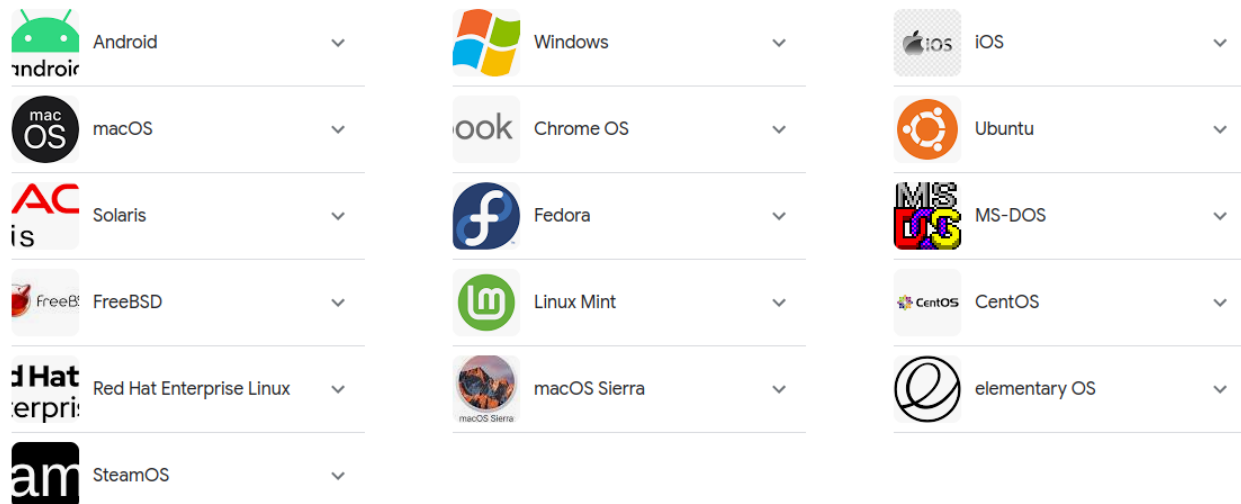


## EXPERIMENT 1

### Q1.Enlist Different Operating System that are available.

Ans:

From sources across the web



### Q2. Open Source vs License Version which one is better and why?

Ans.

#### What is Open Source?

First things first, you might say, what on earth is open source?

Essentially, an open-source solution is a software distributed under a licensing agreement which permits the code to be shared and edited by other parties. This means that anyone who knows how can use and modify open-source software completely free of charge.

Wait, what?

Yes, you read that correctly: **open-source software is available for anyone to use and modify from its original design – for free.** As a result, the software can evolve due to the many alterations of developers from around the world. And, often, it gets to the point where the final form is virtually unrecognisable from the original software. The benefit of open source is that this model produces an increasingly more diverse scope of design perspectives than a single company would ever be able to foster and sustain long-term. In other words, this is the perfect choice for anyone valuing innovation.

However, it can also leave users vulnerable to hackers.

Today, we have an organisation called the Open Source Initiative, which works to promote open-source software. Since its start in 1998, they have approved more than 80 open source licenses worldwide. Interestingly, these licenses tend to fall into one of these categories: permissive licenses and copyleft licenses.

The most basic type of open source license is the permissive license. With this one, you can do whatever you want with the software as long as you follow the general requirements, which are often phrased like this:

- You can do whatever you want with the code
- You must acknowledge the author
- However, you use it at your own risk

On the other hand, you have copyleft licenses. In addition to the requirements of a permissive license, they also require that you:

- Don't place any additional restrictions on the licensee's exercise of the license
- Make the source code for any binaries available
- The source code must be open and available under the same terms as which you got the code

### **What is Licensed Software?**

Licensed software, on the other hand, is proprietary software distributed under a licensing agreement to authorised users only.

**In other words, it's the complete opposite of open source, as the source code is not to be shared with the public for anyone to look at or modify.**

Businesses are often defensive of their product and eager to preserve control of their brand, and the licensing agreements allow them to do this. Naturally, licensed software is the perfect opportunity for those looking for low-security risks, as dedicated developers are the only ones allowed to contribute to the software's code.

### **Main differences between Open Source and Licensed Software**

#### **Cost**

Even though open-source software is technically free, there are long term costs associated with it such as implementation, innovation, support, and investing in the appropriate infrastructure as your organisation progresses, technology evolves, and your requirements grow.

Additionally, it's becoming more common for open software providers to charge extra for add-ons, integration, and additional services. This can, in some instances, undo any cost-saving advantages you might have enjoyed.

The cost of licensed software, on the other hand, can vary considerably depending on the complexity of the solution you want. This can include a base fee for the software, integration, services and annual licensing fees. Though the hard cost can be higher, it's important to remember that you pay for a more customised product from a reliable name. Additionally, you'll also benefit from:

- Improved security

- Improved functionality
- Continuous innovation
- Greater scalability
- Ongoing training and support
- Lower requirement for technical skills

### **Support**

In order to evolve, open-source software depends on a loyal and engaged online community providing support through forums and blogs.

Naturally, the response time of these communities is slower than dedicated support teams from well-known brands. This means that questions may go unanswered for some time, as there may not necessarily be any experts on hand. Additionally, there's no incentive for these communities to help – except for wanting to be cooperative.

The biggest advantage of licensed software is ongoing support, which can be imperative if you're a user without much technical skill. This support can include user manuals and points of contact for immediate assistance from experts who are closely acquainted with the product or service.

### **Security**

As open-source software isn't developed in a controlled environment, we find that security is often a concern for many.

As the developers are situated all around the world, there is often a lack of continuity and shared direction that can counteract effective communication and collaboration. Additionally, as the software isn't always peer-reviewed or validated, a developer could potentially implant a backdoor Trojan into the software without the user being aware of it.

Naturally, this scares many off.

In comparison, licensed software tends to be perceived as the more secure option.

Unlike open-source software, a licensed solution is developed in a controlled environment by a focused team. This team of dedicated developers are the only people who can view or edit the source code, meaning that the product is heavily audited and the risk of backdoor Trojans is considerably diminished.

### **Practicality**

As open-source software tends to accommodate the needs of developers rather than the majority of layperson users, the convenience and practicality of open source are frequently criticised.

Often, there are no user guides or manuals – as they are not a legal requirement – and when they are written, they tend to be written strictly for other developers. In other words, they're not written with the less technically experienced users in mind.

Expert usability testing has enabled licensed software to be more practical for a wider audience. User manuals are usually on hand for instant reference and swift training, and support services ensure that issues are solved quickly. Do you want to learn more about open-source software? Find our article on Open Source Advantages right here.

### How to choose between open source and licensed software?

The pros and cons of open source and licensed software largely depend on your team's technical expertise – and the IT resources you have available.

Additionally, your choice will also depend on the needs and requirements of your business. Does the usefulness of a system which is completely free of cost, outweigh the running costs, security risks and lack of support that comes with it? If so, then it would be in your interest to join the growing trend of open-source software.

If you, on the other hand, are part of a large business with security concerns and a need for quick support, then you may be better suited to licensed software.

### Q3. Use the following linux commands.

#### 1. Pwd

Writes to standard output the full path name of your current directory.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ pwd
/home/ailab
```

#### 2. mkdir

The mkdir command in Linux/Unix allows users to **create or make new directories**.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ mkdir test_1
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

#### 3.rmdir

The rmdir command **removes the directory, specified by the Directory parameter, from the system.**

#### 4.ls

The ls command is used to **list files**. "ls" on its own lists all files in the current directory except for hidden files.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ rmdir test_1
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ ls test_1
ls: cannot access 'test_1': No such file or directory
```

#### 5.cd

The cd command can be used to **change the working directory of the working drive or another lettered drive.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~/test_1$ cd
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

#### 6.touch

The touch command is used to generating an empty file or multiple files all at once.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ touch d.txt
```

#### 7.cat

print the content of a file onto the standard output stream.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ cat d.txt
HELLO ALL
```

#### 8.rm

Use the rm command to **remove files you no longer need**.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ rm d.txt
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ cat d.txt
cat: d.txt: No such file or directory
```

#### 9.cp

You use the cp command for **copying files from one location to another**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ cat d.txt
HELLO ALL
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ cp d.txt d1.txt
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ cat d1.txt
HELLO ALL
```

#### 10.mv

Use the mv command to **move files and directories from one directory to another or to rename a file or directory**.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ mv d1.txt Desktop
```

#### 11.head

The head command **allows you to see the initial lines of a file in standard output without opening a file**.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ head d.txt
HELLO ALL
1
2
3
4
5
6
7
8
9
```

#### 12.tail

Linux tail command is used to **display the last ten lines of one or more files**. Its main purpose is to read the error message.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ tail d.txt
3
4
5
6
7
8
9
10
11
12
```

### 13. tac

tac command in Linux is used to **concatenate and print files in reverse**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ tac d.txt
12
11
10
9
8
7
6
5
4
3
2
1
HELLO ALL
```

### 14.ifconfig

You can use the ifconfig command to **assign an address to a network interface and to configure or display the current network interface configuration information.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ ifconfig
Command 'ifconfig' not found, but can be installed with:
sudo apt install net-tools
```

### 15.wc

wc stands for word count. As the name implies, it is mainly used for **counting purpose.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ wc d.txt
13 14 37 d.txt
```

### 16.useradd

The useradd command **creates a new user account.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ useradd
Usage: useradd [options] LOGIN
       useradd -D
       useradd -D [options]

Options:
  -b, --badnames                do not check for bad names
  -b, --base-dir BASE_DIR       base directory for the home directory of the
                                new account
  -b, --btrfs-subvolume-home     use BTRFS subvolume for home directory
  -c, --comment COMMENT         GECOS field of the new account
  -d, --home-dir HOME_DIR       home directory of the new account
  -D, --defaults                 print or change default useradd configuration
  -e, --expiredate EXPIRE_DATE  expiration date of the new account
  -f, --inactive INACTIVE       password inactivity period of the new account
  -g, --gid GROUP               name or ID of the primary group of the new
                                account
  -G, --groups GROUPS           list of supplementary groups of the new
                                account
  -h, --help                    display this help message and exit
  -k, --skel SKEL_DIR           use this alternative skeleton directory
  -K, --key KEY=VALUE           override /etc/login.defs defaults
  -l, --no-log-init             do not add the user to the lastlog and
                                faillog databases
  -m, --create-home             create the user's home directory
  -M, --no-create-home         do not create the user's home directory
  -N, --no-user-group           do not create a group with the same name as
                                the user
  -o, --non-unique              allow to create users with duplicate
                                (non-unique) UID
  -p, --password PASSWORD       encrypted password of the new account
  -r, --system                 create a system account
  -R, --root CHROOT_DIR         directory to chroot into
  -P, --prefix PREFIX_DIR       prefix directory where are located the /etc/* files
  -s, --shell SHELL             login shell of the new account
  -u, --uid UID                 user ID of the new account
  -U, --user-group             create a group with the same name as the user
  -Z, --selinux-user SEUSER     use a specific SEUSER for the SELinux user mapping
  --extrausers                 Use the extra users database
```

## 17.groupadd

The groupadd command creates a new group account using the values specified on the command line plus the default values from the system.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ groupadd
Usage: groupadd [options] GROUP

Options:
  -f, --force                exit successfully if the group already exists,
                              and cancel -g if the GID is already used
  -g, --gid GID              use GID for the new group
  -h, --help                 display this help message and exit
  -K, --key KEY=VALUE        override /etc/login.defs defaults
  -o, --non-unique           allow to create groups with duplicate
                              (non-unique) GID
  -p, --password PASSWORD    use this encrypted password for the new group
  -r, --system               create a system account
  -R, --root CHROOT_DIR      directory to chroot into
  -P, --prefix PREFIX_DIR    directory prefix
  --extrausers               Use the extra users database
```

## 18.grep

Grep is a useful command to search for matching patterns in a file.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ grep -c "2" d.txt
2
```

## 19.comm

The comm command is a simple Linux utility for comparing files with focus on the common content.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ comm d.txt demo
HELLO ALL
comm: file 1 is not in sorted order
1
2
3
4
5
6
7
8
9
10
11
12
      Hello PDPU Students
comm: file 2 is not in sorted order
      1
      2
      3
      4
      5
      6
      7
      8
      9
      10
      11
      12
      13
comm: input is not in sorted order
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

20.see

It can be used to find path.



```
HELLO ALL
1
2
3
4
5
6
7
8
9
10
11
12
/home/ailab/d.txt (END)
```

### 21.tee

The tee command, used with a pipe, **reads standard input, then writes the output of a program to standard output and simultaneously copies it into the specified file or files.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ wc -l demo|tee -a d.txt
14 demo
```

### 22.uniq

The uniq command **deletes repeated lines in a file.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ uniq d.txt
HELLO ALL
1
2
3
4
5
6
7
8
9
10
11
12
14 demo
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

### 23.od

od command in Linux is used to **convert the content of input in different formats with octal format as the default format.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ od d.txt
0000000 042510 046114 020117 046101 005114 005061 005062 005063
0000020 005064 005065 005066 005067 005070 005071 030061 030412
0000040 005061 031061 030412 020064 062544 067555 000012
0000055
```

#### 24.sort

The sort command is used in Linux **to print the output of a file in given order.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ sort d.txt
1
10
11
12
14 demo
2
3
4
5
6
7
8
9
HELLO ALL
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

#### 25.gzip

Gzip (GNU zip) is a compressing tool, which is used **to truncate the file size.**



```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ gzip d.txt
```

#### 26.date

The "date" command in Linux is a simple but powerful tool used to **display the current date and time, as well as set the system date and time.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ date
Monday 14 August 2023 01:42:56 PM IST
```

#### 27.sleep

Linux sleep command **lets the terminal wait by the specified amount of time.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ sleep 100
```

#### 28.find

It can be used to **find files and directories and perform subsequent operations on them.**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ find d.txt
d.txt
```

#### 29.locate

The locate command is a Unix utility used for **quickly finding files and directories**

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ locate d.txt
/home/ailab/d.txt
/home/ailab/d.txt.gz
/snap/snap-store/582/usr/share/gnupg/help.id.txt
/snap/snap-store/959/usr/share/gnupg/help.id.txt
/usr/share/doc/alsa-base/driver/compress_offload.txt.gz
/usr/share/doc/util-linux/blkid.txt
/usr/share/doc/util-linux/deprecated.txt
/usr/share/gnupg/help.id.txt
```

### 30.rename

The rename command is used to **rename multiple files or directories** in Linux.

### 31.time

Linux time command **displays how long it takes to execute a command**. It helps in checking the performance of the scripts and commands.

```
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$ time

real    0m0.000s
user    0m0.000s
sys     0m0.000s
ailab@ailab-HP-Pro-SFF-400-G9-Desktop-PC:~$
```

### 32.mount

The mount command **attaches the filesystem of an external device to the filesystem of a system**.

**Experiment 2**

Write the program in Linux and run it in shell script

**Q1. Write a script to print user name, date and who logged in.**

```
1 echo "Enter Username"
2 read a
3 echo "Welcome ${a}!"
4 echo "*****"
5 date
6 echo
7 echo "Who is logged in?"
8 echo "${a}"
9 echo "*****"
```

```
ohm@ohm-VirtualBox:~$ ./pr1.sh
Enter Username
Ohm
Welcome Ohm
*****
Sunday 20 August 2023 02:51:26 PM IST

Who is logged in?
Ohm
*****
```

**Q2. Write a script to compare two numbers and find which one is greater.**

```
1 echo "Enter the No A"
2 read d
3 echo "Enter the No B"
4 read e
5 if [ $d -gt $e ]
6 then
7 echo "$d is greater than $e"
8 else
9 echo "$e is greater than $d"
10 fi
```

```

ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter the No A
6
Enter the No B
3
6 is greater than 3
ohm@ohm-VirtualBox:~$ 

```

**Q3. Write a script to find the type of triangle by reading lengths of the side.**

```

14
15 echo "Enter the length of triangle side 1"
16 read a
17 echo "Enter the length of triangle side 2"
18 read b
19 echo "Enter the length of triangle side 3"
20 read c
21 if [ $a -eq $b -a $b -eq $c ]
22 then
23 echo "Triangle is equilateral Triangle"
24 elif [ $a -eq $b -o $b -eq $c -o $c -eq $a ]
25 then
26 echo "Triangle is Isosceles Triangle"
27 else
28 echo "Triangle is scalene Triangle"
29 fi

```

```

ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter the length of triangle side 1
4
Enter the length of triangle side 2
3
Enter the length of triangle side 3
4
Triangle is Isosceles Triangle
ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter the length of triangle side 1
5
Enter the length of triangle side 2
5
Enter the length of triangle side 3
5
Triangle is equilateral Triangle

```

**Q4. Write a script to generate factorial of number.**

```
32
33
34 echo "Enter a number"
35 read num
36
37 fact=1
38 while [ $num -gt 1 ]
39 do
40     fact=$((fact * num))
41     num=$((num - 1))
42 done
43 echo $fact
44
45
46
47
```

```
ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter a number
5
120
```

**Q5. Write a script to generate Fibonacci series.**

```
47
48 echo "Enter the value of n"
49 read n
50 f=0
51 g=1
52 count=2
53 echo "Fibonacci series:"
54 echo $f
55 echo $g
56 while [ $count -le $n ]
57 do
58     fib=`expr $f + $g`
59     f=$g
60     g=$fib
61     echo $fib
62     count=`expr $count + 1`
63 done
64
```

```
ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter the value of n
13
Fibonacci series:
0
1
1
2
3
5
8
13
21
34
55
89
144
233
ohm@ohm-VirtualBox:~$ ./practical2.sh
```

Q6. Write a script to find whether the number is palindrome or not.

```
68 echo "Enter m"
69 read m
70 num=0
71 on=$m
72 while [ $m -gt 0 ]
73 do
74 num=$((expr $num \* 10))
75 k=$((expr $m % 10))
76 num=$((expr $num + $k))
77 m=$((expr $m / 10))
78 done
79 if [ $num -eq $on ]
80 then
81 echo palindrome
82 else
83 echo not palindrome
84 fi
```

```
ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter m
456654
palindrome
ohm@ohm-VirtualBox:~$ ./practical2.sh
Enter m
4567655
not palindrome
ohm@ohm-VirtualBox:~$
```





**Experiment 3**

**Input the process along with the burst time. Find out the average waiting time and average turnaround time using FCFS and SJF algorithms. Compare and conclude the results.**

**FCFS ALGORITHMS**

```
#include <iostream>
#include <algorithm>

using namespace std;

struct Process {
    int id;
    int burstTime;
    int waitingTime;
    int turnaroundTime;
};

int main() {
    int numProcesses;
    cout << "Enter no of processes: ";
    cin >> numProcesses;
    int burstTimes[numProcesses];

    // Manual input for burst times
    for (int i = 0; i < numProcesses; i++)
    {
        cout << "Enter burst time for Process " << (i + 1) << ": ";
        cin >> burstTimes[i];
    }

    // FCFS Scheduling
    int WaitingTimes[numProcesses];
    int TurnaroundTimes[numProcesses];

    WaitingTimes[0] = 0;
    TurnaroundTimes[0] = burstTimes[0];

    for (int i = 1; i <= numProcesses; i++)
    {
        WaitingTimes[i] = WaitingTimes[i-1] + burstTimes[i-1];
        TurnaroundTimes[i] = WaitingTimes[i] + burstTimes[i];
    }

    double AvgWaitingTime = 0.0;
    double AvgTurnaroundTime = 0.0;
```

```

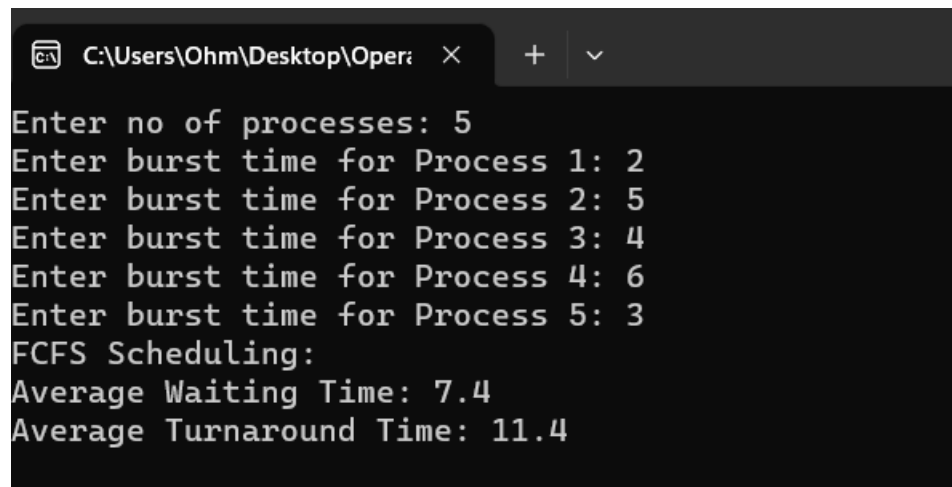
for (int i = 0; i < numProcesses; i++)
{
    AvgWaitingTime += WaitingTimes[i];
    AvgTurnaroundTime += TurnaroundTimes[i];
}

AvgWaitingTime /= numProcesses;
AvgTurnaroundTime /= numProcesses;

cout << "FCFS Scheduling:" << endl;
cout << "Average Waiting Time: " << AvgWaitingTime << endl;
cout << "Average Turnaround Time: " << AvgTurnaroundTime << endl;
}

```

## OUTPUT



```

C:\Users\Ohm\Desktop\Oper: x + v
Enter no of processes: 5
Enter burst time for Process 1: 2
Enter burst time for Process 2: 5
Enter burst time for Process 3: 4
Enter burst time for Process 4: 6
Enter burst time for Process 5: 3
FCFS Scheduling:
Average Waiting Time: 7.4
Average Turnaround Time: 11.4

```

## SJF ALGORITHMS

```

#include <iostream>
#include <algorithm>

using namespace std;

struct Process {
    int id;
    int burstTime;
    int waitingTime;
    int turnaroundTime;
};

int main() {

```

```
int numProcesses;
cout << "Enter no of processes: ";
cin >> numProcesses;
int burstTimes[numProcesses];

// Manual input for burst times
for (int i = 0; i < numProcesses; ++i) {
    cout << "Enter burst time for Process " << (i + 1) << ": ";
    cin >> burstTimes[i];
}

// SJF Scheduling
int sjfWaitingTimes[numProcesses];
int sjfTurnaroundTimes[numProcesses];

int sjfBurstTimes[numProcesses];
copy(burstTimes, burstTimes + numProcesses, sjfBurstTimes);
sort(sjfBurstTimes, sjfBurstTimes + numProcesses);

sjfWaitingTimes[0] = 0;
sjfTurnaroundTimes[0] = sjfBurstTimes[0];

for (int i = 1; i < numProcesses; ++i) {
    sjfWaitingTimes[i] = sjfWaitingTimes[i - 1] + sjfBurstTimes[i - 1];
    sjfTurnaroundTimes[i] = sjfWaitingTimes[i] + sjfBurstTimes[i];
}

double sjfAvgWaitingTime = 0.0;
double sjfAvgTurnaroundTime = 0.0;

for (int i = 0; i < numProcesses; ++i) {
    sjfAvgWaitingTime += sjfWaitingTimes[i];
    sjfAvgTurnaroundTime += sjfTurnaroundTimes[i];
}

sjfAvgWaitingTime /= numProcesses;
sjfAvgTurnaroundTime /= numProcesses;

cout << "SJF Scheduling:" << endl;
cout << "Average Waiting Time: " << sjfAvgWaitingTime << endl;
cout << "Average Turnaround Time: " << sjfAvgTurnaroundTime << endl;

return 0;
}
```

**OUTPUT**

```
Enter no of processes: 5
Enter burst time for Process 1: 2
Enter burst time for Process 2: 5
Enter burst time for Process 3: 4
Enter burst time for Process 4: 6
Enter burst time for Process 5: 3
SJF Scheduling:
Average Waiting Time: 6
Average Turnaround Time: 10
```

**Experiment 4**

**Input the process along with the burst time. Find out the average waiting time and average turnaround time using Priority and Round-robin algorithms. Compare and conclude the results.**

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

using namespace std;
void Priority();
void Round_robin();

void Round_robin()
{
    int numEntries, ts;
    cout << "Enter the TS: ";
    cin >> ts;
    cout << "Enter the number of map entries: ";
    cin >> numEntries;

    queue<int> myqueue;
    int bt[numEntries];
    for (int i = 0; i < numEntries; i++)
    {
        int key1;
        cout << "Enter the " << i + 1 << " element BurstTime : ";
        cin >> key1;
        myqueue.push(key1);
        bt[i] = key1;
    }
    float current = 0;
    vector<int> tat;
    while (!myqueue.empty())
    {
        if (myqueue.front() == 0)
        {
            tat.push_back(current);
            myqueue.pop();
        }
        else if (myqueue.front() == 1)
        {
            current += 1;
            tat.push_back(current);
            myqueue.pop();
        }
    }
```

```

    else
    {
        current += ts;
        int temp = myqueue.front() - ts;
        if (temp == 0)
        {
            tat.push_back(current);
            myqueue.pop();
        }
        else
        {
            myqueue.pop();
            myqueue.push(temp);
        }
    }
    cout << myqueue.front() << endl;
}

sort(bt, bt + numEntries);
float total_tat = accumulate(tat.begin(), tat.end(), 0);

int wt[numEntries];
float total_wt = 0;
for (int i = 0; i < tat.size(); i++)
{
    wt[i] = tat[i] - bt[i];
    total_wt += wt[i];
}

cout << "Round Robin Scheduling:" << endl;
cout << "Average Waiting Time: " << total_wt / numEntries << endl;
cout << "Average Turnaround Time: " << total_tat / numEntries << endl;
}

struct CompareMap
{
    bool operator()(const pair<int, int> &a, const pair<int, int> &b) const
    {
        if (a.second != b.second)
        {
            return a.second > b.second; // Sort by values in descending order
        }
        return a.first < b.first; // Sort by keys in ascending order when values are the same
    }
};

```

```

    }
};

void Priority()
{
    map<pair<int, int>, int, CompareMap> myMap;
    vector<map<pair<int, int>, int>::iterator> it;

    int numEntries;
    cout << "Enter the number of map entries: ";
    cin >> numEntries;
    int total = 0;
    vector<int> pre_tat;
    vector<int> post_tat;
    vector<int> wait;
    vector<int>::iterator it1;

    for (int i = 0; i < numEntries; i++)
    {
        int key1, key2;
        cout << "Enter the " << i + 1 << " element BurstTime : ";
        cin >> key1;
        cout << "Enter the " << i + 1 << " element Priority : ";
        cin >> key2;
        myMap[make_pair(key1, key2)] = i; // Value is set to the loop index for demonstration
        total = total + key1;
    }

    int sum = 0;
    cout << "\nOriginal map:" << endl;
    // for (const auto &entry : myMap)
    // {
    //     cout << entry.first.first << ", " << entry.first.second << endl;
    // }
    pre_tat.push_back(0);
    for (auto it = myMap.begin(); it != myMap.end(); it++)
    {
        sum = sum + it->first.first;
        pre_tat.push_back(sum);
    }

    sort(pre_tat.begin(), pre_tat.end(), greater<int>());

```

```

float total_tat = 0;
float total_wt = 0;
for (int i = 0; i < numEntries; i++)
{
    total_tat += pre_tat[i];
}
int sum1 = 0;
int i = 0;
int j = -1;
for (auto it = myMap.rbegin(); it != myMap.rend(); ++it)
{
    do
    {
        sum1 = pre_tat[i] - it->first.first;
        cout << sum1 << endl;
        wait.push_back(sum1);
        i = i + 1;

    } while (i < j);
}
for (int i = 0; i < numEntries; i++)
{
    total_wt += wait[i];
}
cout << "Priority Scheduling:" << endl;
cout << "Average Waiting Time: " << total_wt / numEntries << endl;
cout << "Average Turnaround Time: " << total_tat / numEntries << endl;
}

int main()
{
    int n;
    cout << "Enter your choose : " << endl;
    cin >> n;

    switch (n)
    {
    case 1:
        Priority();
        break;
    case 2:
        Round_robin();
        break;
    }
}

```



```

default:
    cout << "Choose given one." << endl;
}
return 0;
}

```

## Output :-

```

PS D:\Programming\C++\c++_pro> cd "d:\Pro
1. Priority
2. Round robin
Enter your choose :
1
Enter the number of map entries: 5
Enter the 1 element BurstTime : 3
Enter the 1 element Priority : 5
Enter the 2 element BurstTime : 6
Enter the 2 element Priority : 4
Enter the 3 element BurstTime : 7
Enter the 3 element Priority : 2
Enter the 4 element BurstTime : 9
Enter the 4 element Priority : 3
Enter the 5 element BurstTime : 8
Enter the 5 element Priority : 1
Priority Scheduling:
Average Waiting Time: 11
Average Turnaround Time: 17.6

```

```

PS D:\Programming\C++> cd "d:\Programming\C++\c++_pro
1. Priority
2. Round robin
Enter your choose :
2
Enter the TS: 5
Enter the number of map entries: 5
Enter the 1 element BurstTime : 3
Enter the 2 element BurstTime : 6
Enter the 3 element BurstTime : 4
Enter the 4 element BurstTime : 5
Enter the 5 element BurstTime : 2
Round Robin Scheduling:
Average Waiting Time: 10.2
Average Turnaround Time: 14.2

```

## Assignment – 5

Aim : implement peterson's solution for 2 process P0 and P1 in process synchronization

### CODE: - Peterson's solution

```
# Initialize global variables
flag = [False, False]
Turn = 0
b = 0
a = 0
B = 0
def P0():
    global a, b # Specify that you want to modify global 'a' and 'b' within this function
    while True:
        flag[0] = True
        Turn = 1
        while Turn == 1 and flag[1] == True:
            pass
        # Critical Section
        a = a + 1
        print("Context Switch karna hai?" + "Press 1 for Yes" + "Press 0 for No")
        count = int(input("Enter Your choice : "))
        if count == 1:
            P1()
        else:
            pass
        flag[0] = False
        b = b + 1
        print("b is " + str(b))
        P1()
    return b

def P1():
    global a, B # Specify that you want to modify global 'a' and 'B' within this function
    while True:
        flag[1] = True
```

```
Turn = 0
while Turn == 0 and flag[0] == True:
    pass
    # Critical Section
    a = a + 1
    flag[1] = False
    B = B + 1
    print("B is " + str(B))
    return B

t = 1
while t:
    P0()
    t = t - 1
```

## **Output :-**

```
D:\Programming\ML\Python_projects\project_code\venv\Scripts\python.exe
Context Switch karna hai?Press 1 for YesPress 0 for No
Enter Your choice : 0
b is 1
B is 1
```

## **Assignment – 6**

Aim : a) implement binary semaphore to solve critical section problem for 2 process - process P0 and process P1

b) implement counting semaphore to solve critical section problem to achieve multiprocess synchronization

### **CODE :- 6(A)**

```
import time

class Semaphore:
    def __init__(self):
        self.value = 1

    def wait(self):
        while self.value == 0:
            pass
        self.value = 0

    def signal(self):
        self.value = 1

def critical_section(process_id):
    print(f"Process {process_id} is in critical section.")

def process(semaphore, process_id):
    while True:
        # Non-critical section
        print(f"Process {process_id} is in non-critical section.")

        # Entry section
        semaphore.wait()
        time.sleep(1)

        # Critical section
        critical_section(process_id)
```

```

    # Exit section
    semaphore.signal()

    # Remainder section
    break

if __name__ == "__main__":
    semaphore = Semaphore()

    process(semaphore, 0)
    time.sleep(1)
    process(semaphore, 1)

```

## **Output :-**

```

D:\Programming\ML\Python_projects\project_code\venv\Scripts\python.exe
Process 0 is in non-critical section.
Process 0 is in critical section.
Process 1 is in non-critical section.
Process 1 is in critical section.

```

---

## **CODE :- 6(B)**

```

import heapq
# Global Variable to track the Processes going into Critical Section
COUNTER=1

class Semaphore:
    def __init__(self,value):
        # Value of the Semaphore passed to the Constructor
        self.value=value
        # The Waiting queue which will be using the heapq module of Python
        self.q=list()

    def getSemaphore(self):
        ''' Function to print the Value of the Semaphore Variable '''
        print(f'Semaphore Value: {self.value}')

    def block(process):
        print(f'Process {process} Blocked.')

```

```

def wakeup(process):
    print(f'Process {process} Waked Up and Completed it's work.')

def P(s):
    global COUNTER
    s.value=s.value-1
    if(s.value<0):
        heapq.heappush(s.q,COUNTER)
        block(COUNTER)
    else:
        print(f'Process {COUNTER} gone inside the Critical Section.')
        COUNTER+=1
        return

def V(s):
    global COUNTER
    s.value=s.value+1
    if(s.value<=0):
        p=heapq.heappop(s.q)
        wakeup(p)
        COUNTER-=1
    else:
        print(f'Process {COUNTER} completed it's work.')
        COUNTER-=1
        return

```

# Can Pass the Value of the Counting Semaphore to the Class Constructor

# Example for Counting Semaphore value as 2

```

s1=Semaphore(2)
s1.getSemaphore()
P(s1)
s1.getSemaphore()
P(s1)
s1.getSemaphore()
P(s1)
s1.getSemaphore()
V(s1)
s1.getSemaphore()
V(s1)
s1.getSemaphore()
V(s1)

```

s1.getSemaphore()

### **Output :-**

```
D:\Programming\ML\Python_projects\project_code\venv\Scripts\python.exe
Semaphore Value: 2
Process 1 gone inside the Critical Section.
Semaphore Value: 1
Process 2 gone inside the Critical Section.
Semaphore Value: 0
Process 3 Blocked.
Semaphore Value: -1
Process 3 Waked Up and Completed it's work.
Semaphore Value: 0
Process 2 completed it's work.
Semaphore Value: 1
Process 1 completed it's work.
Semaphore Value: 2
```

---

## Assignment – 7

**Aim :implement banker's algorithm using any programming language.**

### CODE :- 7

```
#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");
    }
}
return 0;
}
```

**Output :-**

```

Enter number of processes: 5
Enter number of resources: 3
Enter Claim Vector: 10 5 7
Enter Allocated Resource Table:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2

Enter Maximum Claim Table:
7 5 3
3 2 2
9 0 2
4 2 2
5 3 3

The Claim Vector is:      10      5
The Allocated Resource Table:
      0      1      0
      2      0      0
      3      0      2
      2      1      1
      0      0      2

```

```

The Maximum Claim Table:
      7      5      3
      3      2      2
      9      0      2
      4      2      2
      5      3      3

Allocated resources:      7      2
Available resources:      3      3

Process2 is executing      []

The process is in safe state
Available vector:      5      3

Process4 is executing

The process is in safe state
Available vector:      7      4

Process1 is executing

The process is in safe state
Available vector:      7      5

Process3 is executing

The process is in safe state
Available vector:      10      5

Process5 is executing

The process is in safe state
Available vector:      10      5
○ PS D:\Programming\C++\c++_pro>

```

## EXPERIMENT: -8

**Aim: - first, best and worst fit for fix and variable size memory allocation.**

**CODE: -**

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

using namespace std;

void First_Fit_fix()
{
    int n;
    cout << "Enter no of process : " << endl;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Process " << i + 1 << ": ";
        cin >> arr[i];
    }
    // 357 210 468 491
    int free_array[] = {200, 400, 600, 500, 300, 250};

    for (int i = 0; i < n; i++)
    {
        bool la = false;
        for (int j = 0; j < 6; j++)
        {
            if (free_array[j] >= arr[i])
            {
                cout << arr[i] << "(" << i + 1 << ")"
                     << " Store in " << j + 1 << "(" << free_array[j] << ")"
                     << " Block" << endl;
                free_array[j] -= arr[i];
                break;
            }
        }
    }
}
```

```

bool sortByValue(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.second < b.second;
}

void Best_Fit_fix()
{
    int n;
    cout << "Enter no of process : " << endl;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Process " << i + 1 << ": ";
        cin >> arr[i];
    }
    // 357 210 468 491
    int free_array[] = {200, 400, 600, 500, 300, 250};
    map<int, int> myMap;
    for (int i = 0; i < 6; i++)
    {
        myMap[i] = free_array[i];
    }

    vector<pair<int, int>> vec(myMap.begin(), myMap.end());

    for (int i = 0; i < n; i++)
    {
        for (auto &pair : vec)
        {
            sort(vec.begin(), vec.end(), sortByValue);

            if (pair.second >= arr[i])
            {
                cout << arr[i] << "(" << i + 1 << ")"
                    << " Store in " << pair.first + 1 << "(" << pair.second << ")"
                    << " Block" << endl;
                pair.second = 0;
                break;
            }
        }
    }
}

```

```

bool sortByValue1(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.second > b.second;
}

void Worst_Fit_fix()
{
    int n;
    cout << "Enter no of process : " << endl;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Process " << i + 1 << ": ";
        cin >> arr[i];
    }
    // 357 210 468 491
    int free_array[] = {200, 400, 600, 500, 300, 250};
    map<int, int> myMap;
    for (int i = 0; i < 6; i++)
    {
        myMap[i] = free_array[i];
    }

    vector<pair<int, int>> vec(myMap.begin(), myMap.end());

    for (int i = 0; i < n; i++)
    {
        for (auto &pair : vec)
        {
            sort(vec.begin(), vec.end(), sortByValue1);

            if (pair.second >= arr[i])
            {
                cout << arr[i] << "(" << i + 1 << ")"
                    << " Store in " << pair.first + 1 << "(" << pair.second << ")"
                    << " Block" << endl;
                pair.second = 0;
                break;
            }
        }
    }
}

```

```

}

void First_Fit()
{
    int n;
    cout << "Enter no of process : " << endl;
    cin >> n;
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        cout << "Process " << i + 1 << ": ";
        cin >> arr[i];
    }
    // 357 210 468 491
    int free_array[] = {0, 150, 0, 350, 0};

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < 5; j++)
        {
            if (free_array[j] >= arr[i])
            {
                cout << arr[i] << "(" << i + 1 << ")"
                    << " Store in " << j + 1 << "(" << free_array[j] << ")"
                    << " Block" << endl;
                free_array[j] -= arr[i];
                break;
            }
        }
    }
}

bool sortByValue2(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.second < b.second;
}

void Best_Fit()
{
    int n;
    cout << "Enter no of process : " << endl;
    cin >> n;
    int arr[n];

```



```

for (int i = 0; i < n; i++)
{
    cout << "Process " << i + 1 << ": ";
    cin >> arr[i];
}
// 357 210 468 491
int free_array[] = {0, 150, 0, 350, 0};
map<int, int> myMap;
for (int i = 0; i < 5; i++)
{
    myMap[i] = free_array[i];
}
vector<pair<int, int>> vec(myMap.begin(), myMap.end());

for (int i = 0; i < n; i++)
{
    for (auto &pair : vec)
    {
        sort(vec.begin(), vec.end(), sortByValue2);

        if (pair.second >= arr[i])
        {
            cout << arr[i] << "(" << i + 1 << ")"
                << " Store in " << pair.first + 1 << "(" << pair.second << ")"
                << " Block" << endl;
            pair.second -= arr[i];
            break;
        }
    }
}

bool sortByValue3(const pair<int, int> &a, const pair<int, int> &b)
{
    return a.second > b.second;
}

void Worst_Fit()
{
    int n;
    cout << "Enter no of process : " << endl;

```

```

cin >> n;
int arr[n];
for (int i = 0; i < n; i++)
{
    cout << "Process " << i + 1 << ": ";
    cin >> arr[i];
}
// 357 210 468 491
int free_array[] = {0, 150, 0, 350, 0};
map<int, int> myMap;
for (int i = 0; i < 5; i++)
{
    myMap[i] = free_array[i];
}
vector<pair<int, int>> vec(myMap.begin(), myMap.end());

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

    for (auto &pair : vec)
    {

        sort(vec.begin(), vec.end(), sortByValue3);

        if (pair.second >= arr[i])
        {
            cout << arr[i] << "(" << i + 1 << ")"
                << " Store in " << pair.first + 1 << "(" << pair.second << ")"
                << " Block" << endl;
            pair.second -= arr[i];
            break;
        }
    }
}

}

int main()
{
    do
    {
        cout << "**** SIMULATION OF FIXED PARTITIONING MEMORY
MANAGEMENT SCHEMES ****" << endl;
        cout << "Options : " << endl;

```

```
    cout << "1. First Fit Fix" << endl;
    cout << "2. Best Fit Fix" << endl;
    cout << "3. Worst Fit Fix" << endl;
    cout << "4. First Fit" << endl;
    cout << "5. Best Fit" << endl;
    cout << "6. Worst Fit" << endl;
    cout << "7. Exit" << endl;
    cout << "Enter your choice : " << endl;
    int ch;
    cin >> ch;
    switch (ch)
    {
    case 1:
        First_Fit_fix();
        break;
    case 2:
        Best_Fit_fix();
        break;
    case 3:
        Worst_Fit_fix();
        break;
    case 4:
        First_Fit();
        break;
    case 5:
        Best_Fit();
        break;
    case 6:
        Worst_Fit();
        break;
    case 7:
        exit(0);
    default:
        cout << "invalid choice " << endl;
    }

} while (true);

return 0;
}
```

## OUTPUT: -

```
*** SIMULATION OF FIXED PARTITIONING MEMORY MANAGEMENT SCHEMES ***  
Options :  
1. First Fit Fix  
2. Best Fit Fix  
3. Worst Fit Fix  
4. First Fit  
5. Best Fit  
6. Worst Fit  
7. Exit  
Enter your choice :
```

```
Enter your choice :  
1  
Enter no of process :  
4  
Process 1: 357  
Process 2: 210  
Process 3: 468  
Process 4: 491  
357(1) Store in 2(400) Block  
210(2) Store in 3(600) Block  
468(3) Store in 4(500) Block
```

```
Enter your choice :  
2  
Enter no of process :  
4  
Process 1: 357  
Process 2: 210  
Process 3: 468  
Process 4: 491  
357(1) Store in 2(400) Block  
210(2) Store in 6(250) Block  
468(3) Store in 4(500) Block  
491(4) Store in 3(600) Block
```

```
Enter your choice :  
3  
Enter no of process :  
4  
Process 1: 357  
Process 2: 210  
Process 3: 468  
Process 4: 491  
357(1) Store in 3(600) Block  
210(2) Store in 4(500) Block
```

```
Enter your choice :  
4  
Enter no of process :  
4  
Process 1: 300  
Process 2: 25  
Process 3: 125  
Process 4: 50  
300(1) Store in 4(350) Block  
25(2) Store in 2(150) Block  
125(3) Store in 2(125) Block  
50(4) Store in 4(50) Block
```

```
Enter your choice :  
5  
Enter no of process :  
4  
Process 1: 300  
Process 2: 25  
Process 3: 125  
Process 4: 50  
300(1) Store in 4(350) Block  
25(2) Store in 4(50) Block  
125(3) Store in 2(150) Block
```

```
Enter your choice :  
6  
Enter no of process :  
4  
Process 1: 300  
Process 2: 25  
Process 3: 125  
Process 4: 50  
300(1) Store in 4(350) Block  
25(2) Store in 2(150) Block  
125(3) Store in 2(125) Block  
50(4) Store in 4(50) Block
```

## CONCLUSION: -

In Given Example:

In fixed sized memory allocation the best case is best fit and worst case is worst fit.

In variable sized memory allocation the best case is the worst fit and best case is the first fit.