Program Code:

```
#!/bin/bash
pwd
date
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./01.sh
/home/rinoy2002/Desktop/OS Lab/Day 3 Shell Scripting
Saturday 18 September 2021 10:36:41 PM IST

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ |
```

Program Code:

```
#!/bin/bash
a=20
b=10
sum=$(($a + $b))
diff=$(($a - $b))
product=$(($a * $b))
division=$(($a / $b))
echo " Sum : $sum"
echo " Difference : $diff"
echo " Product : $product"
echo " Division : $division"
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./02.sh
 Sum : 30
 Difference : 10
 Product : 200
 Division : 2
```

Program Code:

```
#!/bin/bash
echo "Count of Arguments: $#"
echo $*
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./03.sh
Count of Arguments: 0


┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./03.sh how are you
Count of Arguments: 3
how are you

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./03.sh hello
Count of Arguments: 1
hello
```

Program Code:

```
#!/bin/bash
if [ $# == 3 ]
then
echo "No. of Arguments is 3"
else
echo "No. of Arguments is not 3"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./04.sh
No. of Arguments is not 3

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./04.sh hello
No. of Arguments is not 3

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./04.sh hello how are
No. of Arguments is 3

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./04.sh hello how are you
No. of Arguments is not 3
```

Program Code:

```
#! /bin/bash
if [ $# -lt 2 ]
then
echo 'pass 2 Strings'
elif [ $1 == $2 ]
then
echo "both strings are same"
elif [ $1 != $2 ]
then
"both strings are different"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./05.sh hello hello
both strings are same

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./05.sh hello helle
both strings are different

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ |
```

Program Code:

```
#!/bin/bash
FILE=$1
if [ -e "$FILE" ]
then
echo "file found"
else
echo "file not found"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./06a.sh rinoy
file not found

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./06a.sh test
file found

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ |
```

Program Code:

```
#!/bin/bash
FILE=$1
if [ -e "$FILE" ]
then
echo "file found"
mv $1 $2
echo "file renamed"
else
echo "file not found"
fi
echo "No. of Arguments is $#"
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./06b.sh test rinoy
file found
file renamed
No. of Arguments is 2

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./06b.sh royal test
file not found
No. of Arguments is 2
```

Program Code:

```
#!/bin/bash
echo "enter the file name :"
read file
if [ -e "$file" ]
then
echo "enter the word :"
read word
isfound=$(grep -cw $word $file)
if [ $isfound -gt 0 ]
then
echo "word found"
else
echo "word not found"
fi
else
echo "file not found"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./07.sh
enter the file name :
test
enter the word :
house
word found

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./07.sh
enter the file name :
test
enter the word :
yellow
word not found
```

Program Code:

```
#!/bin/bash
if [ -e "file1" ]
then
if [ -e "file2" ]
then
cat file1>>file2
echo "file2 exists,file1 appended to file2"
else
cp file1 file2
echo "file2 doesn't exists,file1 copied to file2"
fi
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./08.sh
file2 exists,file1 appended to file2

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ cat file2
my name is rinoy kuriyakose
hello how are you


┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ cat file1
hello how are you
```

Program Code:

```
#!/bin/bash
if [ -e "firstfile" ] && [ -e "secondfile" ]
then
head firstfile>>$1
head secondfile>>$1
cat $1
else
echo "file not found"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./09.sh thirdfile
red
green
yellow
orange
voilet
indigo
blue
black
white
pink
tejas
rinoy
roshan
amit
rahul
shubo
royal
reena
beena
seena
```

Program Code:

```
#!/bin/bash
echo "matches found are :"
find -name "${1}*" -type f | wc -l
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./10.sh t
matches found are :
2

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./10.sh f
matches found are :
3

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./10.sh r
matches found are :
1
```

Program Code:

```
#!/bin/bash
head -n1 test | tr 'aeiou' 'AEIOU' > output
cat output
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./11.sh
my nAmE Is rInOy kUrIyAkOsE

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./11.sh
I Am lIvIng In kErAlA
```

Program Code:

```
#!/bin/bash
echo "enter the name :"
read name
echo "reverse : "
echo $name | rev
echo "length : "
echo ${#name}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./12.sh
enter the name :
rinoy
reverse :
yonir
length :
5

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./12.sh
enter the name :
kuriyakose
reverse :
esokayiruk
length :
10
```

Program Code:

```
#!/bin/bash
if [ -e "school.dat"]
then
cat school.dat | sort -r -k3
else
echo "file not found"
fi
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ cat school.dat
Roll No    Name          Marks
 1         Rinoy          99
 2         Kuriyakose     89
 3         Reena          91
 4         Royal          78
 5         Tejas          94

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./13.sh
Roll No    Name          Marks
 1         Rinoy          99
 5         Tejas          94
 3         Reena          91
 2         Kuriyakose     89
 4         Royal          78
```

Program Code:

```
#!/bin/bash
echo "enter value n :"
read n
for ((i=1;i<=n;i++))
do
sum=$((sum + i))
done
echo "sum : $sum"
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./14.sh
enter value n :
10
sum : 55

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./14.sh
enter value n :
100
sum : 5050
```

Program Code:

```
#!/bin/bash
echo " MENU "
echo "1.Sunday"
echo "2.Monday"
echo "3.Tuesday"
echo "4.Wednesday"
echo "5.Thursday"
echo "6.Friday"
echo "7.Saturday"
echo "enter the choice : "
read option
case $option in
1)
  echo "Sunday"
;;
2)
  echo "Monday"
;;
3)
  echo "Tuesday"
;;
4)
  echo "Wednesday"
```

```
;;
5)
  echo "Thursday"
;;
6)
  echo "Friday"
;;
7)
  echo "Saturday"
;;
*)
  echo "Invalid Input"
;;
esac
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./15.sh
 MENU
1.Sunday
2.Monday
3.Tuesday
4.Wednesday
5.Thursday
6.Friday
7.Saturday
enter the choice :
6
Friday

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 3 Shell Scripting]
└─$ ./15.sh
 MENU
1.Sunday
2.Monday
3.Tuesday
4.Wednesday
5.Thursday
6.Friday
7.Saturday
enter the choice :
3
Tuesday
```

Program Code:

```
#!/bin/bash
echo -e " \nOperations:"
echo " 1)Addition: +"
echo " 2)Subtraction: -"
echo " 3)Multiplication: \*"
echo " 4)Division: /"
echo " 5)Modulus: %"
echo " 6)Exponent: ^"
echo " 7)Functions: sin() cos() tan() log() etc"
echo " 8)Parentheses: ()"
echo -ne " \nEnter the expression: "
read a
echo -e " \nResult: `awk "BEGIN{print $a}"`\n"
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 4 Scientific Calculator]
└─$ ./scientific-calculator.sh

Operations:
 1)Addition: +
 2)Subtraction: -
 3)Multiplication: \*
 4)Division: /
 5)Modulus: %
 6)Exponent: ^
 7)Functions: sin() cos() tan() log() etc
 8)Parentheses: ()

Enter the expression: (1+5)/3+10-2^3

Result: 4
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 4 Scientific Calculator]
└─$ ./scientific-calculator.sh

Operations:
 1)Addition: +
 2)Subtraction: -
 3)Multiplication: \*
 4)Division: /
 5)Modulus: %
 6)Exponent: ^
 7)Functions: sin() cos() tan() log() etc
 8)Parentheses: ()

Enter the expression: sin(60)

Result: -0.304811
```

Program Code:

```
#include<stdio.h>
void main()
 {
    struct process{
        int no;
        float bt;
        float at;
        float tat;
        float wt;
    };
    int n,i,j;
    float time=0,temp1,temp2,temp3,avgtat,avgwt,stat=0,swt=0;
    printf("no. of processes :");
    scanf("%d",&n);
    struct process p[n];
    printf("burst time for the processes :\n");
    for(i=0;i<n;i++){
        printf("p[%d] : ",i+1);
        scanf("%f",&p[i].bt);
    }
    printf("arrival time for the processes :\n");
    for(i=0;i<n;i++){
        printf("p[%d] : ",i+1);
        scanf("%f",&p[i].at);
    }
    for(i=0;i<n;i++){
        p[i].no = i+1;
    }
   for(i=0;i<n;i++){
        for(j=0;j<n-i-1;j++){
           if(p[j].at>p[j+1].at){
               temp1 = p[j].no;
               temp2 = p[j].bt;
               temp3 = p[j].at;
               p[j].no = p[j+1].no;
               p[j].bt = p[j+1].bt;
               p[j].at = p[j+1].at;
               p[j+1].no = temp1;
               p[j+1].bt = temp2;
               p[j+1].at = temp3;
           }
        }
    }
    for(i=0;i<n;i++){
        if(time<=p[i].at){
            time =p[i].at;
        }
        time = time + p[i].bt;
        p[i].tat = time-p[i].at;
        stat = stat + p[i].tat;
```

```
        p[i].wt = p[i].tat - p[i].bt;
        swt = swt + p[i].wt;
    }
    avgtat = stat/n;
    avgwt = swt/n;
      printf("\n  process | burst time  |  arrival time  |  turn-around time  |  waiting time \n");
        printf("  ---------------------------------------------------------------------------------\n");
    for(i=0;i<n;i++){
        printf("  p[%d] : ",p[i].no);
        printf("   %.3f ns",p[i].bt);
        printf("       %.3f ns",p[i].at);
        printf("           %.3f ns",p[i].tat);
        printf("           %.3f ns\n\n",p[i].wt);
    }
    printf(" average turn-around time : %.3f ns\n",avgtat);
    printf(" average waiting time     : %.3f ns\n",avgwt);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./FCFSScheduling
no. of processes :4
burst time for the processes :
p[1] : 5
p[2] : 6
p[3] : 4
p[4] : 8
arrival time for the processes :
p[1] : 0
p[2] : 0
p[3] : 0
p[4] : 0

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------------
  p[1] :    5.000 ns       0.000 ns            5.000 ns            0.000 ns

  p[2] :    6.000 ns       0.000 ns            11.000 ns            5.000 ns

  p[3] :    4.000 ns       0.000 ns            15.000 ns            11.000 ns

  p[4] :    8.000 ns       0.000 ns            23.000 ns            15.000 ns

 average turn-around time : 13.500 ns
 average waiting time     : 7.750 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./FCFSScheduling
no. of processes :4
burst time for the processes :
p[1] : 9
p[2] : 1
p[3] : 8
p[4] : 2
arrival time for the processes :
p[1] : 10
p[2] : 0
p[3] : 2
p[4] : 7

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------
  p[2] :    1.000 ns       0.000 ns            1.000 ns            0.000 ns

  p[3] :    8.000 ns       2.000 ns            8.000 ns            0.000 ns

  p[4] :    2.000 ns       7.000 ns            5.000 ns            3.000 ns

  p[1] :    9.000 ns       10.000 ns            11.000 ns            2.000 ns

 average turn-around time : 6.250 ns
 average waiting time     : 1.250 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./FCFSScheduling
no. of processes :4
burst time for the processes :
p[1] : 2
p[2] : 4
p[3] : 7
p[4] : 8
arrival time for the processes :
p[1] : 0
p[2] : 0
p[3] : 0
p[4] : 1

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------
  p[1] :    2.000 ns       0.000 ns            2.000 ns            0.000 ns

  p[2] :    4.000 ns       0.000 ns            6.000 ns            2.000 ns

  p[3] :    7.000 ns       0.000 ns            13.000 ns            6.000 ns

  p[4] :    8.000 ns       1.000 ns            20.000 ns            12.000 ns

 average turn-around time : 10.250 ns
 average waiting time     : 5.000 ns
```

Program Code:

```
include<stdio.h>
void main()
 {
    struct process{
        int no;
        float bt;
        float at;
        float tat;
        float wt;
    };
    int n,i,j,d,k=1;
    float time=0,temp1,temp2,temp3,avgtat,avgwt,stat=0,swt=0,min,sum,btime;
    printf("no. of processes :");
    scanf("%d",&n);
    struct process p[n];
    printf("burst time for the processes :\n");
    for(i=0;i<n;i++){
        printf("p[%d] : ",i+1);
        scanf("%f",&p[i].bt);
    }
    printf("arrival time for the processes :\n");
    for(i=0;i<n;i++){
        printf("p[%d] : ",i+1);
        scanf("%f",&p[i].at);
    }
    for(i=0;i<n;i++){
        p[i].no = i+1;
    }
    for(i=0;i<n;i++){
        for(j=0;j<n-i-1;j++){
           if(p[j].bt>p[j+1].bt){
               temp1 = p[j].no;
               temp2 = p[j].bt;
               temp3 = p[j].at;
               p[j].no = p[j+1].no;
               p[j].bt = p[j+1].bt;
               p[j].at = p[j+1].at;
               p[j+1].no = temp1;
               p[j+1].bt = temp2;
               p[j+1].at = temp3;
           }
        }
    }
    for(i=0;i<n;i++){
        for(j=0;j<n-i-1;j++){
           if(p[j].at>p[j+1].at){
               temp1 = p[j].no;
               temp2 = p[j].bt;
               temp3 = p[j].at;
               p[j].no = p[j+1].no;
```

```
            p[j].bt = p[j+1].bt;
            p[j].at = p[j+1].at;
            p[j+1].no = temp1;
            p[j+1].bt = temp2;
            p[j+1].at = temp3;
        }
    }
}
for(j=0;j<n;j++){
    btime=btime+p[j].bt;
    min=p[k].bt;
    for(i=k;i<n;i++){
        if ((btime>=p[i].at) && (p[i].bt<min)){
            min = p[i].bt;
            temp1 = p[i].no;
            temp2 = p[i].bt;
            temp3 = p[i].at;
            p[i].no = p[k].no;
            p[i].bt = p[k].bt;
            p[i].at = p[k].at;
            p[k].no = temp1;
            p[k].bt = temp2;
            p[k].at = temp3;
        }
    }
    k++;
}
sum = p[0].at;
for(i=0;i<n;i++){
    if(sum<p[i].at){
        sum=p[i].at;
    }
    sum = sum + p[i].bt;
p[i].tat=sum-p[i].at;
p[i].wt=p[i].tat-p[i].bt;
stat=stat+p[i].tat;
swt=swt+p[i].wt;
}
avgtat = stat/n;
avgwt = swt/n;
printf("\n  process | burst time  |  arrival time  |  turn-around time  |  waiting time \n");
    printf("  ---------------------------------------------------------------------------------\n");
for(i=0;i<n;i++){
    printf("  p[%d] : ",p[i].no);
    printf("   %.3f ns",p[i].bt);
    printf("       %.3f ns",p[i].at);
    printf("            %.3f ns",p[i].tat);
    printf("            %.3f ns\n\n",p[i].wt);
}
printf(" average turn-around time : %.3f ns\n",avgtat);
printf(" average waiting time     : %.3f ns\n",avgwt);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./SJFScheduling
no. of processes :4
burst time for the processes :
p[1] : 6
p[2] : 1
p[3] : 5
p[4] : 2
arrival time for the processes :
p[1] : 0
p[2] : 1
p[3] : 1
p[4] : 2

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------
  p[1] :    6.000 ns       0.000 ns            6.000 ns            0.000 ns

  p[2] :    1.000 ns       1.000 ns            6.000 ns            5.000 ns

  p[4] :    2.000 ns       2.000 ns            7.000 ns            5.000 ns

  p[3] :    5.000 ns       1.000 ns            13.000 ns            8.000 ns

 average turn-around time : 8.000 ns
 average waiting time     : 4.500 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./SJFScheduling
no. of processes :3
burst time for the processes :
p[1] : 10
p[2] : 3
p[3] : 1
arrival time for the processes :
p[1] : 0
p[2] : 1
p[3] : 2

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------
  p[1] :    10.000 ns       0.000 ns            10.000 ns            0.000 ns

  p[3] :    1.000 ns       2.000 ns            9.000 ns            8.000 ns

  p[2] :    3.000 ns       1.000 ns            13.000 ns            10.000 ns

 average turn-around time : 10.667 ns
 average waiting time     : 6.000 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./SJFScheduling
no. of processes :4
burst time for the processes :
p[1] : 6
p[2] : 1
p[3] : 5
p[4] : 2
arrival time for the processes :
p[1] : 0
p[2] : 1
p[3] : 1
p[4] : 2

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------
  p[1] :    6.000 ns       0.000 ns            6.000 ns            0.000 ns

  p[2] :    1.000 ns       1.000 ns            6.000 ns            5.000 ns

  p[4] :    2.000 ns       2.000 ns            7.000 ns            5.000 ns

  p[3] :    5.000 ns       1.000 ns            13.000 ns            8.000 ns

 average turn-around time : 8.000 ns
 average waiting time     : 4.500 ns
```

Program Code:

```
#include<stdio.h>
void main()
 {
   struct process{
      int no;
      float bt;
      float at;
      float tat;
      float wt;
   };
   int n,i,j,quantum,count,var;
   float time=0,temp1,temp2,temp3,temp4,avgtat,avgwt,stat=0,swt=0,sum;
   printf(" no. of processes :");
   scanf("%d",&n);
   struct process p[n];
   float temp[n];
   printf(" burst time for the processes :\n");
   for(i=0;i<n;i++){
      printf(" p[%d] : ",i+1);
      scanf("%f",&p[i].bt);
      temp[i]=p[i].bt;
   }
   printf(" arrival time for the processes :\n");
   for(i=0;i<n;i++){
      printf(" p[%d] : ",i+1);
      scanf("%f",&p[i].at);
```

```
 }
 for(i=0;i<n;i++){
    p[i].no = i+1;
 }
 printf(" time quantum for the processes: ");
 scanf("%d", &quantum);
for(i=0;i<n;i++){
    for(j=0;j<n-i-1;j++){
       if(p[j].at>p[j+1].at){
          temp1 = p[j].no;
          temp2 = p[j].bt;
          temp3 = p[j].at;
          temp4 = temp[j];
          p[j].no = p[j+1].no;
          p[j].bt = p[j+1].bt;
          p[j].at = p[j+1].at;
          temp[j] = temp[j+1];
          p[j+1].no = temp1;
          p[j+1].bt = temp2;
          p[j+1].at = temp3;
          temp[j+1] = temp4;
       }
    }
 }
 sum=p[0].at;
 var=n;
 for( i = 0; var!=0; ){
    if(temp[i] <= quantum && temp[i] > 0){
       sum = sum + temp[i];
       temp[i] = 0;
       count=1;
    }
    else if(temp[i] > 0){
       temp[i] = temp[i] - quantum;
       sum = sum + quantum;
    }
    if(temp[i]==0 && count==1){
       var--;
       p[i].tat=sum-p[i].at;
       p[i].wt=sum-p[i].at-p[i].bt;
       swt = swt+p[i].wt;
       stat = stat+p[i].tat;
       count =0;
    }
    if(i==n-1){
       i=0;
    }
    else if(p[i+1].at<=sum){
       i++;
    }
    /*else if(p[i+1].at>=sum){
       sum=p[i+1].at;
```

```
        i++;
      }*/
      else{
        i=0;
      }
    }
    avgtat = stat/n;
    avgwt = swt/n;
      printf("\n  process | burst time  |  arrival time  |  turn-around time  |  waiting time \n");
      printf("  ---------------------------------------------------------------------------------\n");
    for(i=0;i<n;i++){
      printf("  p[%d] : ",p[i].no);
      printf("   %.3f ms",p[i].bt);
      printf("       %.3f ms",p[i].at);
      printf("            %.3f ms",p[i].tat);
      printf("            %.3f ms\n\n",p[i].wt);
    }
    printf(" average turn-around time : %.3f ms\n",avgtat);
    printf(" average waiting time     : %.3f ms\n",avgwt);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./RRScheduling
 no. of processes :4
 burst time for the processes :
 p[1] : 5
 p[2] : 6
 p[3] : 3
 p[4] : 8
 arrival time for the processes :
 p[1] : 0
 p[2] : 0
 p[3] : 0
 p[4] : 0
 time quantum for the processes: 2

  process | burst time  |  arrival time  |  turn-around time  |  waiting time
  ---------------------------------------------------------------------------------
  p[1] :    5.000 ms       0.000 ms            16.000 ms            11.000 ms

  p[2] :    6.000 ms       0.000 ms            18.000 ms            12.000 ms

  p[3] :    3.000 ms       0.000 ms            13.000 ms            10.000 ms

  p[4] :    8.000 ms       0.000 ms            22.000 ms            14.000 ms

 average turn-around time : 17.250 ms
 average waiting time     : 11.750 ms
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./RRScheduling
no. of processes :3
burst time for the processes :
p[1] : 8
p[2] : 4
p[3] : 6
arrival time for the processes :
p[1] : 0
p[2] : 0
p[3] : 0
time quantum for the processes: 4

 process | burst time  |  arrival time  |  turn-around time  |  waiting time
 ---------------------------------------------------------------------------
 p[1] :    8.000 ms       0.000 ms            16.000 ms            8.000 ms

 p[2] :    4.000 ms       0.000 ms            8.000 ms            4.000 ms

 p[3] :    6.000 ms       0.000 ms            18.000 ms            12.000 ms

average turn-around time : 14.000 ms
average waiting time     : 8.000 ms
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./RRScheduling
no. of processes :4
burst time for the processes :
p[1] : 13
p[2] : 9
p[3] : 5
p[4] : 7
arrival time for the processes :
p[1] : 0
p[2] : 2
p[3] : 2
p[4] : 3
time quantum for the processes: 3

 process | burst time  |  arrival time  |  turn-around time  |  waiting time
 ---------------------------------------------------------------------------
 p[1] :    13.000 ms       0.000 ms            34.000 ms            21.000 ms

 p[2] :    9.000 ms       2.000 ms            27.000 ms            18.000 ms

 p[3] :    5.000 ms       2.000 ms            18.000 ms            13.000 ms

 p[4] :    7.000 ms       3.000 ms            27.000 ms            20.000 ms

average turn-around time : 26.500 ms
average waiting time     : 18.000 ms
```

Program Code:

```
#include<stdio.h>
void main()
 {
   struct process{
      int no;
      float bt;
      float at;
      float tat;
      float wt;
      int pr;
   };
   int n,i,j,d,k=1,min,temp4;
   float time=0,temp1,temp2,temp3,avgtat,avgwt,stat=0,swt=0,sum,btime;
   printf("no. of processes :");
   scanf("%d",&n);
   struct process p[n];
   printf("burst time for the processes :\n");
   for(i=0;i<n;i++){
      printf("p[%d] : ",i+1);
      scanf("%f",&p[i].bt);
   }
   printf("arrival time for the processes :\n");
   for(i=0;i<n;i++){
      printf("p[%d] : ",i+1);
      scanf("%f",&p[i].at);
   }
   printf("priority for the processes :\n");
   for(i=0;i<n;i++){
      printf("p[%d] : ",i+1);
      scanf("%d",&p[i].pr);
   }
   for(i=0;i<n;i++){
      p[i].no = i+1;
   }
   for(i=0;i<n;i++){
      for(j=0;j<n-i-1;j++){
         if(p[j].pr>p[j+1].pr){
            temp1 = p[j].no;
            temp2 = p[j].bt;
            temp3 = p[j].at;
            temp4 = p[j].pr;
            p[j].no = p[j+1].no;
            p[j].bt = p[j+1].bt;
            p[j].at = p[j+1].at;
            p[j].pr = p[j+1].pr;
            p[j+1].no = temp1;
            p[j+1].bt = temp2;
            p[j+1].at = temp3;
            p[j+1].pr = temp4;
         }
```

```
    }
}
for(i=0;i<n;i++){
    for(j=0;j<n-i-1;j++){
      if(p[j].at>p[j+1].at){
          temp1 = p[j].no;
          temp2 = p[j].bt;
          temp3 = p[j].at;
          temp4 = p[j].pr;
          p[j].no = p[j+1].no;
          p[j].bt = p[j+1].bt;
          p[j].at = p[j+1].at;
          p[j].pr = p[j+1].pr;
          p[j+1].no = temp1;
          p[j+1].bt = temp2;
          p[j+1].at = temp3;
          p[j+1].pr = temp4;
      }
    }
}
for(j=0;j<n;j++){
    btime=btime+p[j].bt;
    min=p[k].pr;
    for(i=k;i<n;i++){
       if ((btime>=p[i].at) && (p[i].pr<min)){
           min = p[i].pr;
           temp1 = p[i].no;
           temp2 = p[i].bt;
           temp3 = p[i].at;
           temp4 = p[i].pr;
           p[i].no = p[k].no;
           p[i].bt = p[k].bt;
           p[i].at = p[k].at;
           p[i].pr = p[k].pr;
           p[k].no = temp1;
           p[k].bt = temp2;
           p[k].at = temp3;
           p[k].pr = temp4;
       }
    }
    k++;
}
sum = p[0].at;
for(i=0;i<n;i++){
    if(sum<p[i].at){
       sum=p[i].at;
    }
    sum = sum + p[i].bt;
p[i].tat=sum-p[i].at;
p[i].wt=p[i].tat-p[i].bt;
stat=stat+p[i].tat;
swt=swt+p[i].wt;
```

```
    }
    avgtat = stat/n;
    avgwt = swt/n;
    printf("\n process | burst time  |  arrival time  |   priority   |  turn-around time  |  waiting time \n");
    printf(" ----------------------------------------------------------------------------------------------\n");
    for(i=0;i<n;i++){
        printf("  p[%d] : ",p[i].no);
        printf("   %.3f ns",p[i].bt);
        printf("       %.3f ns",p[i].at);
         printf("          %d",p[i].pr);
        printf("              %.3f ns",p[i].tat);
        printf("            %.3f ns\n\n",p[i].wt);
    }
    printf(" average turn-around time : %.3f ns\n",avgtat);
    printf(" average waiting time     : %.3f ns\n",avgwt);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./PriorityScheduling
no. of processes :4
burst time for the processes :
p[1] : 5
p[2] : 4
p[3] : 1
p[4] : 2
arrival time for the processes :
p[1] : 0
p[2] : 2
p[3] : 3
p[4] : 5
priority for the processes :
p[1] : 3
p[2] : 1
p[3] : 2
p[4] : 4

 process | burst time  |  arrival time  |   priority   |  turn-around time  |  waiting time
 ----------------------------------------------------------------------------------------------
  p[1] :    5.000 ns       0.000 ns          3              5.000 ns             0.000 ns

  p[2] :    4.000 ns       2.000 ns          1              7.000 ns             3.000 ns

  p[3] :    1.000 ns       3.000 ns          2              7.000 ns             6.000 ns

  p[4] :    2.000 ns       5.000 ns          4              7.000 ns             5.000 ns

 average turn-around time : 6.500 ns
 average waiting time     : 3.500 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./PriorityScheduling
no. of processes :4
burst time for the processes :
p[1] : 5
p[2] : 6
p[3] : 3
p[4] : 8
arrival time for the processes :
p[1] : 0
p[2] : 0
p[3] : 0
p[4] : 0
priority for the processes :
p[1] : 3
p[2] : 1
p[3] : 4
p[4] : 2

 process | burst time  |  arrival time  |   priority   |  turn-around time  |  waiting time
 ---------------------------------------------------------------------------------------------
  p[2] :    6.000 ns       0.000 ns          1              6.000 ns             0.000 ns

  p[4] :    8.000 ns       0.000 ns          2              14.000 ns             6.000 ns

  p[1] :    5.000 ns       0.000 ns          3              19.000 ns             14.000 ns

  p[3] :    3.000 ns       0.000 ns          4              22.000 ns             19.000 ns

 average turn-around time : 15.250 ns
 average waiting time     : 9.750 ns
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 5 Scheduling]
└─$ ./PriorityScheduling
no. of processes :3
burst time for the processes :
p[1] : 2
p[2] : 3
p[3] : 6
arrival time for the processes :
p[1] : 5
p[2] : 0
p[3] : 3
priority for the processes :
p[1] : 1
p[2] : 2
p[3] : 3

 process | burst time  |  arrival time  |   priority   |  turn-around time  |  waiting time
 ---------------------------------------------------------------------------------------------
  p[2] :    3.000 ns       0.000 ns          2              3.000 ns             0.000 ns

  p[3] :    6.000 ns       3.000 ns          3              6.000 ns             0.000 ns

  p[1] :    2.000 ns       5.000 ns          1              6.000 ns             4.000 ns

 average turn-around time : 5.000 ns
 average waiting time     : 1.333 ns
```

Program Code:

```
#include <stdio.h>
void main(){
    int n,m,i,j,k;
    printf("\n Enter total number of processes : ");
    scanf("%d", &n);
    printf("\n Enter total number of resources : ");
    scanf("%d", &m);
    int Alloc[n][m],Max[n][m],Need[n][m],Avail[m],Finish[n],Work[m],count=0,flag,seq[n],l=0;
    for(i = 0; i < n; i++){
        printf("\n Process %d\n", i);
        for(j = 0; j < m; j++){
            printf(" Allocation for resource %d : ", j+1 );
            scanf("%d", &Alloc[i][j]);
            printf(" Maximum for resource %d : ", j+1 );
            scanf("%d", &Max[i][j]);
        }
    }
    printf("\n Available Resources : \n");
    for (j = 0; j < m; j++){
        printf(" Resource %d : ", j+1);
        scanf("%d", &Avail[j]);
    }
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            Need[i][j] = Max[i][j] - Alloc[i][j];
        }
    }
    printf("\n Allocation Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Alloc[i][j]);
        }
        printf("\n");
    }
    printf("\n Maximum Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Max[i][j]);
        }
        printf("\n");
    }
    printf("\n Available Matrix : \n");
        for (j = 0; j < m; j++){
            printf(" %d ",Avail[j]);
        }

    printf("\n\n Need Matrix : \n");
    for(i = 0; i < n; i++){
        for (j = 0; j < m; j++){
            printf(" %d ",Need[i][j]);
```

```
        }
        printf("\n");
    }
    for(i=0;i<n;i++){
        Finish[i]=0;
    }
    for(j=0;j<m;j++){
        Work[j]=Avail[j];
    }
    for (k = 0; k < n; k++){
        for (i = 0; i < n; i++){
            if (Finish[i] == 0){
                flag = 0;
                for (j = 0; j < m; j++){
                    if (Need[i][j] > Work[j]){
                        flag = 1;
                    }
                }
                if (flag== 0 && Finish[i] == 0){
                    for (j = 0; j < m; j++){
                        Work[j] += Alloc[i][j];
                    }
                    Finish[i] = 1;
                    count++;
                    seq[l]=i;
                    l++;
                }
            }
        }
    }
    if(count==n){
        printf(" Safe Sequence : ");
        for(i=0;i<l;i++){
        printf(" P%d ",seq[i]);
        }
    }else{
        printf(" Deadlock Occurs ");
    }
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 6 Banker's Algorithm]
└─$ ./BankersAlgorithm

Enter total number of processes : 5

Enter total number of resources : 3

Process 0
Allocation for resource 1 : 0
Maximum for resource 1 : 7
Allocation for resource 2 : 1
Maximum for resource 2 : 5
Allocation for resource 3 : 0
Maximum for resource 3 : 3

Process 1
Allocation for resource 1 : 2
Maximum for resource 1 : 3
Allocation for resource 2 : 0
Maximum for resource 2 : 2
Allocation for resource 3 : 0
Maximum for resource 3 : 2

Process 2
Allocation for resource 1 : 3
Maximum for resource 1 : 9
Allocation for resource 2 : 0
Maximum for resource 2 : 0
Allocation for resource 3 : 2
Maximum for resource 3 : 2
```

```
Process 3
Allocation for resource 1 : 2
Maximum for resource 1 : 2
Allocation for resource 2 : 1
Maximum for resource 2 : 2
Allocation for resource 3 : 1
Maximum for resource 3 : 2

Process 4
Allocation for resource 1 : 0
Maximum for resource 1 : 4
Allocation for resource 2 : 0
Maximum for resource 2 : 3
Allocation for resource 3 : 2
Maximum for resource 3 : 3

Available Resources :
Resource 1 : 3
Resource 2 : 3
Resource 3 : 2

Allocation Matrix :
0  1  0
2  0  0
3  0  2
2  1  1
0  0  2
```

```
Maximum Matrix :
7  5  3
3  2  2
9  0  2
2  2  2
4  3  3

Available Matrix :
3  3  2

Need Matrix :
7  4  3
1  2  2
6  0  0
0  1  1
4  3  1
Safe Sequence :  P1  P3  P4  P0  P2
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 6 Banker's Algorithm]
└─$ ./BankersAlgorithm

Enter total number of processes : 5

Enter total number of resources : 1

Process 0
Allocation for resource 1 : 0
Maximum for resource 1 : 7

Process 1
Allocation for resource 1 : 2
Maximum for resource 1 : 3

Process 2
Allocation for resource 1 : 3
Maximum for resource 1 : 4

Process 3
Allocation for resource 1 : 2
Maximum for resource 1 : 13

Process 4
Allocation for resource 1 : 0
Maximum for resource 1 : 4

Available Resources :
Resource 1 : 3
```

```
Allocation Matrix :
0
2
3
2
0

Maximum Matrix :
7
3
4
13
4

Available Matrix :
3

Need Matrix :
7
1
1
11
4
Deadlock Occurs
```

Program Code :

```
#include<stdio.h>
void main(){
   int block[50], process[50], bsize, psize, iscompleted[50], allocation[50], i, j;
   printf(" Enter no. of memory blocks: ");
   scanf("%d", &bsize);
   printf("\n Enter size of each memory block: ");
   for(i = 0; i < bsize; i++){
      scanf("%d", &block[i]);
   }
   printf("\n Enter no. of processes: ");
   scanf("%d", &psize);
   printf("\n Enter size of each process: ");
   for(i = 0; i < psize; i++){
      scanf("%d", &process[i]);
   }
   for(i = 0; i < bsize; i++){
      allocation[i] = -1;
   }
   for(i = 0; i < psize; i++){
      iscompleted[i] = 0;
   }
   for(i = 0; i < psize; i++){
      for(j = 0; j < bsize; j++){
         if(allocation[j] == -1 && block[j] >= process[i]){
            allocation[j] = i;
            iscompleted[i] = 1;
            break;
         }
      }
   }
   printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
   for(i = 0; i < bsize; i++){
      printf("\n %d\t\t%dK\t\t", i+1, block[i]);
      if(allocation[i] != -1){
         printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
      }else{
         printf("Not allocated\t\t\t\t%dK",block[i]);
      }
   }
   for(i = 0; i < psize; i++){
      if(iscompleted[i] != 1){
         printf("\n process %d can't be allocated",i+1);
      }
   }

}
```

Sample Output :

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./FirstFit
Enter no. of memory blocks: 6

Enter size of each memory block: 200 400 600 500 300 250

Enter no. of processes: 4

Enter size of each process: 357 210 468 491

Block no.      size            process no.             size            unused memory

1              200K            Not allocated                           200K
2              400K            1                       357K            43K
3              600K            2                       210K            390K
4              500K            3                       468K            32K
5              300K            Not allocated                           300K
6              250K            Not allocated                           250K
process 4 can't be allocated
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./FirstFit
Enter no. of memory blocks: 5

Enter size of each memory block: 50 100 90 200 50

Enter no. of processes: 4

Enter size of each process: 90 20 50 10

Block no.      size            process no.             size            unused memory

1              50K             2                       20K             30K
2              100K            1                       90K             10K
3              90K             3                       50K             40K
4              200K            4                       10K             190K
5              50K             Not allocated                           50K
```

Program Code :

```
#include<stdio.h>
void main(){
   int block[50], process[50], bsize, psize, iscompleted[50], allocation[50],i, j,index;
   printf(" Enter no. of memory blocks: ");
   scanf("%d", &bsize);
   printf("\n Enter size of each memory block: ");
   for(i = 0; i < bsize; i++){
      scanf("%d", &block[i]);
   }
   printf("\n Enter no. of processes: ");
   scanf("%d", &psize);
   printf("\n Enter size of each process: ");
   for(i = 0; i < psize; i++){
      scanf("%d", &process[i]);
   }
   for(i = 0; i < bsize; i++){
      allocation[i] = -1;
   }
   for(i = 0; i < psize; i++){
      iscompleted[i] = 0;
   }
   for(i=0;i<psize;i++){
      index =-1;
      for(j=0;j<bsize;j++){
         if((block[j]>=process[i])&&(allocation[j]==-1)){
            if(index==-1){
               index=j;
            }else if( block[index] < block[j]){
               index = j;
            }
         }
      }
      if(index!=-1){
         allocation[index] = i;
         iscompleted[i]=1;
      }
   }
   printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
   for(i = 0; i < bsize; i++){
      printf("\n %d\t\t%dK\t\t", i+1, block[i]);
      if(allocation[i] != -1){
         printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
      }else{
         printf("Not allocated\t\t\t\t%dK",block[i]);
      }
   }
   for(i = 0; i < psize; i++){
      if(iscompleted[i] != 1){
         printf("\n process %d can't be allocated",i+1);
```

```
        }
    }
}
```

Sample Output :

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./WorstFit
Enter no. of memory blocks: 5

Enter size of each memory block: 50 100 90 200 50

Enter no. of processes: 4

Enter size of each process: 10 20 30 70

Block no.      size            process no.            size            unused memory

1              50K             Not allocated                          50K
2              100K            2                      20K             80K
3              90K             3                      30K             60K
4              200K            1                      10K             190K
5              50K             Not allocated                          50K
process 4 can't be allocated
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./WorstFit
Enter no. of memory blocks: 6

Enter size of each memory block: 200 400 600 500 300 250

Enter no. of processes: 4

Enter size of each process: 357 210 468 491

Block no.      size            process no.            size            unused memory

1              200K            Not allocated                          200K
2              400K            Not allocated                          400K
3              600K            1                      357K            243K
4              500K            2                      210K            290K
5              300K            Not allocated                          300K
6              250K            Not allocated                          250K
process 3 can't be allocated
process 4 can't be allocated
```

Program Code :

```
#include<stdio.h>
void main(){
   int block[50], process[50], bsize, psize, iscompleted[50], allocation[50],temp,i, j,index=-1;
   printf(" Enter no. of memory blocks: ");
   scanf("%d", &bsize);
   printf("\n Enter size of each memory block: ");
   for(i = 0; i < bsize; i++){
      scanf("%d", &block[i]);
   }
   printf("\n Enter no. of processes: ");
   scanf("%d", &psize);
   printf("\n Enter size of each process: ");
   for(i = 0; i < psize; i++){
      scanf("%d", &process[i]);
   }
   for(i = 0; i < bsize; i++){
      allocation[i] = -1;
   }
   for(i = 0; i < psize; i++){
      iscompleted[i] = 0;
   }
   for(i=0;i<psize;i++){
      index =-1;
      for(j=0;j<bsize;j++){
         if((block[j]>=process[i])&&(allocation[j]==-1)){
            if(index==-1){
               index=j;
            }else if( block[index] > block[j]){
               index = j;
            }
         }
      }
      if(index!=-1){
         allocation[index] = i;
         iscompleted[i]=1;
      }
   }
   printf("\n Block no.\tsize\t\tprocess no.\t\tsize\t\tunused memory\n");
   for(i = 0; i < bsize; i++){
      printf("\n %d\t\t%dK\t\t", i+1, block[i]);
      if(allocation[i] != -1){
         printf("%d\t\t\t%dK\t\t%dK",allocation[i]+1,process[allocation[i]],block[i]-
process[allocation[i]]);
      }else{
         printf("Not allocated\t\t\t\t%dK",block[i]);
      }
   }
   for(i = 0; i < psize; i++){
      if(iscompleted[i] != 1){
         printf("\n process %d can't be allocated",i+1);
```

```
        }
    }
}
```

Sample Output :

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./BestFit
Enter no. of memory blocks: 6

Enter size of each memory block: 200 400 600 500 300 250

Enter no. of processes: 4

Enter size of each process: 357 210 468 491

Block no.      size           process no.            size           unused memory

1              200K           Not allocated                         200K
2              400K           1                      357K           43K
3              600K           4                      491K           109K
4              500K           3                      468K           32K
5              300K           Not allocated                         300K
6              250K           2                      210K           40K
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 7 Memory Allocation Methods]
└─$ ./BestFit
Enter no. of memory blocks: 5

Enter size of each memory block: 50 100 90 200 50

Enter no. of processes: 4

Enter size of each process: 90 20 50 200

Block no.      size           process no.            size           unused memory

1              50K            2                      20K            30K
2              100K           Not allocated                         100K
3              90K            1                      90K            0K
4              200K           4                      200K           0K
5              50K            3                      50K            0K
```

Program Code :

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
void fibonacci(int n){
    int a =0 ,b = 1,c,i;
    printf(" %d  %d ",a,b);
    c=a+b;
    while(a+b<=n) {
        c = a + b;
        printf(" %d ",c);
        a = b;
        b = c;
   }
}

void prime(int n){
    int i,j,flag;
    for (i=2; i<=n; i++){
        flag=1;
        for (j=2; j<=i/2; j++){
            if(i%j == 0){
                flag=0;
                break;
            }
        }
        if(flag==1){
            printf(" %d ",i);
        }
    }
}

int main(){
    int n;
    printf("\n\nEnter the value of n :");
                scanf("%d",&n);
                if(fork() ==0 ){
                    printf("\n\nChild Process : Fibonacci Series \n");
                    fibonacci(n);
                }else{
                    wait(NULL);
                    printf("\n\nParent Process : Prime Numbers \n");
                    prime(n);
                }
                printf("\n");

}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./a
Enter the value of n :10
Child Process : Fibonacci Series
 0  1  1  2  3  5  8

Parent Process : Prime Numbers
 2  3  5  7

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./a
Enter the value of n :20
Child Process : Fibonacci Series
 0  1  1  2  3  5  8  13

Parent Process : Prime Numbers
 2  3  5  7  11  13  17  19

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./a
Enter the value of n :50
Child Process : Fibonacci Series
 0  1  1  2  3  5  8  13  21  34

Parent Process : Prime Numbers
 2  3  5  7  11  13  17  19  23  29  31  37  41  43  47

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ |
```

Program Code :

```
#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>


void main(){
    int n,i;
    printf("Enter the value of N : ");
    scanf("%d",&n);
    for(i =0; i<n ; i++){
        if(fork() == 0){
            printf("Level %d:  Child pid: %d   Parent pid: %d \n",i,getpid(),getppid());
        }else{
            wait(NULL);
            break;
        }
    }
}
```

Sample Output :

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./b
Enter the value of N : 10
Level 0:  Child pid: 2025   Parent pid: 2020
Level 1:  Child pid: 2026   Parent pid: 2025
Level 2:  Child pid: 2027   Parent pid: 2026
Level 3:  Child pid: 2028   Parent pid: 2027
Level 4:  Child pid: 2029   Parent pid: 2028
Level 5:  Child pid: 2030   Parent pid: 2029
Level 6:  Child pid: 2031   Parent pid: 2030
Level 7:  Child pid: 2032   Parent pid: 2031
Level 8:  Child pid: 2033   Parent pid: 2032
Level 9:  Child pid: 2034   Parent pid: 2033

┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./b
Enter the value of N : 5
Level 0:  Child pid: 2036   Parent pid: 2035
Level 1:  Child pid: 2037   Parent pid: 2036
Level 2:  Child pid: 2038   Parent pid: 2037
Level 3:  Child pid: 2039   Parent pid: 2038
Level 4:  Child pid: 2040   Parent pid: 2039
```

Program Code :

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
void main() {
  printf("Parent A: %d\n", getpid());
  if (fork() == 0) {
     printf(" Child B: %d forked by Parent A: %d\n", getpid(), getppid());
     if (fork() == 0) {
        printf(" Child D: %d forked by Parent B: %d\n", getpid(), getppid());
        if (fork() == 0) {
           printf(" Child H: %d forked by Parent D: %d\n", getpid(), getppid());
           if (fork() == 0) {
              printf(" Child I: %d forked by Parent H: %d\n", getpid(), getppid());
           }else{
            wait(NULL);
           }
        }else{
           wait(NULL);
        }
     }else if (fork() == 0) {
        printf(" Child E: %d forked by Parent B: %d\n", getpid(), getppid());
     }else if (fork() == 0) {
        printf(" Child F: %d forked by Parent B: %d\n", getpid(), getppid());
     }else{
        wait(NULL);
     }
  }else if (fork() == 0) {
     printf(" Child C: %d forked by Parent A: %d\n", getpid(), getppid());
     if (fork() == 0) {
       printf(" Child G: %d forked by Parent C: %d\n", getpid(), getppid());
     }else{
       wait(NULL);
     }
  }else{
     wait(NULL);
  }
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 9 System Calls]
└─$ ./c
Parent A: 2099
 Child B: 2100 forked by Parent A: 2099
 Child C: 2101 forked by Parent A: 2099
 Child D: 2102 forked by Parent B: 2100
 Child G: 2103 forked by Parent C: 2101
 Child E: 2104 forked by Parent B: 2100
 Child F: 2105 forked by Parent B: 2100
 Child H: 2106 forked by Parent D: 2102
 Child I: 2107 forked by Parent H: 2106
```

Program Code :

```
#include<stdio.h>
int isHit(int data,int frame[],int fsize){
    int hit=0;
    for(int j=0; j<fsize; j++){
        if(frame[j]==data){
            hit=1;
            break;
        }
    }
    return hit;
}

void main(){
    int pgsize,fsize,pstring[50],i,nfault=0,k;
    printf("\nEnter length of page reference string:");
    scanf("%d",&pgsize);
    printf("\nEnter the page reference string:");
    for(i=0; i<pgsize; i++){
        scanf("%d",&pstring[i]);
    }
    printf("\nEnter no of frames:");
    scanf("%d",&fsize);
    int frame[fsize];
    for(i=0; i<fsize; i++){
        frame[i]=9999;
    }
    for(i=0; i<pgsize; i++){
        printf("\nFor %d :",pstring[i]);
        if(isHit(pstring[i],frame,fsize)==0){
            for(k=0; k<fsize-1; k++){
                frame[k]=frame[k+1];
            }
            frame[k]=pstring[i];
            nfault++;
            for (k=0; k<fsize; k++){
                if(frame[k]!=9999){
                    printf(" %d",frame[k]);
                }
            }
        }
        else{
            printf("No page fault");
        }
    }
    printf("\nTotal no of page faults: %d",nfault);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./FIFO

Enter length of page reference string:10

Enter the page reference string:1 5 4 2 4 5 6 1 6 8

Enter no of frames:4

For 1 : 1
For 5 : 1 5
For 4 : 1 5 4
For 2 : 1 5 4 2
For 4 : No page fault
For 5 : No page fault
For 6 : 5 4 2 6
For 1 : 4 2 6 1
For 6 : No page fault
For 8 : 2 6 1 8

Total no of page faults: 7
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./FIFO

Enter length of page reference string:12

Enter the page reference string:3 2 1 0 3 2 4 3 2 1 0 4

Enter no of frames:3

For 3 : 3
For 2 : 3 2
For 1 : 3 2 1
For 0 : 2 1 0
For 3 : 1 0 3
For 2 : 0 3 2
For 4 : 3 2 4
For 3 : No page fault
For 2 : No page fault
For 1 : 2 4 1
For 0 : 4 1 0
For 4 : No page fault

Total no of page faults: 9
```

Program Code :

```
#include<stdio.h>
int isHit(int data,int frame[],int fsize){
    int hit=0;
    for(int j=0; j<fsize; j++){
        if(frame[j]==data){
            hit=1;
            break;
        }
    }
    return hit;
}

void main(){
    int pgsize,fsize,pstring[50],i,j,nfault=0,k,past_index[50],page,found,min,index;
    printf("\nEnter length of page reference string:");
    scanf("%d",&pgsize);
    printf("\nEnter the page reference string:");
    for(i=0; i<pgsize; i++){
        scanf("%d",&pstring[i]);
    }
    printf("\nEnter no of frames:");
    scanf("%d",&fsize);
    int frame[fsize];
    for(i=0; i<fsize; i++){
        frame[i]=9999;
    }
    for(i=0; i<pgsize; i++){
        printf("\nFor %d :",pstring[i]);
        if(isHit(pstring[i],frame,fsize)==0){
            for(j=0; j<fsize;j++){
                page = frame[j];
                found=0;
                for(k=i-1;k>=0;k--){
                    if(page==pstring[k]){
                        past_index[j]=k;
                        found=1;
                        break;
                    }
                    else{
                        found=0;
                    }
                }
                if(found==0){
                    past_index[j]=-9999;
                }
            }
            min=9999;
           /*printf("\n");
            for(int l=0;l<fsize;l++){
                printf(" %d ",past_index[l]);
```

```
            }
            printf("\n");*/
            for(j=0; j<fsize; j++)
            {
                if(past_index[j]<min)
                {
                    min=past_index[j];
                    index=j;
                }
            }
            frame[index]=pstring[i];
            nfault++;
            for (k=0; k<fsize; k++){
                if(frame[k]!=9999){
                    printf(" %d",frame[k]);
                }
            }
        }
        else{
            printf("No page fault");
        }
    }
    printf("\nTotal no of page faults: %d",nfault);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LRU

Enter length of page reference string:13

Enter the page reference string:7 0 1 2 0 3 0 4 2 3 0 3 2

Enter no of frames:4

For 7 : 7
For 0 : 7 0
For 1 : 7 0 1
For 2 : 7 0 1 2
For 0 : No page fault
For 3 : 3 0 1 2
For 0 : No page fault
For 4 : 3 0 4 2
For 2 : No page fault
For 3 : No page fault
For 0 : No page fault
For 3 : No page fault
For 2 : No page fault

Total no of page faults: 6
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LRU

Enter length of page reference string:15

Enter the page reference string:7 2 3 1 2 5 3 4 6 7 7 1 0 5 4

Enter no of frames:3

For 7 : 7
For 2 : 7 2
For 3 : 7 2 3
For 1 : 1 2 3
For 2 : No page fault
For 5 : 1 2 5
For 3 : 3 2 5
For 4 : 3 4 5
For 6 : 3 4 6
For 7 : 7 4 6
For 7 : No page fault
For 1 : 7 1 6
For 0 : 7 1 0
For 5 : 5 1 0
For 4 : 5 4 0

Total no of page faults: 13
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LRU

Enter length of page reference string:12

Enter the page reference string:3 2 1 0 3 2 4 3 2 1 0 4

Enter no of frames:3

For 3 : 3
For 2 : 3 2
For 1 : 3 2 1
For 0 : 0 2 1
For 3 : 0 3 1
For 2 : 0 3 2
For 4 : 4 3 2
For 3 : No page fault
For 2 : No page fault
For 1 : 1 3 2
For 0 : 1 0 2
For 4 : 1 0 4

Total no of page faults: 10
```

Program Code :

```
#include<stdio.h>
int isHit(int data,int frame[],int fsize){
    int hit=0;
    for(int j=0; j<fsize; j++){
        if(frame[j]==data){
            hit=1;
            break;
        }
    }
    return hit;
}
void main(){
    int pgsize,fsize,pstring[50],i,j,nfault=0,k,count[50],min,index,val;
    printf("\nEnter length of page reference string:");
    scanf("%d",&pgsize);
    printf("\nEnter the page reference string:");
    for(i=0; i<pgsize; i++){
        scanf("%d",&pstring[i]);
    }
    printf("\nEnter no of frames:");
    scanf("%d",&fsize);
    int frame[fsize];
    for(i=0; i<fsize; i++){
        frame[i]=9999;
        count[i]=0;
    }
    j=0;
    for(i=0; i<pgsize; i++){
        printf("\nFor %d :",pstring[i]);
        if(isHit(pstring[i],frame,fsize)==0){
            if(j<fsize){
                frame[j]=pstring[i];
                count[j]=count[j]+1;
                j++;
            }else{
                min=9999;
                for(k=0; k<fsize; k++){
                    if(count[k]<min){
                        min=count[k];
                        index=k;
                    }
                }
                for(k=index;k<fsize-1;k++){
                    frame[k]=frame[k+1];
                    count[k]=count[k+1];
                }
                frame[fsize-1]=pstring[i];
                val=0;
                for(k=0; k<=i; k++){
                    if(pstring[i]==pstring[k]){
```

```
                val=val+1;
            }
        }
        count[fsize-1]=val;
    }
    for (k=0; k<fsize; k++){
        if(frame[k]!=9999){
            printf(" %d",frame[k]);
        }
    }
    nfault++;
}
else{
    for(k=0; k<fsize; k++){
        if(frame[k]==pstring[i]){
            index=k;
            break;
        }
    }
    count[k]=count[k]+1;
    printf("No page fault");
}
printf("\n");
for(int l=0;l<fsize;l++){
    printf(" %d ",count[l]);
}
}
printf("\nTotal no of page faults: %d",nfault);
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LFU

Enter length of page reference string:13

Enter the page reference string:7 0 1 2 0 3 0 4 2 3 0 3 2

Enter no of frames:4

For 7 : 7
For 0 : 7 0
For 1 : 7 0 1
For 2 : 7 0 1 2
For 0 : No page fault
For 3 : 0 1 2 3
For 0 : No page fault
For 4 : 0 2 3 4
For 2 : No page fault
For 3 : No page fault
For 0 : No page fault
For 3 : No page fault
For 2 : No page fault
Total no of page faults: 6
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LFU

Enter length of page reference string:15

Enter the page reference string:7 2 3 1 2 5 3 4 6 7 7 1 0 5 4

Enter no of frames:3

For 7 : 7
For 2 : 7 2
For 3 : 7 2 3
For 1 : 2 3 1
For 2 : No page fault
For 5 : 2 1 5
For 3 : 2 5 3
For 4 : 2 3 4
For 6 : 2 4 6
For 7 : 2 6 7
For 7 : No page fault
For 1 : 2 7 1
For 0 : 2 7 0
For 5 : 2 7 5
For 4 : 2 7 4
Total no of page faults: 13
```

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 8 Page Replacement Algorithm]
└─$ ./LFU

Enter length of page reference string:10

Enter the page reference string:2 3 4 2 1 3 7 5 4 3

Enter no of frames:3

For 2 : 2
For 3 : 2 3
For 4 : 2 3 4
For 2 : No page fault
For 1 : 2 4 1
For 3 : 2 1 3
For 7 : 2 3 7
For 5 : 2 7 5
For 4 : 2 5 4
For 3 : 2 4 3
Total no of page faults: 9
```

Program Code :

```
#include<stdio.h>
int mutex=1,full,empty,buffer[20];

int wait(int val){
   return (--val);
}
int signal(int val){
   return (++val);
}
void producer(int item){
   empty=wait(empty);
   mutex=wait(mutex);
   buffer[full]=item;
   printf("  Producer produces the item %d",item);
   mutex=signal(mutex);
   full=signal(full);
}
void consumer(){
   full=wait(full);
   mutex=wait(mutex);
   int item = buffer[full];
   printf("  Consumer consumes item %d",item);
   mutex=signal(mutex);
   empty=signal(empty);
}
void main(){
   int n,size,var=1,item;
   printf("Enter the size of Buffer :");
   scanf("%d",&size);
   full=0;
   empty=size;
   printf("\nProducer Consumer Problem");
   printf("\n  1.Produce");
   printf("\n  2.Consume");
   printf("\n  3.Exit");
   while(var==1){
      printf("\nEnter your choice:");
      scanf("%d",&n);
      switch(n){
         case 1:
            if((mutex==1)&&(empty!=0)){
               printf("  Enter item to add in Buffer : ");
               scanf("%d",&item);
               producer(item);
            }else{
               printf("  Buffer is full!!");
            }break;
         case 2:
            if((mutex==1)&&(full!=0)){
               consumer();
```

```
            }else{
                printf("  Buffer is empty!!");
            }break;
        case 3:
            var=0;
            break;
    }
  }
}
```

Sample Output:

```
┌──(rinoy2002㉿kali-virtual)-[~/Desktop/OS Lab/Day 11 Producer Consumer Problem]
└─$ ./ProducerConsumer
Enter the size of Buffer :5

Producer Consumer Problem
  1.Produce
  2.Consume
  3.Exit
Enter your choice:1
  Enter item to add in Buffer : 4
  Producer produces the item 4
Enter your choice:1
  Enter item to add in Buffer : 9
  Producer produces the item 9
Enter your choice:1
  Enter item to add in Buffer : 0
  Producer produces the item 0
Enter your choice:2
  Consumer consumes item 0
Enter your choice:1
  Enter item to add in Buffer : 3
  Producer produces the item 3
Enter your choice:2
  Consumer consumes item 3
Enter your choice:2
  Consumer consumes item 9
Enter your choice:1
  Enter item to add in Buffer : 2
  Producer produces the item 2
Enter your choice:1
  Enter item to add in Buffer : 90
```

```
  Producer produces the item 90
Enter your choice:1
  Enter item to add in Buffer : 67
  Producer produces the item 67
Enter your choice:1
  Enter item to add in Buffer : 45
  Producer produces the item 45
Enter your choice:1
  Buffer is full!!
Enter your choice:2
  Consumer consumes item 45
Enter your choice:2
  Consumer consumes item 67
Enter your choice:2
  Consumer consumes item 90
Enter your choice:2
  Consumer consumes item 2
Enter your choice:2
  Consumer consumes item 4
Enter your choice:2
  Buffer is empty!!
Enter your choice:3
```