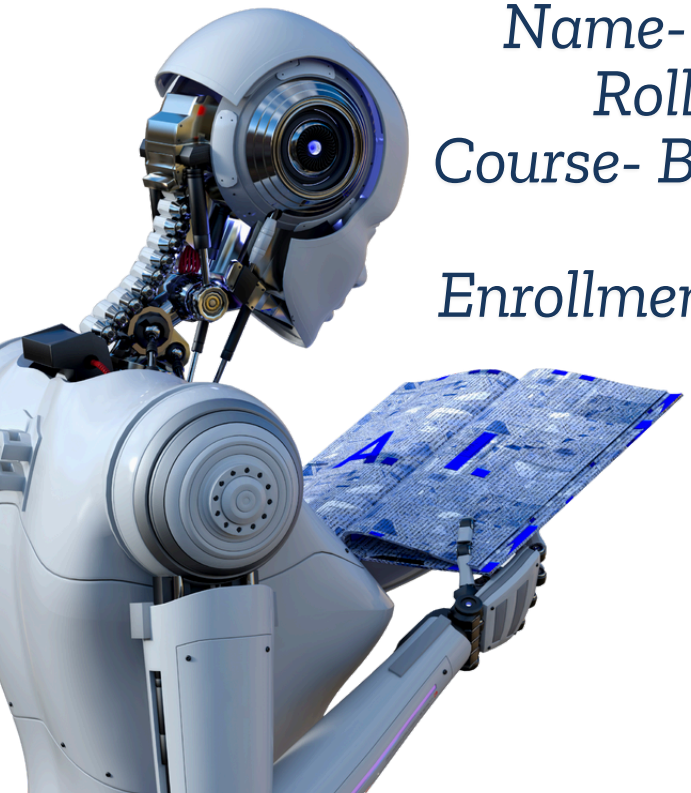
OPERATING SYSTEMS

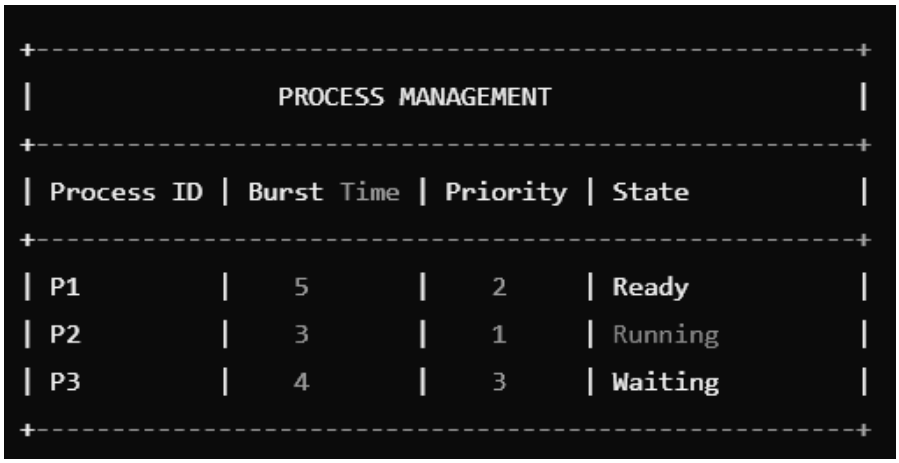
PRACTICAL FILE

Submitted to-
DR. PARUL JAIN

Name- Aaradhya Sharma
Roll no.- 24/48009
Course- Bsc. Computer science hons.
Enrollment No.- 24003570019

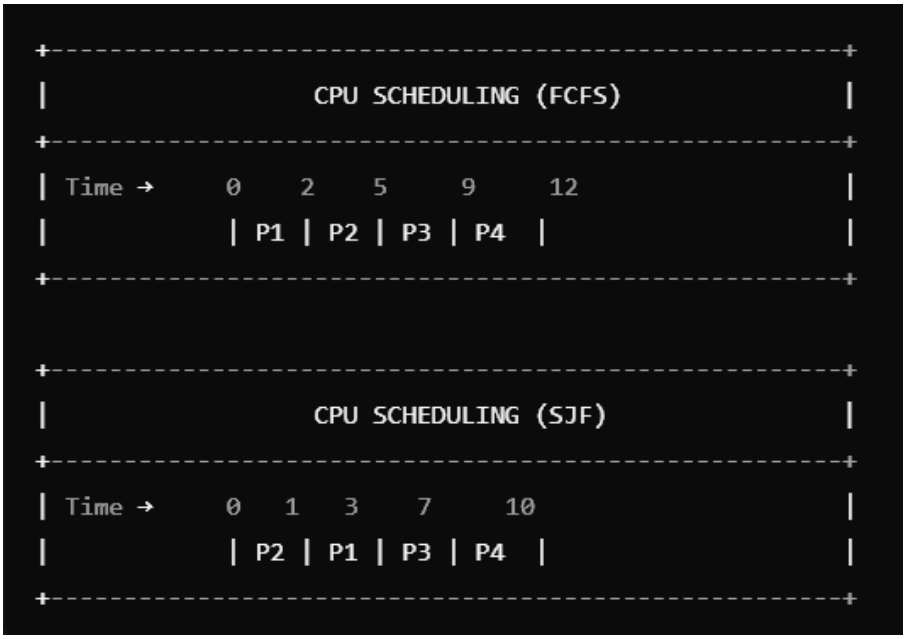


Q1. Demonstration of various Operating System functions using OS Simulator.



PROCESS MANAGEMENT

Process ID	Burst Time	Priority	State
P1	5	2	Ready
P2	3	1	Running
P3	4	3	Waiting

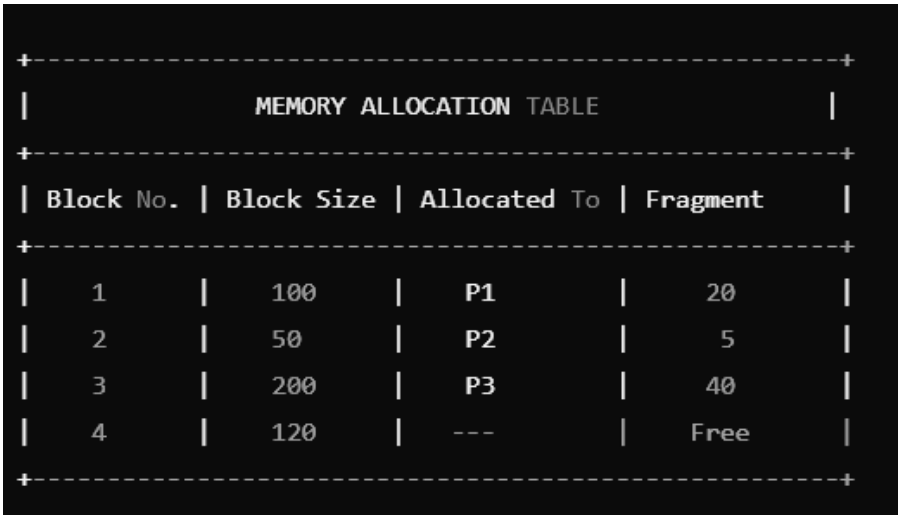


CPU SCHEDULING (FCFS)

Time →	0	2	5	9	12
	P1	P2	P3	P4	

CPU SCHEDULING (SJF)

Time →	0	1	3	7	10
	P2	P1	P3	P4	



MEMORY ALLOCATION TABLE

Block No.	Block Size	Allocated To	Fragment
1	100	P1	20
2	50	P2	5
3	200	P3	40
4	120	---	Free

```
Reference String: 7 0 1 2 0 3 0 4 2 3
Frames:

+-----------+-----------+-----------+
|     7     |     0     |     1     |
+-----------+-----------+-----------+
|     2     |     0     |     3     |
+-----------+-----------+-----------+
|     0     |     4     |     2     |
+-----------+-----------+-----------+
Page Faults: 9
```

```
+-----------+-----------+-----------+
|     7     |     0     |     1     |
+-----------+-----------+-----------+
|     0     |     2     |     3     |
+-----------+-----------+-----------+
|     4     |     2     |     3     |
+-----------+-----------+-----------+
Page Faults: 7
```

```
+-----------------------------------------------------+
|                  FILE SYSTEM VIEW                   |
+-----------------------------------------------------+
| /root                                               |
|  ├── documents                                      |
|  |     ├── os_notes.txt                             |
|  |     ├── schedule.c                               |
|  |     └── memory.md                                |
|  ├── bin                                            |
|  └── downloads                                      |
+-----------------------------------------------------+
```

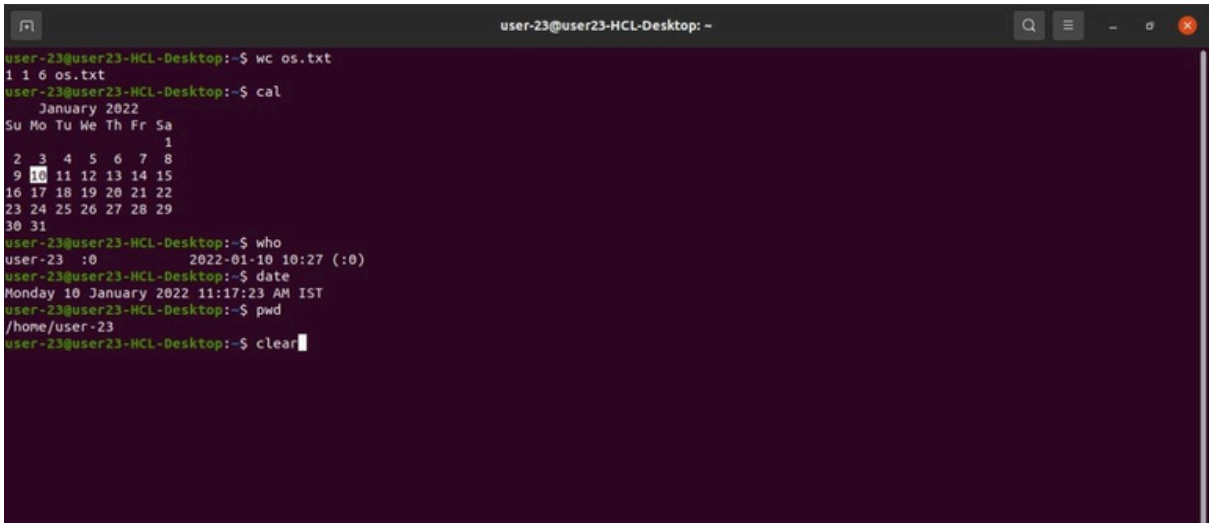
```
+---------------------------+
|        DEADLOCK RAG       |
+---------------------------+
| P1 → R1 → P2 → R2 → P1    |
|                           |
|  Deadlock Detected ✓      |
+---------------------------+
```

Q2. Execute various LINUX commands for:

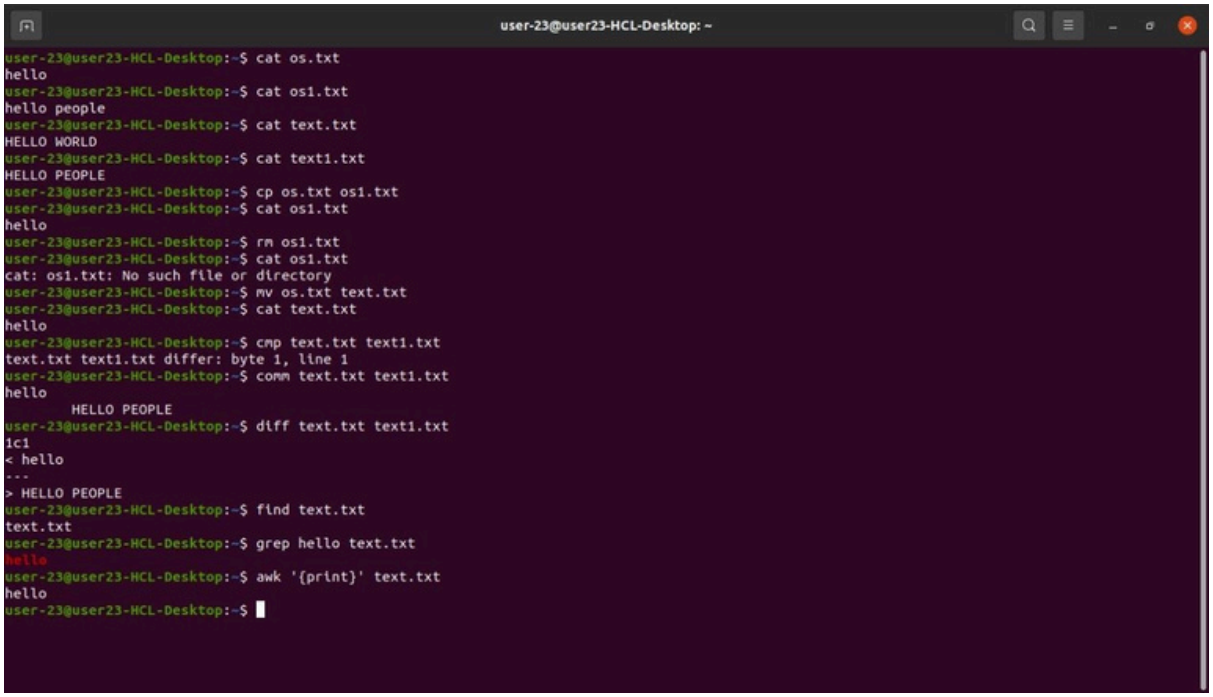
i. Information Maintenance: wc, clear, cal, who, date, pwd
ii. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk
iii. Directory Management : cd, mkdir, rmdir, ls

Output-

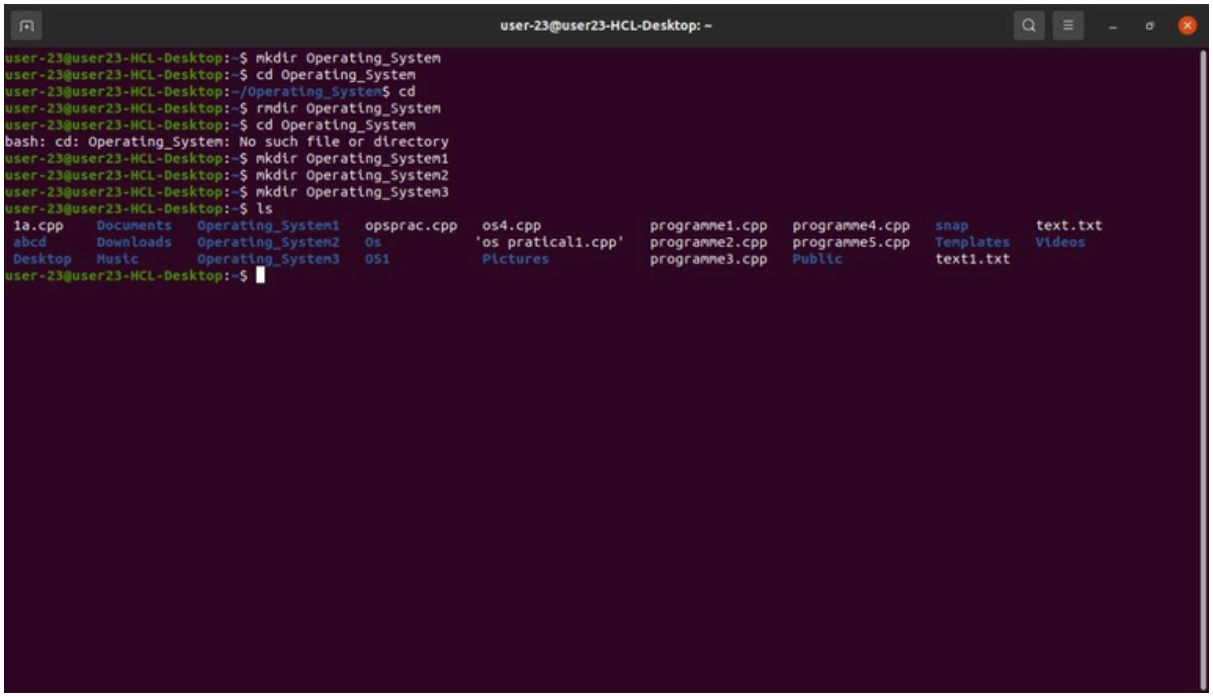
i.



ii.



iii.


user-23@user23-HCL-Desktop: ~
user-23@user23-HCL-Desktop:~$ mkdir Operating_System
user-23@user23-HCL-Desktop:~$ cd Operating_System
user-23@user23-HCL-Desktop:~/Operating_System$ cd
user-23@user23-HCL-Desktop:~$ rmdir Operating_System
user-23@user23-HCL-Desktop:~$ cd Operating_System
bash: cd: Operating_System: No such file or directory
user-23@user23-HCL-Desktop:~$ mkdir Operating_System1
user-23@user23-HCL-Desktop:~$ mkdir Operating_System2
user-23@user23-HCL-Desktop:~$ mkdir Operating_System3
user-23@user23-HCL-Desktop:~$ ls
1a.cpp Documents Operating_System1 opsprac.cpp os4.cpp programme1.cpp programme4.cpp snap text.txt
abcd Downloads Operating_System2 Os 'os pratical1.cpp' programme2.cpp programme5.cpp Templates Videos
Desktop Music Operating_System3 OS1 Pictures programme3.cpp Public text1.txt
user-23@user23-HCL-Desktop:~$

Q3. Execute various LINUX commands for:

i. Process Control: fork, getpid, ps, kill, sleep
ii. Communication: Input-output redirection, Pipe
iii. Protection Management: chmod, chown, chgrp

Output-

i.

```
export "PS1=$ "
export "PS1=$ "

$ id
export "PS1=$ "
id

uid=1000(webmaster) gid=1000(webmaster) groups=1000(webmaster)
export "PS1=$ "

$ getpid
bash: getpid: command not found
$ sleep 2
$ kill -1
kill: usage: kill [-s sigspec | -n signum | -sigspec] pid | jobspec
$ fork
bash: fork: command not found
```

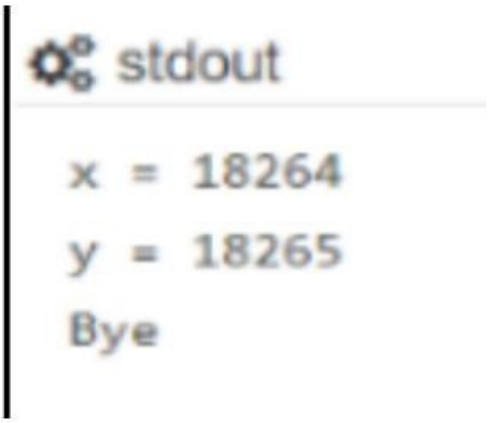
Q4. Write a program(using fork() and/or exec() commands) where parent and child execute:

i. same program, same code.
ii. same program, different code.
iii. before terminating, the parent waits for the child to finish its task.

i)

```
#include <iostream>
#include <unistd.h>
using namespace std;

int main() {
    int x, y = 0;
    x = fork();
    cout << "\n x = " << x;
    cout << "\n y = " << y;
    cout << "\n Bye" << endl;
    return 0;
}
```

Output:

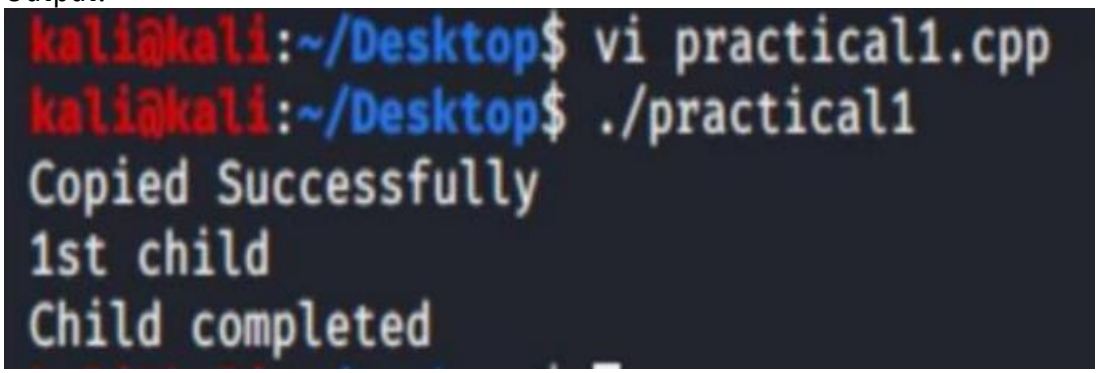


ii)

```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
using namespace std;

int main() {

    int x;
    x = fork();
```

```
    if (x == 0) {
        execlp("/home/kali/Desktop/practical5", "practical5", NULL);
    } else {
        wait(NULL);
        cout << "Child completed\n";
    }

    return 0;
}
```

Output:



iii.

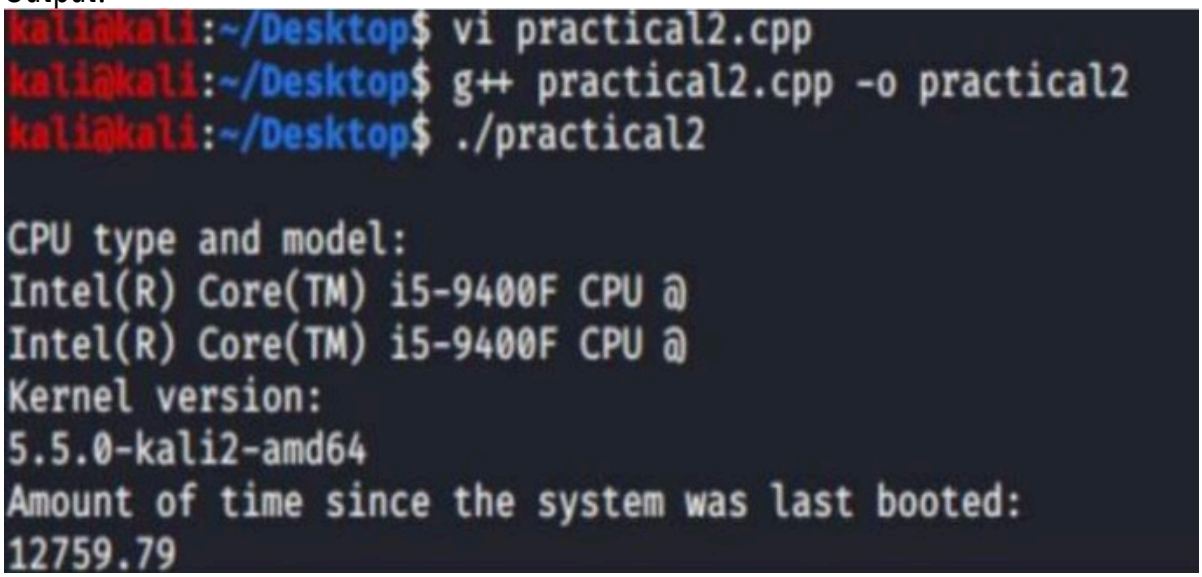
```
#include <iostream>
#include <sys/wait.h>
#include <unistd.h>
using namespace std;

int main() {
    pid_t id = fork();

    if (id == 0) {
        cout << "\n Child Terminated";
    } else {
        wait(NULL);
        cout << "\n Parent Terminated";
    }

    return 0;
}
```

Q5. Write a program to to report behavior of Linux kernel including kernel version, CPU type and model. (CPU information)

```
#include<iostream>
#include<unistd.h>
#include<string.h>
using namespace std;
void defVersion() {

cout << "\nCPU type and model: \n";
system("grep 'model name' /proc/cpuinfo | awk '{print $4,$5, $6, $7, $8}'");
cout << "Kernel version: \n";
system("cat /proc/sys/kernel/osrelease");
cout << "Amount of time since the system was last booted:\n";
system("cat /proc/uptime | awk '{print $1\n}'");}

void secVersion() {
cout << "\nPRINT IN AWK\n";
cout << "Amount of time user has spent in:\nUser mode:\n";
system("grep 'cpu' /proc/stat | awk 'NR == 1 {print $2}'");
cout << "System mode: \n";
system("grep 'cpu' /proc/stat | awk ' NR == 1 {print $4}'");
cout << "Idle: \n";
system("grep 'cpu' /proc/stat | awk 'NR == 1 {print $5}'");
cout << "\nSHELL SCRIPT\n";
cout << "Amount of time user has spent in:\n";
system("Q2SecVersion");
cout << "\nNumber of disk requests:\n";
system("grep 'intr' /proc/stat | awk '{print $17}'");
cout << "Number of context switches: \n";
system("grep 'ctxt' /proc/stat | awk '{print $2}'");
cout << "Time at which system was booted:\n";
system("grep 'btime' /proc/stat | awk '{print $2}'");
cout << "Time at which system was booted:\n";
system("Q2EpochToLocal");
cout << "Number of processes created:\n";
```

```
}


cout << "\nAmount of memory configured into system: " <<sInfo.totalram/1024 <<
"KB";
cout << "\nAmount of memory currently available: " <<sInfo.freeram/1024 << "KB";
cout << "\nLoad averages: \n";
for (int i = 1; i <= b/a; i++) {

cout << i << ". " << sInfo.loads[0] << "\n";
if (i == b/a) break;
sleep(a);
if (sysinfo(&sInfo) < 0) {

cout << "\nError in finding information about the system";
exit(1);

}
}

}
int main(int ac, char *av[]) {

switch (ac){
case 1: defVersion();

break;
case 2: if (strcmp(av[1], "-s") == 0) {

defVersion(); secVersion();
}
else cout << "\nInvalid option\n";

break;
case 4: if (strcmp(av[1], "-l") == 0) {

defVersion();
secVersion();
thirdVersion(atoi(av[2]), atoi(av[3]));
}
else cout << "\nInvalid option\n";
break;
default: cout << "\nInvalid number of arguments\n";

}
}
```

Output:


kali@kali:~/Desktop$ vi practical2.cpp
kali@kali:~/Desktop$ g++ practical2.cpp -o practical2
kali@kali:~/Desktop$ ./practical2
CPU type and model:
Intel(R) Core(TM) i5-9400F CPU @
Intel(R) Core(TM) i5-9400F CPU @
Kernel version:
5.5.0-kali2-amd64
Amount of time since the system was last booted:
12759.79

Q6. Write a program to report behaviour of Linux kernel including information on 19 configured memory, amount of free and used memory. (Memory information)

```
#include<iostream>
#include<unistd.h>
#include<string.h>
using namespace std;
void defVersion() {

cout << "\nCPU type and model: \n";
system("grep 'model name' /proc/cpuinfo | awk '{print $4,$5, $6, $7, $8}'");
cout << "Kernel version: \n";
system("cat /proc/sys/kernel/osrelease");
cout << "Amount of time since the system was last booted:\n";
system("cat /proc/uptime | awk '{print $1\n}'");
cout<<"The configured memory is; \n";
system("cat /proc/meminfo | awk 'NR==1{print $2}'\n");
cout<<"Amount of free memory is: \n";
system("cat /proc/meminfo | awk 'NR==2{print $2}'\n");
cout<<"Amount of used memory is: \n";
system("cat /proc/meminfo | awk '{if (NR==1) a=$2;
if(NR==2) b=$2} END {print ab}'\n"); }

int main(int ac, char *av[]){
defVersion();
return 0;

}
```

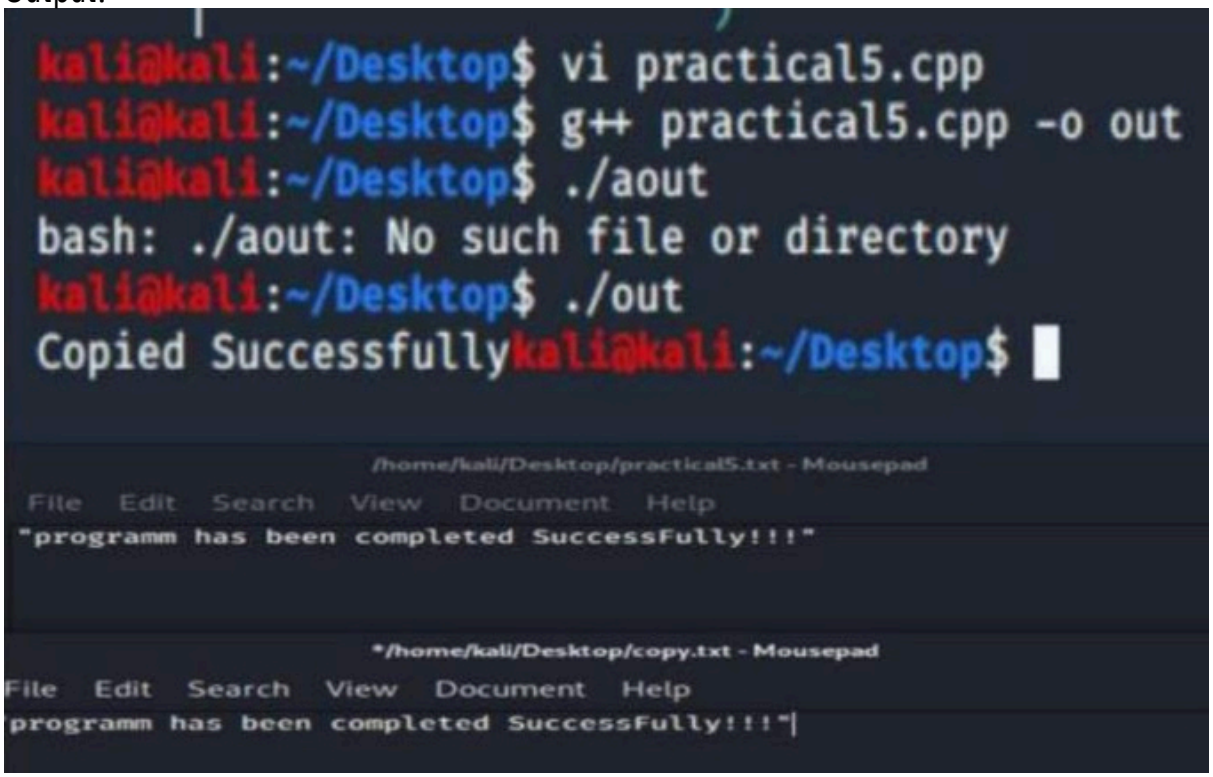
Output-

```
kali@kali:~/Desktop$ vi practical3.cpp
kali@kali:~/Desktop$ g++ practical3.cpp -o practical3
kali@kali:~/Desktop$ ./practical3

CPU type and model:
Intel(R) Core(TM) i5-9400F CPU @
Intel(R) Core(TM) i5-9400F CPU @
Kernel version:
5.5.0-kali2-amd64
Amount of time since the system was last booted:
15065.01
The configured memory is;
2039488
Amount of free memory is:
197396
Amount of used memory is:
1842092
```

Q7.Write a program to copy files using system calls.

```
#include<iostream>
#include<fstream>
using namespace std;
int main() {

ifstream fs;
ofstream ft;
char ch;
fs.open("practical5.txt");
if(!fs){

cout<<"ERROR in opening";
exit(1);

}
ft.open("copy.txt");
if(!ft){

cout<<"ERROR in opening";
fs.close();
exit(2);

}
while(fs.eof()==0){

fs>>ch;
ft<<<"Copied Successfully";
fs.close();
ft.close();

return 0;
}
```

Output:


kali@kali:~/Desktop$ vi practical5.cpp
kali@kali:~/Desktop$ g++ practical5.cpp -o out
kali@kali:~/Desktop$ ./aout
bash: ./aout: No such file or directory
kali@kali:~/Desktop$ ./out
Copied Successfullykali@kali:~/Desktop$
/home/kali/Desktop/practical5.txt - Mousepad
File Edit Search View Document Help
"programm has been completed SuccessFully!!!"
*/home/kali/Desktop/copy.txt - Mousepad
File Edit Search View Document Help
programm has been completed SuccessFully!!!"

Q8. Write a program to implement FCFS scheduling algorithm.

```
#include<iostream>
using namespace std;
int main()
{

int n;
cout<<"Enter no of processes = ";
cin>>n;
int arrival[n],p[n],burst[n];
cout<<"Enter processes and burst time "<<endl;
for(int i=0;i<n;i++)
{

cin>>p[i] >>burst[i];
}
cout<<"Enter Arrival time of process"<<endl;
for(int i=0;i<n;i++)
{

cin>>arrival[i];
}
cout<<"\nProcesses\t\t \tBurst Time\t\t\tArrival
Time"<<endl;
for(int i=0;i<n; i++)
{

cout<<p[i]<<" \t\t\t"<<burst[i]<<"\t\t\t"<<arrival[i]<<endl;
}
int sum1=0,tat[n];
cout<<"Turn around time"<<endl;
for(int i=0;i<n ;i++)
{

sum1=sum1+burst[i];
tat[i]=sum1-arrival[i];
cout<<"Turn around time "<<tat[i];

}
int wait[n],sum=0;
float average =0.0;
cout<<"Waiting time ->"<<endl;
for(int i=0;i<n ;i++)
{

wait[i]=tat[i]-burst[i];
sum=sum+wait[i];
cout<<wait[i]<<endl;

}
average=sum/n;
cout<<"Average time:"<<average<<endl;}
```

Output:

```
Enter no of processes = 4
Enter processes and burst time
1 17
2 7
3 8
4 18
Enter Arrival time of process
1
4
2
3

Processes                       Burst Time                      Arrival Time
1                       17                      1
2                       7                       4
3                       8                       2
4                       18                      3
Turn around time
Turn around time 16Turn around time 20Turn around time 30Turn around time 47Waiting time ->
-1
13
22
29
Average time:15

--------------------------------
Process exited after 35.23 seconds with return value 0
Press any key to continue . . .
```

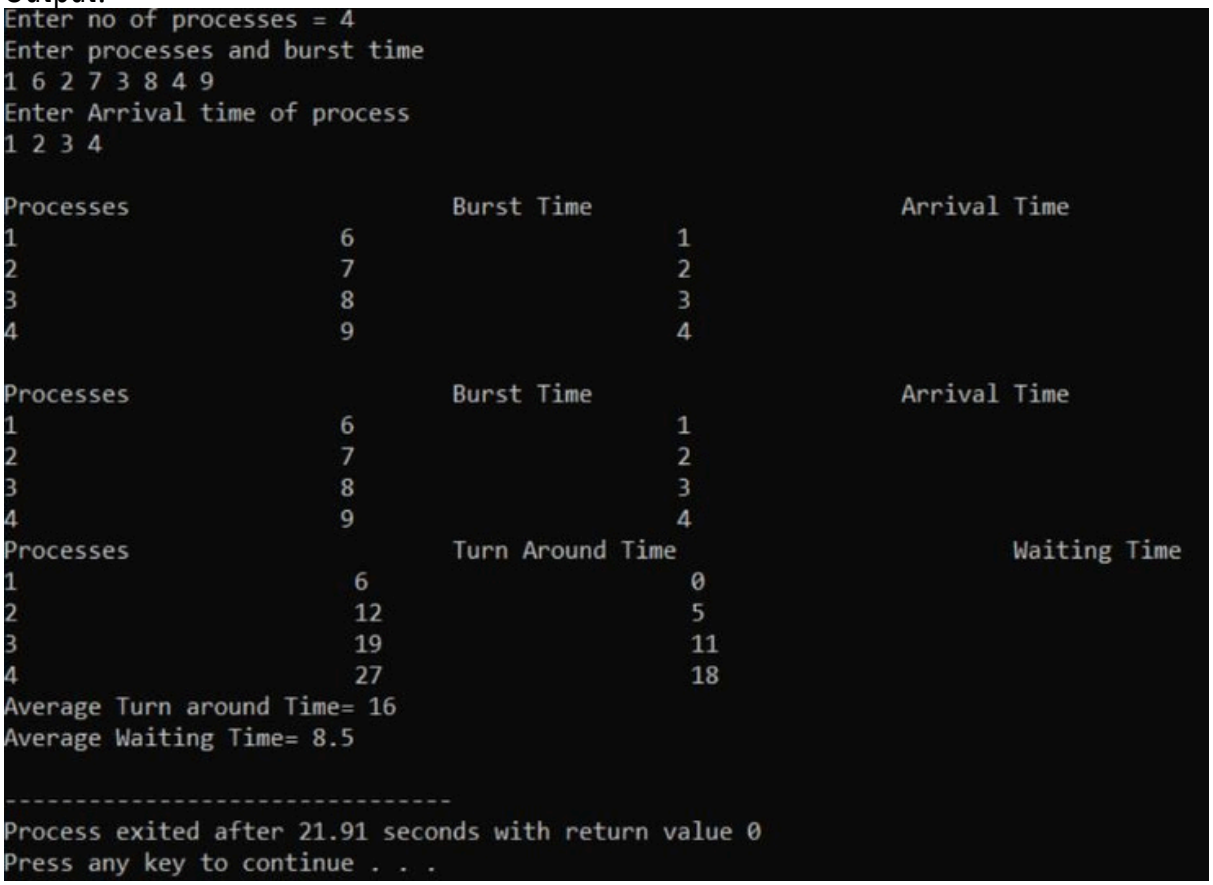
Q9.Write a program to implement SJF scheduling algorithm.

```
#include<iostream>
using namespace std;
int main()
{

int n;
cout<<"Enter no of processes = ";
cin>>n;
int arrival[n],p[n],burst[n],wait[n],tat[n] ,temp;
float total1 =0.0,total=0.0,average=0.0,avg=0.0;
float min, exit[n],var;
int a;
cout<<"Enter processes and burst time "<<endl;
for(int i=0;i<n;i++)
{

>>p[i] >>burst[i];
}
cout<<"Enter Arrival time of process"<<endl;
for(int i=0;i<n;i++)
{

cin>>arrival[i];
}
cout<<"\nProcesses\t\t \tBurst Time\t\t\tArrival
Time"<<endl;
for(int i=0;i<n;i++)
{

cout<<p[i]<<" \t\t\t"<<burst[i]<<"\t\t\t"<<arrival[i]<<endl;
}
for(int i=0;i<n;i++)
{

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

if(burst[i]>burst[j])
{

temp=arrival[i];
arrival[i]=arrival[j];
arrival[j]=temp;
temp=burst[i];
burst[i]=burst[j];
burst[j]=temp;
temp=p[i];
p[i]=p[j];
p[j]=temp;

}
```

```
}
}
cout<<"\nProcesses\t\t \tBurst Time\t\t\tArrival
Time"<<endl;
for(int i=0;i<n ;i++)
{

cout<<p[i]<<" \t\t\t"<<burst[i]<<"\t\t\t"<<arrival[i]<<endl;
}
min=arrival[0];
for(int i=0;i<n ;i++){

if(min>arrival[i]){
min=arrival[i];
a=i;

}
}
var=min;
exit[a]=var+burst[a];
var=exit[a];
for(int i=0;i<n ;i++){

if(arrival[i]!=min){
exit[i]=burst[i]+var;
var=exit[i];

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

tat[i]=exit[i]-arrival[i];
total=total+tat[i];
wait[i]=tat[i]-burst[i];
total1=total1+wait[i];

}
average=total/n;
cout<<"Processes\t\t\tTurn Around Time\t\t\tWaiting Time"<<endl;
for(int i=0;i<n ;i++)
{

cout<<p[i]<<" \t\t\t "<<tat[i]<<"\t\t\t "<<wait[i]<<endl;
}
avg=total1/n;
cout<<"Average Turn around Time= "<<average<<endl;
cout<<"Average Waiting Time= "<<avg<<endl;

}
```

}

Output:


Enter no of processes = 4
Enter processes and burst time
1 6 2 7 3 8 4 9
Enter Arrival time of process
1 2 3 4
Processes Burst Time Arrival Time
1 6 1
2 7 2
3 8 3
4 9 4
Processes Burst Time Arrival Time
1 6 1
2 7 2
3 8 3
4 9 4
Processes Turn Around Time Waiting Time
1 6 0
2 12 5
3 19 11
4 27 18
Average Turn around Time= 16
Average Waiting Time= 8.5
Process exited after 21.91 seconds with return value 0
Press any key to continue . . .

Q10. Write a program to implement non-preemptive priority based scheduling algorithm.

```
#include<iostream>
using namespace std;
struct Process
{

int burst ,arrival ,priority;
};
Process p[10];
int main()
{

float avg=0,avg_turn_at=0,n;
int temp, min,b,k=1, temp_arr[10], priority[10], arrival[10], burst[10], wait[10],

TAT[10] ,avg_wt,avg_tat,i;
cout<<"Enter the number of process : ";
cin>>n;
cout<<"\n Enter process : time priorities \n";
for(i=0;i<n ;i++)
{

cout<<"\nBurst time of process no "<<i+1<<" ->";
cin>>burst[i];
cout<<"\nEnter priority of process no "<<i+1<<" ->";
cin>>priority[i];
cout<<"\nEnter Arrival Time of process no "<<i+1<<" ->";
cin>>arrival[i];

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

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

if(arrival[i]<arrival[j])
{

temp=arrival[j];
arrival[j]=arrival[i];
arrival[i]=temp;
temp=burst[j];
burst[j]=burst[i];
burst[i]=temp;

}
}
}

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

b=b+burst[j];
min=burst[k];
for(int i=k ;i<n; i++){
```

```
min=priority[k];
if(b>=arrival[i])
{

if(priority[i]<min){
temp=arrival[k];
arrival[k]=arrival[i];
arrival[i]=temp;
temp=burst[k];
burst[k]=burst[i];
burst[i]=temp;
temp=priority[k];
priority[k]=priority[i];
priority[i]=temp;
}

}

}
k++;
}
temp_arr[0]=0;
cout<<"PROCESS\t BURST\t
ARRIVAL\tPRIORITY\tWAIT_TIME\tTAT\n";
for(int i=0;i<n ;i++)
{

wait[i]=0;
TAT[i]=0;
temp_arr[i+1]=temp_arr[i]+burst[i];
wait[i]=temp_arr[i]-arrival[i];
TAT[i]=wait[i]+burst[i];
avg_wt=avg_wt+wait[i];
avg_tat=avg_tat+TAT[i];
cout<<i+1<<"\t "<<burst[i]<<"\t

"<<arrival[i]<<"\t\t"<<priority[i]<<"\t\t"<<wait[i]<<"\t\t"<<TAT[i]<<endl;
}
avg_wt=avg_wt/n;
avg_tat=avg_tat/n;
cout<<"\n Average Wait Time : \n"<<avg_wt;
cout<<"\n Average Turn Around Time : \n"<<avg_tat;

return 0;
}
```

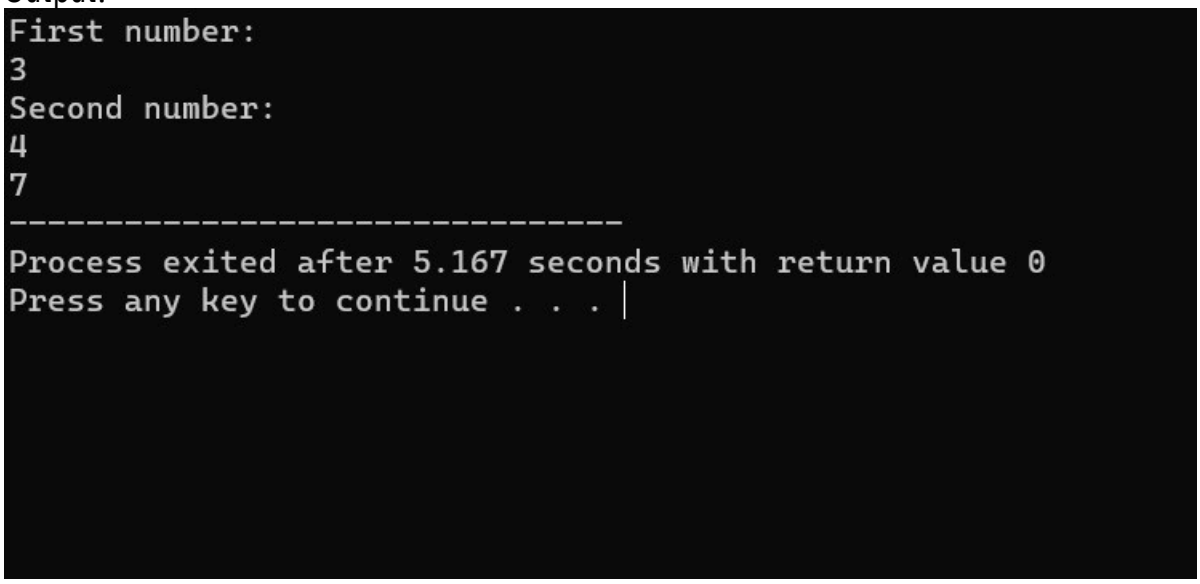
Output:

```
Enter the number of process : 3

 Enter process : time priorities

Burst time of process no 1 ->3

Enter priority of process no 1 ->2

Enter Arrival Time of process no 1 ->1

Burst time of process no 2 ->5

Enter priority of process no 2 ->1

Enter Arrival Time of process no 2 ->4

Burst time of process no 3 ->7

Enter priority of process no 3 ->3

Enter Arrival Time of process no 3 ->2
PROCESS  BURST   ARRIVAL        PRIORITY        WAIT_TIME       TAT
1        3       1               2              -1              2
2        7       2               1              1               8
3        5       4               3              6               11

 Average Wait Time :
1
 Average Turn Around Time :
6
--------------------------------
Process exited after 80.34 seconds with return value 0
Press any key to continue . . .
```

Q11. Write a program to calculate sum of n numbers using Pthreads. A list of n numbers is divided into two smaller list of equal size, two separate threads are used to sum the sublists.

```
#include <iostream>
#include<pthread.h>
using namespace std;
int global[2];
void *sum_thread(void *arg)
{

int *args_array;
args_array = (int*)arg;
int n1,n2,sum;
n1=global[0];
n2=global[1];
sum = n1+n2;
cout<<"\n Sum = "<<sum;
return NULL;

}
int main()
{

cout<<"\n First number:\n ";
cin>>global[0];
cout<<"\n Second number: \n";
cin>>global[1];
pthread_t tid_sum;
pthread_create(&tid_sum,NULL,sum_thread,(void*)&global);
pthread_join(tid_sum,NULL);
return 0;

}
```

Output:


First number:
3
Second number:
4
7

Process exited after 5.167 seconds with return value 0
Press any key to continue . . .

Q12. Write a program to implement first-fit, best-fit and worst-fit allocation strategies

i.first fit

```
#include<iostream>
using namespace std;
void show(int block[],int burst[],int n,int c ,int allocation[]){

cout<<"processes No "<<"size "<<"Allocated at "<<"Block size\n";
for(int i=0;i<c;i++){

if(allocation[i]==-1){
cout<<i+1<<"\t"<<burst[i]<<"\t\t"<<"Not Allocated\n";

}
else{

cout<<i+1<<"\t"<<burst[i]<<"\t\t"<<allocation[i]+1<<"\t"<<block[allocation[i]]
<<"\n";} }
}
void first_fit(int block[],int burst[],int n,int c){

int allocation[c];
for(int i=0;i<c;i++)

allocation[i] = -1;
int memoryused[n];

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

memoryused[i]= 0;
for(int i=0;i<c;i++){//c=No of Processes

for(int j=0;j<n;j++){//j=No of blocks
if((block[j]-memoryused[j])>=burst[i]){

memoryused[j]+=burst[i];
allocation[i]=j;
j=n; }

}
show(block,burst,n,c,allocation);

}
int main(){
```

```
int n,c;
int allocation[];
cout<<"No of Memory blocks you want to enter:-";
cin>>n;
int Block[n];
cout<<"Enter no of process u want";
cin>>c;
int Burst[c];
cout<<"Enter Size of memory blocks respectively:-";
for(int i=0;i<n;i++)



cin>>Block[i];

cout<<"Enter Size of Processess respectively:-";
for(int i=0;i<n;i++)

cin>>Burst[i];
first_fit(Block,Burst,n,c);
return 0;

}
```

Output:

```
No of Memory blocks you want to enter:-4
Enter no of process u want4
Enter Size of memory blocks resp:-50
20
80
30
Enter Size of Processess resp:-25
40
70
30
processes No size Allocated at Block size
1       25              1       50
2       40              3       80
3       70              Not Allocated
4       30              3       80

--------------------------------
Process exited after 35.51 seconds with return value 0
Press any key to continue . . .
```

ii. best fit

```
#include<iostream>
using namespace std;
void show(int block[],int burst[],int n,int c ,int allocation[]){

cout<<"processes size "<<"Allocated at "<<"Block size\n";
for(int i=0;i<c;i++){

if(allocation[i]==-1){
cout<<burst[i]<<"\t\t"<<"Not Allocated\n";

}
else{

cout<<burst[i]<<"\t\t"<<allocation[i]+1<<"\t"<<block[allocation[i]]<<"
\n";
}

}

}
```

```
void best_fit(int block[],int burst[],int n,int c){

int allocation[c];
for(int i=0;i<c;i++)

allocation[i] = -1;
int memoryused[n];
for(int i=0;i<n;i++)

memoryused[i]= 0;

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

if(block[j]<block[j+1]){
int t=0;
t=block[j];
block[j]=block[j+1];
block[j+1]=t;

}
}

}
for(int i=0;i<c-1;i++){

for(int j=0;j<c-1;j++)
{

if(burst[j]<burst[j+1]){
int t=0;
t=burst[j];
burst[j]=burst[j+1];
burst[j+1]=t;

}
}

}
for(int i=0;i<c;i++){//c=No of Processes

for(int j=0;j<n;j++){//j=No of blocks
if((block[j]-memoryused[j])>=burst[i]){

memoryused[j]+=burst[i];
allocation[i]=j;
j=n;
```

```
}
}

}
show(block,burst,n,c,allocation);
}
int main(){

int n, c;
cout<<"No of Memory blocks you want to enter:-";

cin>>n;
int Block[n];
cout<<"Enter no of process u want";
cin>>c;
int Burst[c];
cout<<"Enter Size of memory blocks respectively:-";
for(int i=0;i<n;i++)

cin>>Block[i];
cout<<"Enter Size of Processess respectively:-";

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

cin>>Burst[i];
best_fit(Block,Burst,n,c);

}
```

Output:

```
No of Memory blocks you want to enter:-4
Enter no of process u want4
Enter Size of memory blocks resp:-50
20
80
30
Enter Size of Processess resp:-25
40
70
30
processes size Allocated at Block size
70              1       80
40              2       50
30              3       30
25              Not Allocated

--------------------------------
Process exited after 51.37 seconds with return value 0
Press any key to continue . . .
```

iii. worst fit

```
#include<iostream>
using namespace std;
void show(int block[],int burst[],int n,int c,int allocation[]){

cout<<"processes size "<<"Allocated at "<<"Block size\n";
for(int i=0;i<c;i++){

if(allocation[i]==-1){
cout<<burst[i]<<"\t\t"<<"Not Allocated\n";

}
else{

cout<<burst[i]<<"\t\t"<<allocation[i]+1<<"\t"<<block[allocation[i]]<<"
\n";
}

}
```

```
}
void worst_fit(int block[],int burst[],int n,int c){

int allocation[c];
for(int i=0;i<c;i++)

allocation[i] = -1;
int memoryused[n];

for(int i=0;i<n;i++)
memoryused[i]= 0;

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

for(int j=0;j<n-1;j++){
if(block[j]<block[j+1]){

int t=0;
t=block[j];
block[j]=block[j+1];
block[j+1]=t;

}
}

}
for(int i=0;i<c-1;i++){

for(int j=0;j<c-1;j++){
if(burst[j]>burst[j+1]){

int t=0;
t=burst[j];
burst[j]=burst[j+1];
burst[j+1]=t;

}
}

}
for(int i=0;i<c;i++){//c=No of Processes

for(int j=0;j<n;j++){//j=No of blocks

if((block[j]-memoryused[j])>=burst[i]){
memoryused[j]+=burst[i];
allocation[i]=j;
j=n;

}
}
```

```
show(block,burst,n,c,allocation);

}
int main(){

int n, c;
cout<<"No of Memory blocks you want to enter:-";

cin>>n;
int Block[n];
cout<<"Enter no of process u want";
cin>>c;
int Burst[c];
cout<<"Enter Size of memory blocks respectively:-";
for(int i=0;i<n;i++)

cin>>Block[i];
cout<<"Enter Size of Processess respectively:-";
for(int i=0;i<n;i++)

cin>>Burst[i];

worst_fit(Block,Burst,n,c);
}
```

Output:

```
No of Memory blocks you want to enter:-4
Enter no of process u want4
Enter Size of memory blocks resp:-50
20
80
30
Enter Size of Processess resp:-25
40
70
30
processes size Allocated at Block size
25                1       80
30                1       80
40                2       50
70                Not Allocated

--------------------------------
Process exited after 277 seconds with return value 0
Press any key to continue . . .
```