in /home/jitanshu/assigments/COMP/Comp1/CPU/ALU [yn]? Y

**Question-13**

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ rm /home/jitanshu/assigments/COMP/Comp2/In\_out/\*.log

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ tree

.

├── Comp1

│   └── CPU

│   ├── ALU

│   └── Control

│   ├── Comp.c

│   ├── Computer.c

│   ├── p1.txt

│   ├── p2.txt

│   └── p3.txt

├── Comp2

│   ├── In\_out

│   │   ├── p3.txt

│   │   └── pict1.jpeg

│   ├── p3.txt

│   └── pict1.jpeg

└── Comp3

└── Memory

├── RAM

│   ├── p3.txt

│   ├── pict1.jpeg

│   ├── pict2.jpeg

│   ├── play2.wml

│   └── playlist.wml

└── ROM

10 directories, 14 files

**Question-14**

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ ls

Comp.c Computer.c p1.txt p2.txt p3.txt

┌──(jitanshu㉿kali)-[~/…/COMP/Comp1/CPU/Control]

└─$ ls -la

total 28

drwxr-xr-x 2 jitanshu jitanshu 4096 Jan 18 19:13 .

drwxr-xr-x 4 jitanshu jitanshu 4096 Jan 18 19:00 ..

-rw-r--r-- 1 jitanshu jitanshu 6 Jan 18 19:00 Comp.c

-rw-r--r-- 1 jitanshu jitanshu 6 Jan 18 19:10 Computer.c

-rw-r--r-- 1 jitanshu jitanshu 3 Jan 18 19:00 p1.txt

-rw-r--r-- 1 jitanshu jitanshu 9 Jan 18 19:00 p2.txt

-rw-r--r-- 1 jitanshu jitanshu 9 Jan 18 19:13 p3.txt

**Question-15**

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ who

jitanshu tty7 2023-01-23 09:25 (:0)

┌──(jitanshu㉿kali)-[~/assigments/COMP]

└─$ whoami

jitanshu

**Assignment 3**

1. **Print "IIIT Surat" message 5 times using for, while and do--while loop**

┌──(jitanshu㉿kali)-[~]

└─$ a=5

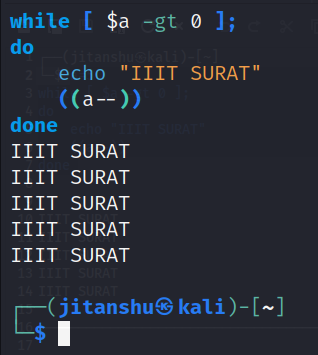
while [ $a -gt 0 ];

do

echo "IIIT SURAT"

((a--))

done

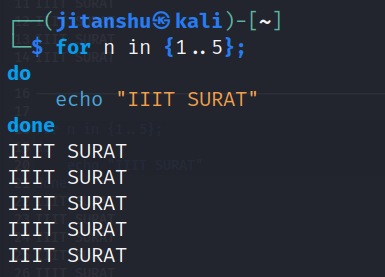


for n in {1..5};

do

echo "IIIT SURAT"

done



1. **Check entered number is even or odd**

echo -n "Enter Number: "

read x

if [ $((x%2)) == 0 ]; then

echo "Number is Even"

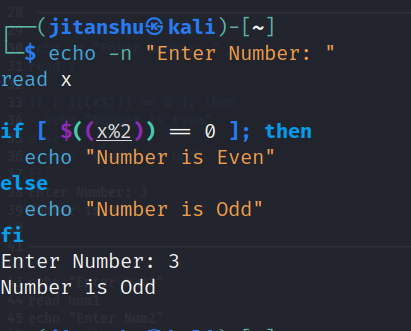
else

echo "Number is Odd"

fi

Enter Number: 3

Number is Odd



1. **Check greatest among three integers on user input**

echo "Enter Num1"

read num1

echo "Enter Num2"

read num2

echo "Enter Num3"

read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]

then

echo "MAX -" $num1

elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]

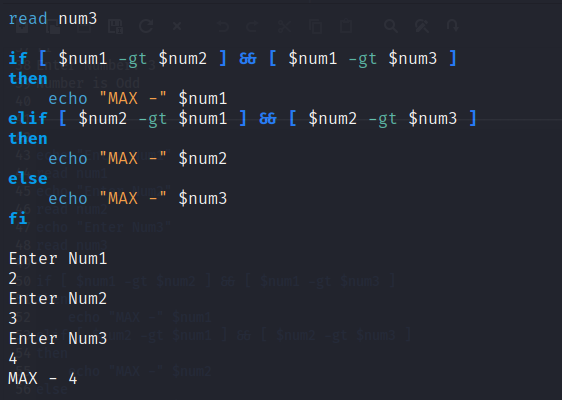
then

echo "MAX -" $num2

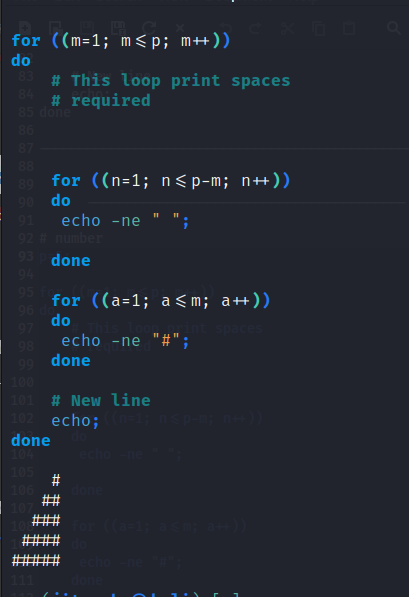
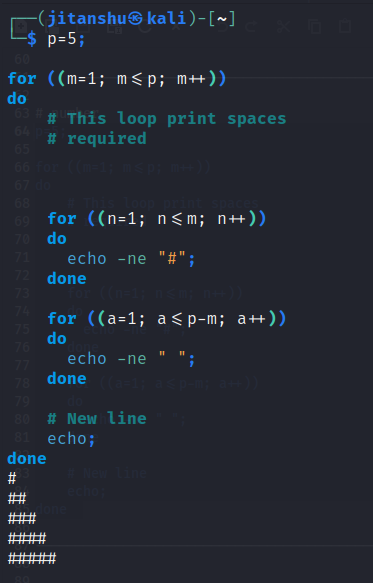
else

echo "MAX -" $num3

fi



5 . **Print the following pattern**



Assignment 4

**Implement a program for memory allocation strategies using switch case or functions as follows:**

**1. Best fit**

**2. First fit**

**3. Worst fit**

**Worst fit**

*#include*<stdio.h>

*#include*<conio.h>

*#define* max 25

void main()

{

int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;

static int bf[max],ff[max];

printf("\n\tMemory Management Scheme - Worst Fit");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of files:");

scanf("%d",&nf);

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

*for*(i=1;i<=nb;i++)

{

printf("Block %d:",i);

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

}

printf("Enter the size of the files :-\n");

*for*(i=1;i<=nf;i++)

{

printf("File %d:",i);

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

}

*for*(i=1;i<=nf;i++)

{

*for*(j=1;j<=nb;j++)

{

*if*(bf[j]!=1) *//if bf[j] is not allocated*

{

temp=b[j]-f[i];

*if*(temp>=0)

*if*(highest<temp)

{

ff[i]=j;

highest=temp;

}

}

frag[i]=highest;

bf[ff[i]]=1;

highest=0;

}

ff[i]=j;

highest=temp;

}

printf("\nFile\_no:\tFile\_size :\tBlock\_no:\tBlock\_size:\tFragement");

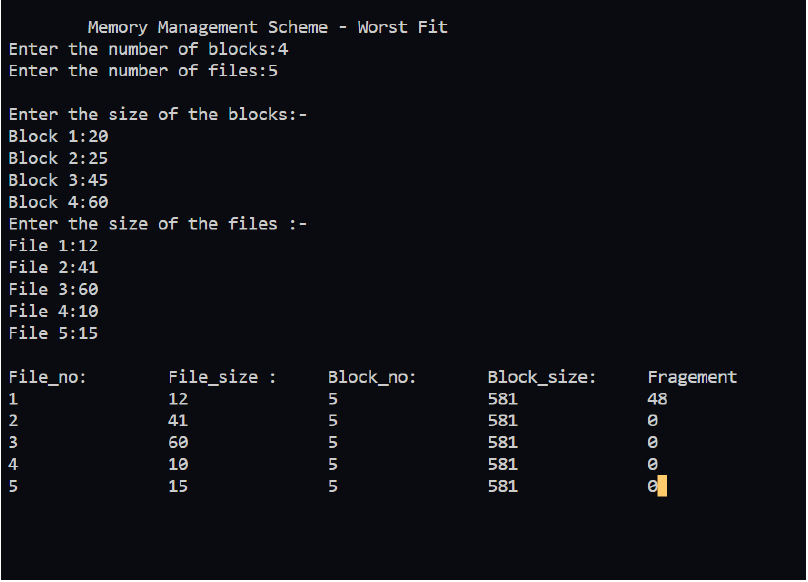
*for*(i=1;i<=nf;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

getch();

}

OUTPUT-----



**First Fit:**

*#include*<stdio.h>

void 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]]);

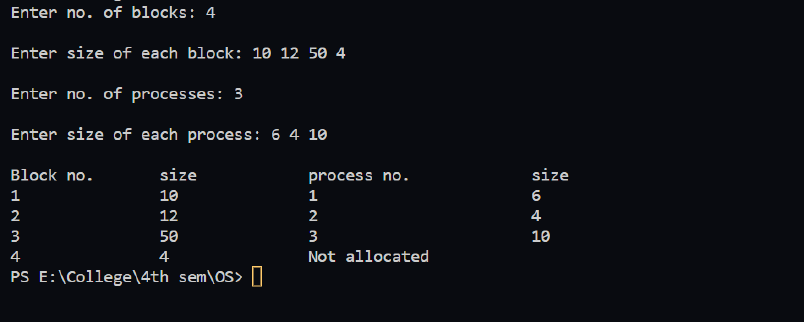
        else

            printf("Not allocated");

    }

}

Output----



**Best Fit:**

*#include*<stdio.h>

void main()

{

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

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

printf("\n\t\t\tMemory Management Scheme - Best Fit");

printf("\nEnter the number of blocks:");

scanf("%d",&nb);

printf("Enter the number of processes:");

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");

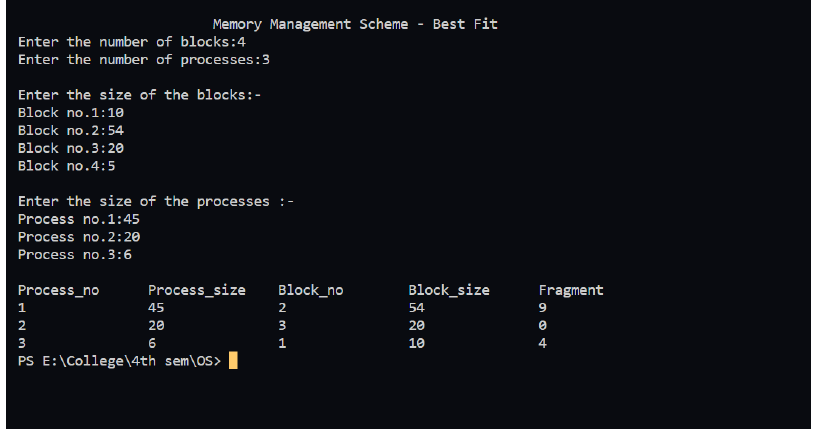
*for*(i=1;i<=np && parray[i]!=0;i++)

printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,p[i],parray[i],b[parray[i]],fragme

nt[i]);

}

Output----



Assignment 5

Implement a CPU scheduling algorithms using switch case (in single program file)

1. FCFS

2. SJF

3. SRTF

4. Round Robin

5. Priority

Code:

#include <bits/stdc++.h>

using namespace std;

void print(vector<int> v)

{

string space(1, ' ');

for (auto i : v)

cout << i << space << endl;

}

bool sortcol(vector<int> a, vector<int> b)

{

return a[1] < b[1];

}

void fcfs(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n);

sort(v.begin(), v.end(), sortcol);

int c = v[0][0];

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

{

c += v[i][1];

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

}

cout << endl;

cout << "Process No.\t"

<< "AT\t\t"

<< "BT\t\t"

<< "CT\t\t"

<< "TAT\t\t"

<< "WT\t\t" << endl;

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

{

cout << v[i][2] << "\t\t" << v[i][0] << "\t\t" << v[i][1] << "\t\t" <<

ct[i] << "\t\t" << turn[i] << "\t\t" << wait[i] << endl;

}

}

void sjf(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n);

int c = 0;

int i;

sort(v.begin(), v.end());

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,

int>>> pq;

pq.push({v[0][1], 0});

map<int, int> m;

while (!pq.empty())

{

i = pq.top().second;

// cout<<i<<endl;

m[i] = 1;

c += v[i][1];

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

pq.pop();

if (m.size() != n)

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

if (v[j][0] <= c && !m[j])

pq.push({v[j][1], j});

else

break;

}

cout << "Process No.\t"

<< "AT\t\t"

<< "BT\t\t"

<< "CT\t\t"

<< "TAT\t\t"

<< "WT\t\t" << endl;

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

{

cout << v[i][2] << "\t\t" << v[i][0] << "\t\t" << v[i][1] << "\t\t" <<

ct[i] << "\t\t" << turn[i] << "\t\t" << wait[i] << endl;

}

}

void srtf(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n), rem\_time(n);

int i, c = 0, count = 0;

sort(v.begin(), v.end());

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

rem\_time[i] = v[i][1];

priority\_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int,

int>>> pq;

while (count != n)

{

for (int j = 0; j < n; j++)

if (v[j][0] <= c && rem\_time[j] > 0)

pq.push({rem\_time[j], j});

i = pq.top().second;

pq.pop();

if ((rem\_time[i] - 1) < 0)

continue;

c++;

rem\_time[i]--;

if (!rem\_time[i])

{

count++;

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

}

}

cout << "Process No.\t"

<< "AT\t\t"

<< "BT\t\t"

<< "CT\t\t"

<< "TAT\t\t"

<< "WT\t\t" << endl;

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

cout << v[i][2] << "\t\t" << v[i][0] << "\t\t" << v[i][1] << "\t\t" <<

ct[i] << "\t\t" << turn[i] << "\t\t" << wait[i] << endl;

}

void prnp(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n);

int c = 0;

int i;

cout << "Enter Priorities : ";

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

cin >> v[i][3];

sort(v.begin(), v.end());

priority\_queue<pair<int, int>, vector<pair<int, int>>,

function<bool(pair<int, int>, pair<int, int>)>> pq =

priority\_queue<pair<int, int>, vector<pair<int, int>>,

function<bool(pair<int, int>, pair<int, int>)>>(

[](pair<int, int> a, pair<int, int> b)

{

if (a.first != b.first)

{

return a.first < b.first;

}

else

{

return a.second > b.second;

}

});

// priority\_queue<pii>pq;

vector<pair<int, int>> vv;

// Stores BT and index of vec have<int>g that BT after Sorting

pq.push({v[0][3], 0});

map<int, int> m, vis;

while (!pq.empty())

{

i = pq.top().second;

// cout<<i+1<<endl;

m[i] = 1;

c += v[i][1];

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

pq.pop();

if (vis.size() != n)

{

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

if (v[j][0] <= c && !vis[j])

{

pq.push({v[j][3], j});

vis[j] = 1;

}

else

break;

}

}

cout << "Process No.\t"

<< "AT\t"

<< "BT\t"

<< "Priority\t"

<< "CT\t"

<< "TAT\t"

<< "WT\t" << endl;

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

{

cout << v[i][2] << "\t\t" << v[i][0] << "\t" << v[i][1] << "\t" <<

v[i][3] << "\t\t" << ct[i] << "\t" << turn[i] << "\

t" << wait[i] << endl;

}

}

void prp(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n), rem\_time(n);

int c = 0;

int i;

cout << "Enter Priorities : ";

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

cin >> v[i][3];

sort(v.begin(), v.end());

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

rem\_time[i] = v[i][1];

priority\_queue<pair<int, int>, vector<pair<int, int>>,

function<bool(pair<int, int>, pair<int, int>)>> pq =

priority\_queue<pair<int, int>, vector<pair<int, int>>,

function<bool(pair<int, int>, pair<int, int>)>>(

[](pair<int, int> a, pair<int, int> b)

{

if (a.first != b.first)

{

return a.first < b.first;

}

else

{

return a.second > b.second;

}

});

// pq.push({v[0][3],0});

map<int, int> m, vis;

int count = 0;

while (count != n)

{

for (int j = 0; j < n; j++)

if (v[j][0] <= c && rem\_time[j] > 0)

pq.push({v[j][3], j});

i = pq.top().second;

pq.pop();

if ((rem\_time[i] - 1) < 0)

continue;

c++;

rem\_time[i]--;

if (!rem\_time[i])

{

count++;

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

}

}

cout << "Process No.\t"

<< "AT\t"

<< "BT\t"

<< "Priority\t"

<< "CT\t"

<< "TAT\t"

<< "WT\t" << endl;

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

{

cout << v[i][2] << "\t\t" << v[i][0] << "\t" << v[i][1] << "\t" <<

v[i][3] << "\t\t" << ct[i] << "\t" << turn[i] << "\

t" << wait[i] << endl;

}

}

void rr(vector<vector<int>> v)

{

int n = v.size();

vector<int> ct(n), turn(n), wait(n), rem\_time(n);

int i, quan, c = 0;

cout << "Enter the quantum for round robin: ";

cin >> quan;

sort(v.begin(), v.end());

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

rem\_time[i] = v[i][1];

queue<pair<int, int>> pq;

map<int, int> vis;

int count = 0;

pq.push({rem\_time[0], 0});

vis[0] = 1;

while (count != n)

{

i = pq.front().second;

pq.pop();

if (rem\_time[i] > quan)

{

c += quan;

rem\_time[i] -= quan;

}

else

{

c += rem\_time[i];

rem\_time[i] = 0;

}

for (int j = 0; j < n; j++)

if (v[j][0] <= c && rem\_time[j] > 0 && !vis[j])

{

pq.push({rem\_time[j], j});

vis[j] = 1;

}

if (rem\_time[i])

pq.push({rem\_time[i], i});

queue<pair<int, int>> qq = pq;

if (!rem\_time[i])

{

count++;

ct[i] = c;

turn[i] = ct[i] - v[i][0];

wait[i] = turn[i] - v[i][1];

}

}

cout << "Process No.\t"

<< "AT\t"

<< "BT\t"

<< "CT\t"

<< "TAT\t"

<< "WT\t" << endl;

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

{

cout << v[i][2] << "\t\t" << v[i][0] << "\t" << v[i][1] << "\t" <<

ct[i] << "\t" << turn[i] << "\t" << wait[i] << endl;

}

}

int main()

{

cout << " Enter Number of Processes : ";

int n;

cin >> n;

vector<vector<int>> ab(n, vector<int>(4));

cout << "Enter ArrivalTime & Burst Time: ";

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

{

cin >> ab[i][0] >> ab[i][1];

ab[i][2] = i + 1;

}

while (1)

{

cout<<"Enter 0(Exit),1(FCFS),2(SJF),3(SRTF),4(Preemptive Priority),5(Non-

Preemptive Priority),6(Round Robin) : ";

int whichalgo;

cin >> whichalgo;

switch (whichalgo)

{

case 1:

{

fcfs(ab);

break;

}

case 2:

{

sjf(ab);

break;

}

case 3:

{

srtf(ab);

break;

}

case 4:

{

prp(ab);

break;

}

case 5:

{

prnp(ab);

break;

}

case 6:

{

rr(ab);

break;

}

case 0:

{

exit(0);

break;

}

default:

{

cout << "Enter Valid Number for Algorithm";

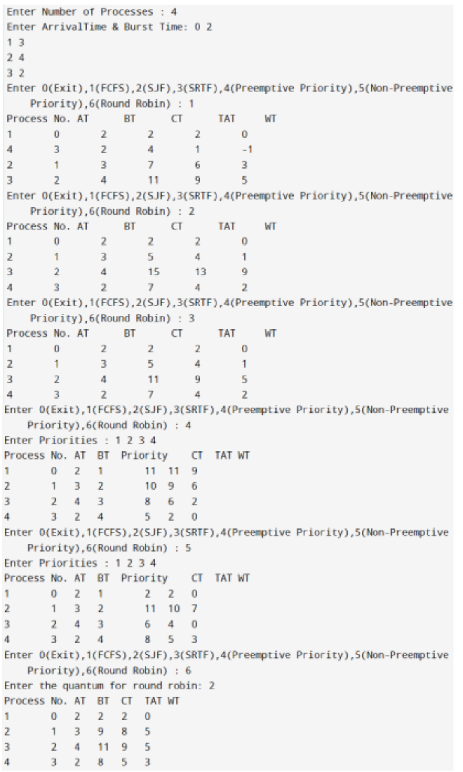
}

}

}

}

Output:



Assignment 6

Implement a program for memory management technique PAGING

Code:

#include "stdio.h"

#include "stdbool.h"

#include "stdlib.h"

struct page{

int page\_no;

int frame;

};

int main()

{

int size\_logical\_address,size\_physical\_address,i,size\_of\_page,j;

printf("Enter size of logical address space: ");

scanf("%d",&size\_logical\_address);

printf("Enter size of physical address space: ");

scanf("%d",&size\_physical\_address);

printf("Enter size of page: ");

scanf("%d",&size\_of\_page);

int number\_of\_frames = size\_physical\_address/size\_of\_page;

int number\_of\_pages = size\_logical\_address/size\_of\_page;

struct page pageTable[number\_of\_pages];

printf("Enter page table: \n");

for(i=0;i<number\_of\_pages;i++)

{

pageTable[i].frame = -1;

}

for(i=0;i<number\_of\_pages;i++)

{

int frame;

bool replica = false;

pageTable[i].page\_no = i;

printf("Enter frame for %d page number(-1 if frame doesn't exist):

",i);

scanf("%d",&frame);

for(j=0;j<number\_of\_pages;j++)

{

if(frame!= -1 && pageTable[j].frame == frame)

{

replica = true;

printf("Frame number already stored\n");

}

}

if(frame > number\_of\_frames)

{

replica = true;

printf("Cannot exceed frame size\n");

}

if(replica == false)

{

pageTable[i].frame = frame;

}

}

int logical\_address;

printf("Enter -1 to exit\n");

while(1)

{

printf("Enter logical address: ");

scanf("%d",&logical\_address);

if(logical\_address == -1)

return 0;

int page\_no = logical\_address/size\_of\_page;

int offset = logical\_address%size\_of\_page;

if(pageTable[page\_no].frame == -1)

{

printf("No such logical address exist\n");

}

else

{

printf("Page no: %d \nOffset: %d\nFrame no: %d\nPhysical address:

%d\n",page\_no, offset,

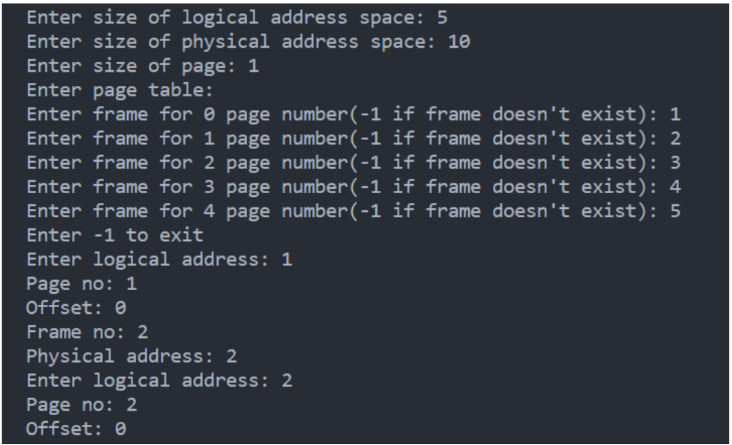
pageTable[page\_no].frame,pageTable[page\_no].frame\*size\_of\_page + offset );

}

}

}

Output:



Assignment 7

Implement a program for readers writers, producer consumer and printer spooler using semaphore (use switch case or function)

CODE:

import random

import threading

import time

import sys

print("0. Exit")

print("1. Reader Writer")

print("2. Producer Consumer")

print("3. Printer Spooler")

choice = int(input("Enter your choice: "))

if choice == 0:

sys.exit()

elif choice == 1:

class ReaderWriter():

def \_\_init\_\_(self):

# initializing semaphores using Semaphore class in threading module for reading and wrting

self.rd = threading.Semaphore()

self.wrt = threading.Semaphore()

self.readCount = 0 # initializing number of reader present

def reader(self):

while True:

self.rd.acquire() # wait on read semaphore

self.readCount += 1 # increase count for reader by 1

if self.readCount == 1: # since reader is present, prevent writing on data

self.wrt.acquire() # wait on write semaphore

self.rd.release() # sinal on read semaphore

print(f"Reader {self.readCount} is reading")

self.rd.acquire() # wait on read semaphore

self.readCount -= 1 # reading performed by reader hence decrementing readercount

if self.readCount == 0: # if no reader is present allow writer to write the data

self.wrt.release() # signal on write semphore, now writer can write

self.rd.release() # sinal on read semaphore

time.sleep(3)

def writer(self):

while True:

self.wrt.acquire() # wait on write semaphore

print("Wrting data.....") # write the data

print("-"\*20)

self.wrt.release() # sinal on write semaphore

time.sleep(3)

def main(self):

# calling mutliple readers and writers

t1 = threading.Thread(target=self.reader)

t1.start()

t2 = threading.Thread(target=self.writer)

t2.start()

t3 = threading.Thread(target=self.reader)

t3.start()

t4 = threading.Thread(target=self.reader)

t4.start()

t6 = threading.Thread(target=self.writer)

t6.start()

t5 = threading.Thread(target=self.reader)

t5.start()

if \_\_name\_\_ == "\_\_main\_\_":

c = ReaderWriter()

c.main()

elif choice == 2:

# Shared Memory variables

CAPACITY = 10

buffer = [-1 for i in range(CAPACITY)]

in\_index = 0

out\_index = 0

# Declaring Semaphores

mutex = threading.Semaphore()

empty = threading.Semaphore(CAPACITY)

full = threading.Semaphore(0)

# Producer Thread Class

class Producer(threading.Thread):

def run(self):

global CAPACITY, buffer, in\_index, out\_index

global mutex, empty, full

items\_produced = 0

counter = 0

while items\_produced < 20:

empty.acquire()

mutex.acquire()

counter += 1

buffer[in\_index] = counter

in\_index = (in\_index + 1) % CAPACITY

print("Producer produced : ", counter)

mutex.release()

full.release()

time.sleep(1)

items\_produced += 1

# Consumer Thread Class

class Consumer(threading.Thread):

def run(self):

global CAPACITY, buffer, in\_index, out\_index, counter

global mutex, empty, full

items\_consumed = 0

while items\_consumed < 20:

full.acquire()

mutex.acquire()

item = buffer[out\_index]

out\_index = (out\_index + 1) % CAPACITY

print("Consumer consumed item : ", item)

mutex.release()

empty.release()

time.sleep(2.5)

items\_consumed += 1

# Creating Threads

producer = Producer()

consumer = Consumer()

# Starting Threads

consumer.start()

producer.start()

# Waiting for threads to complete

producer.join()

consumer.join()

elif choice==3:

# Define the maximum number of jobs that can be queued

MAX\_JOBS = 5

# Define the shared job queue

job\_queue = []

# Define a semaphore for controlling access to the job queue

job\_queue\_mutex = threading.Semaphore(1)

# Define a semaphore for indicating that the job queue is not empty

job\_queue\_not\_empty = threading.Semaphore(0)

# Define a semaphore for indicating that the job queue is not full

job\_queue\_not\_full = threading.Semaphore(MAX\_JOBS)

# Define the printer thread function

def printer\_thread():

while True:

# Acquire the job queue not empty semaphore to wait for a job to be added to the queue

job\_queue\_not\_empty.acquire()

# Acquire the job queue mutex to remove a job from the queue

job\_queue\_mutex.acquire()

# Remove the first job from the queue

job = job\_queue.pop(0)

# Release the job queue mutex

job\_queue\_mutex.release()

# Release the job queue not full semaphore to signal that there is now room for another job

job\_queue\_not\_full.release()

# Print the job

print(f"Printing job {job}")

time.sleep(1) # Simulate the time it takes to print the job

# Define the user thread function

def user\_thread():

global job\_queue

# Generate a job ID (in this case, the current time as an integer)

job\_id = random.randint(1, 50)

# Acquire the job queue not full semaphore to wait for space to add the job to the queue

job\_queue\_not\_full.acquire()

# Acquire the job queue mutex to add the job to the queue

job\_queue\_mutex.acquire()

# Add the job ID to the end of the queue

job\_queue.append(job\_id)

# Release the job queue mutex

job\_queue\_mutex.release()

# Release the job queue not empty semaphore to signal that there is now a job in the queue

job\_queue\_not\_empty.release()

print(f"Added job {job\_id} to the queue.")

# Create a printer thread and start it

printer = threading.Thread(target=printer\_thread, name="Printer")

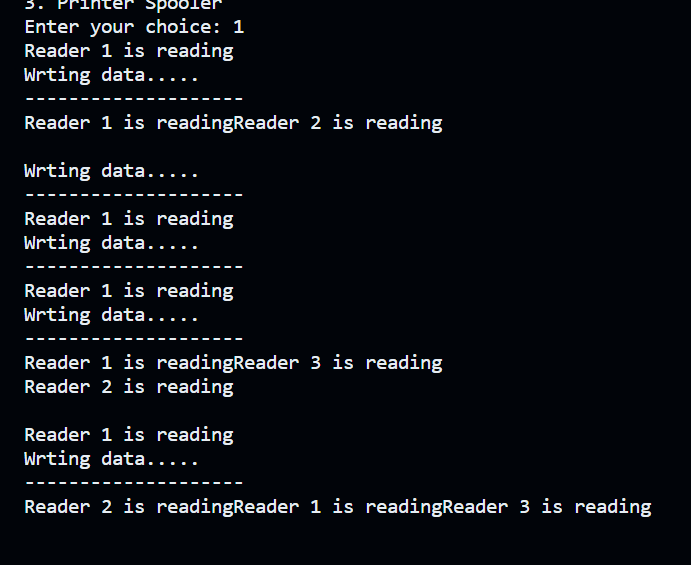
printer.start()

# Create some user threads to add jobs to the queue

for i in range(10):

threading.Thread(target=user\_thread, name=f"User {i+1}").start()

OUTPUT:



Assignment 8

Implement a code for Banker's Algorithm for Deadlock Avoidance and check for user input system is deadlock free or not.

CODE:

#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

int need[100][100],allot[100][100],max[100][100],available[100];

bool isFinished[100];

int sequence[100];

void isSafe(int N,int M)

{

int i,j,work[100],count=0;

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

work[i]=available[i];

for(i=0;i<100;i++)

isFinished[i]=false;

while(count<N)

{

bool canAllot=false;

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

{

if(isFinished[i]==false)

{

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

{

if(work[j]<need[i][j])

{

break;

}

}

if(j==M)

{

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

{

work[j]+=allot[i][j];

}

sequence[count++]=i;

isFinished[i]=true;

canAllot=true;

}

}

}

if(canAllot==false)

{

printf("System Is not safe\n");

return ;

}

}

printf("System is in safe state\n");

printf("Safe sequence :");

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

printf("%d ",sequence[i]);

printf("\n");

}

int main()

{

int i,j,N,M;

printf("Enter the number of process and resources :");

scanf("%d %d",&N,&M);

printf("Enter the available resources :\n");

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

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

printf("Enter the Allocation Matrix :\n");

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

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

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

printf("Enter the matrix for maximum demand of each process :\n");

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

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

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

//calculation of need matrix

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

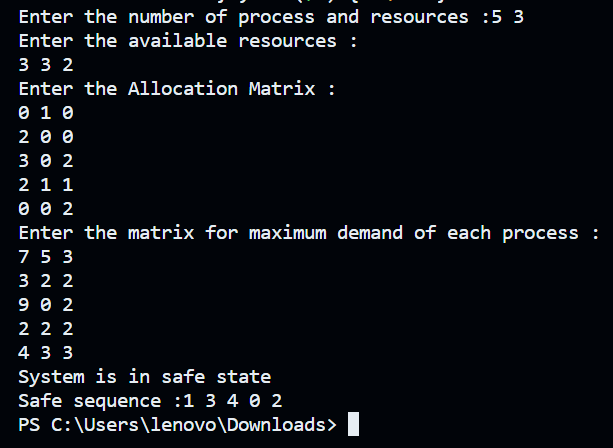
for(j=0;j<M;j++)

need[i][j]=max[i][j]-allot[i][j];

isSafe(N,M);

}

OUTPUT:



Assignment 10

Implement a program for disk scheduling algorithm and state which is better for number of head movements

1. FCFS

2. SSTF

3. SCAN

4. C-SCAN

5. LOOK

6. C-LOOK

Code:

/\*

Disk Scheduling

C-Look

\*/

#include "stdio.h"

#include "stdlib.h"

#include "stdbool.h"

struct request

{

int request\_track\_number;

bool visited;

};

int clook()

{

int i, no\_of\_requests, initial\_head, limit, j, choice, previous\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

struct request req[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

scanf("%d", &req[i].request\_track\_number);

req[i].visited = false;

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

printf("Enter the previous position of R/W head: ");

scanf("%d", &previous\_head);

printf("Enter the cylinder size: ");

scanf("%d", &limit);

if (previous\_head - initial\_head > 0)

{

choice = 2;

}

else

choice = 1;

// scanf("%d",&choice);

int seek\_time = 0;

printf("%d -> ", initial\_head);

int cp\_initial\_head = initial\_head;

if (choice == 1)

{

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

initial\_head = 0;

for (i = 0; i < cp\_initial\_head; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

else if (choice == 2)

{

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

initial\_head = limit - 1;

for (i = limit; i > cp\_initial\_head; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

printf("Seek Time: %d\n", seek\_time);

}

int sstf()

{

int i, no\_of\_requests, initial\_head, limit, j, choice, previous\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

struct request req[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

scanf("%d", &req[i].request\_track\_number);

req[i].visited = false;

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

int seek\_time = 0;

printf("%d -> ", initial\_head);

int n = no\_of\_requests;

while (n)

{

int min = 1e9;

int min\_track\_number, position;

for (i = 0; i < no\_of\_requests; i++)

{

if (abs(initial\_head - req[i].request\_track\_number) < min && req[i].visited == false)

{

min = abs(initial\_head - req[i].request\_track\_number);

min\_track\_number = req[i].request\_track\_number;

position = i;

}

}

initial\_head = req[position].request\_track\_number;

req[position].visited = true;

printf("%d ->", min\_track\_number);

seek\_time += min;

n--;

}

printf("\nSeek Time: %d\n", seek\_time);

}

int scan()

{

int i, no\_of\_requests, initial\_head, limit, j, choice, previous\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

struct request req[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

scanf("%d", &req[i].request\_track\_number);

req[i].visited = false;

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

printf("Enter the previous position of R/W head: ");

scanf("%d", &previous\_head);

printf("Enter the cylinder size: ");

scanf("%d", &limit);

if (previous\_head - initial\_head > 0)

{

choice = 2;

}

else

choice = 1;

// scanf("%d",&choice);

int seek\_time = 0;

printf("%d -> ", initial\_head);

if (choice == 1)

{

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("%d -> ", limit - 1);

seek\_time += abs(limit - 1 - initial\_head);

initial\_head = limit - 1;

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

seek\_time += abs(initial\_head - 0);

printf("0 \n");

}

else if (choice == 2)

{

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("%d -> ", 0);

seek\_time += abs(0 - initial\_head);

initial\_head = 0;

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

seek\_time += abs(limit - 1 - initial\_head);

printf("%d \n", limit - 1);

}

printf("Seek Time: %d\n", seek\_time);

}

int cscan()

{

int i, no\_of\_requests, initial\_head, limit, j, choice, previous\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

struct request req[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

scanf("%d", &req[i].request\_track\_number);

req[i].visited = false;

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

printf("Enter the previous position of R/W head: ");

scanf("%d", &previous\_head);

printf("Enter the cylinder size: ");

scanf("%d", &limit);

if (previous\_head - initial\_head > 0)

{

choice = 2;

}

else

choice = 1;

// scanf("%d",&choice);

int seek\_time = 0;

printf("%d -> ", initial\_head);

int cp\_initial\_head = initial\_head;

if (choice == 1)

{

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("%d -> \n", limit - 1);

seek\_time += abs(limit - 1 - initial\_head);

initial\_head = 0;

for (i = 0; i < cp\_initial\_head; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

else if (choice == 2)

{

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("%d -> ", 0);

seek\_time += abs(initial\_head - 0);

initial\_head = limit - 1;

for (i = limit; i > cp\_initial\_head; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

printf("Seek Time: %d\n", seek\_time);

}

int fcfs()

{

int i, no\_of\_requests, initial\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

int request[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

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

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

int seek\_time = 0;

printf("%d -> ", initial\_head);

for (i = 0; i < no\_of\_requests; i++)

{

if (i == no\_of\_requests - 1)

printf("%d\n", request[i]);

else

printf("%d -> ", request[i]);

seek\_time += abs(request[i] - initial\_head);

initial\_head = request[i];

}

printf("Seek Time: %d\n", seek\_time);

}

int lookscan()

{

int i, no\_of\_requests, initial\_head, limit, j, choice, previous\_head;

printf("Enter the number of requests: ");

scanf("%d", &no\_of\_requests);

struct request req[no\_of\_requests];

printf("Enter the requests: ");

for (i = 0; i < no\_of\_requests; ++i)

{

scanf("%d", &req[i].request\_track\_number);

req[i].visited = false;

}

printf("Enter initial position of R/W head: ");

scanf("%d", &initial\_head);

printf("Enter the previous position of R/W head: ");

scanf("%d", &previous\_head);

printf("Enter the cylinder size: ");

scanf("%d", &limit);

if (previous\_head - initial\_head > 0)

{

choice = 2;

}

else

choice = 1;

// scanf("%d",&choice);

int seek\_time = 0;

printf("%d -> ", initial\_head);

if (choice == 1)

{

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

else if (choice == 2)

{

for (i = initial\_head; i >= 0; i--)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

for (i = initial\_head; i < limit; i++)

{

for (j = 0; j < no\_of\_requests; j++)

{

if (req[j].request\_track\_number == i && req[j].visited == false)

{

printf("%d -> ", req[j].request\_track\_number);

req[j].visited = true;

seek\_time += abs(req[j].request\_track\_number - initial\_head);

initial\_head = req[j].request\_track\_number;

}

}

}

printf("\n");

}

printf("Seek Time: %d\n", seek\_time);

}

int main()

{

while (true)

{

/\* code \*/

int n;

printf("1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK \n");

scanf("%d", &n);

switch (n)

{

case 1:

{

fcfs();

}

break;

case 2:

{

sstf();

}

break;

case 3:

{

scan();

}

break;

case 4:

{

cscan();

}

break;

case 5:

{

lookscan();

}

break;

case 6:

{

clook();

}

break;

default:

break;

}

}

}

Output:

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

1

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

50 -> 82 -> 170 -> 43

Seek Time: 247

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

2

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

50 -> 43 ->82 ->170 ->

Seek Time: 134

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

3

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

Enter the previous position of R/W head: 50

Enter the cylinder size: 190

50 -> 82 -> 170 -> 189 -> 43 -> 0

Seek Time: 328

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

4

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

Enter the previous position of R/W head: 10

Enter the cylinder size: 190

50 -> 82 -> 170 -> 189 ->

43 ->

Seek Time: 182

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

5

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

Enter the previous position of R/W head: 10

Enter the cylinder size: 190

50 -> 82 -> 170 -> 43 ->

Seek Time: 247

1->FCFS 2->SSTF 3->SCAN 4->C-SCAN 5->LOOK 6->C-LOOK

6

Enter the number of requests: 3

Enter the requests: 82 170 43

Enter initial position of R/W head: 50

Enter the previous position of R/W head: 10

Enter the cylinder size: 190

50 -> 82 -> 170 -> 43 ->

Seek Time: 163