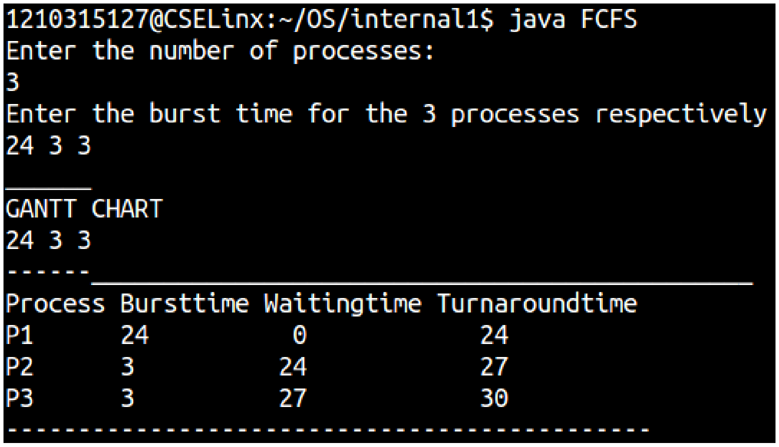
1. FCFS without preemption

```
import java.util.*;
class FCFS
{
        public static void main(String args[])
        {
                int i,n;
                System.out.println("Enter the number of processes: ");
                Scanner s = new Scanner(System.in);
                n = s.nextInt();
                int b[];
                b = new int[n];
                System.out.println("Enter the burst time for the "+n+" processes
respectively");
                for(i=0;i<n;i++)
                {
                        b[i]=s.nextInt();
                }
                for(i=0;i<n;i++)
                        System.out.print("__");
                System.out.println("");
                System.out.println("GANTT CHART");
                for(i=0;i<n;i++)
                {
                        System.out.print(b[i]+" ");
                }
                System.out.print("");
                System.out.println("");
                for(i=0;i<n;i++)
                        System.out.print("--");
                int waittime=0,turnaroundtime;

        System.out.println("_______________________________________________
_");
```

```
        System.out.println("Process Bursttime Waitingtime
Turnaroundtime");
        for(i=0;i<n;i++)
        {
            if(i==0)
            {
                waittime=waittime;
                turnaroundtime=b[i]+waittime;
            }
            else
            {
                waittime=waittime+b[i-1];
                turnaroundtime=b[i]+waittime;
            }
        System.out.println("P"+(i+1)+"      "+b[i]+"         "+waittime+"
"+turnaroundtime+"            ");
        }
        System.out.println("--------------------------------------------");
    }
}
```

Output:

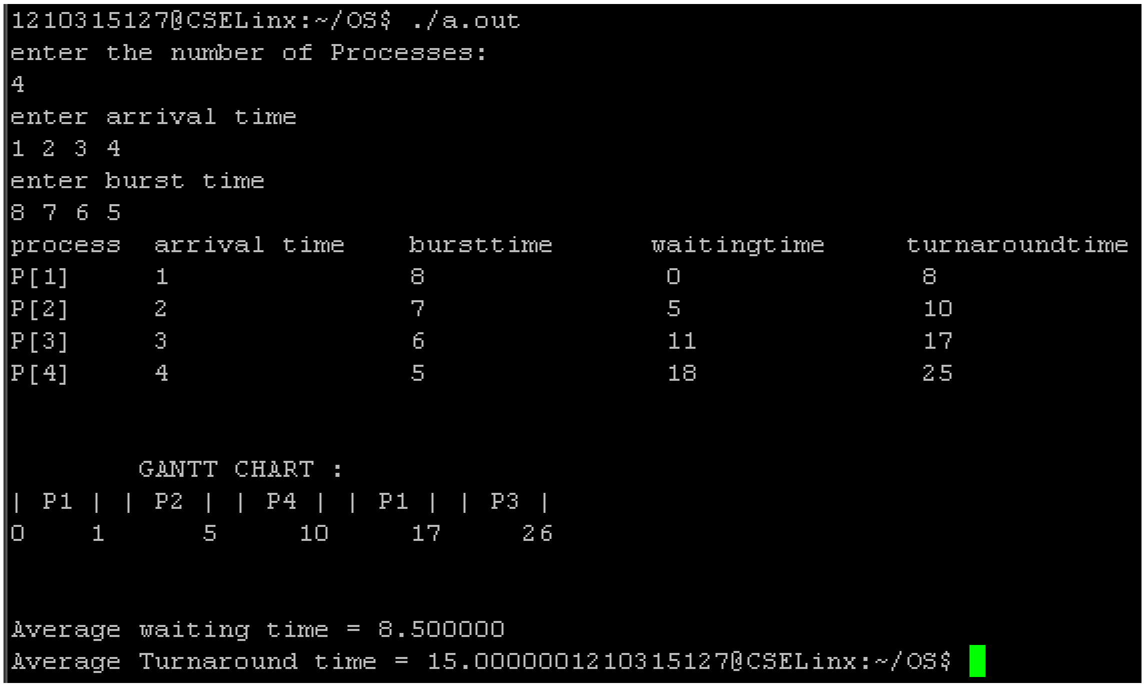


```
1210315127@CSELinx:~/OS/internal1$ java FCFS
Enter the number of processes:
3
Enter the burst time for the 3 processes respectively
24 3 3

______
GANTT CHART
24 3 3
------______________________________________________
Process Bursttime Waitingtime Turnaroundtime
P1      24          0            24
P2      3          24            27
P3      3          27            30
---------------------------------------------
```

2. SJF with Preemption

```
#include <stdio.h>
int main()
{
 int k=0,y[10],z[10],a[10],b[10],x[10],i,j,smallest,count=0,time,n;
 double avg=0,tt=0,end;
  printf("enter the number of Processes:\n");
  scanf("%d",&n);
 printf("enter arrival time\n");
 for(i=0;i<n;i++)
 scanf("%d",&a[i]);
 printf("enter burst time\n");
 for(i=0;i<n;i++)
 scanf("%d",&b[i]);
 for(i=0;i<n;i++)
 x[i]=b[i];
  b[9]=9999;
 for(time=0;count!=n;time++)
 { smallest=9;
  for(i=0;i<n;i++)
{  if(a[i]<=time && b[i]<b[smallest] && b[i]>0 )
   smallest=i;}
  b[smallest]--;
  if(b[smallest]==0)
  {count++;
   end=time+1;
   z[k]=end-a[smallest]-x[smallest];
   y[k]=+end-a[smallest];
   avg=avg+end-a[smallest]-x[smallest];
   tt= tt+end-a[smallest];
    k++;} }
  printf("process\t arrival time \t bursttime\twaitingtime\tturnaroundtime\n");
  for(i=0;i<n;i++)
  {printf("P[%d]\t %d \t\t %d \t\t %d \t\t %d \n",i+1,a[i],x[i],z[i],y[i]);}
  printf("\n\n\tGANTT CHART :\n");
 printf("| P1 | | P2 | | P4 | | P1 | | P3 | ");
  printf("\n");
  printf("0    1      5     10     17     26 ");
  printf("\n");
  printf("\n\nAverage waiting time = %lf\n",avg/n);
    printf("Average Turnaround time = %lf",tt/n);
   return 0;}
```

Output:



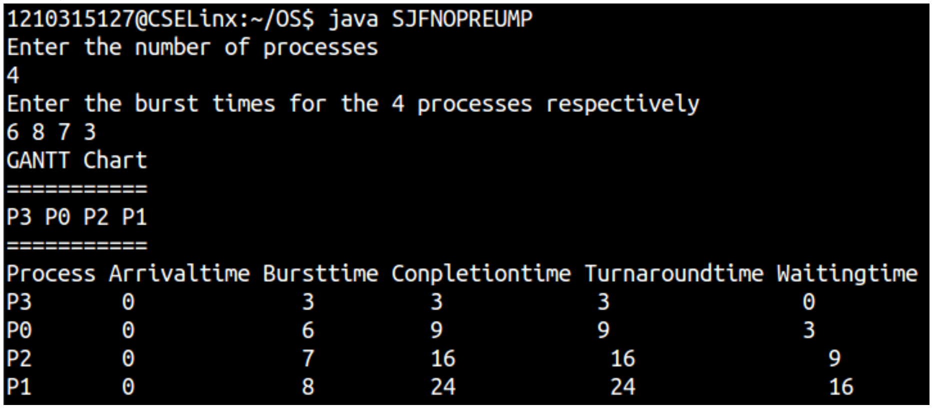
```
1210315127@CSELinx:~/OS$ ./a.out
enter the number of Processes:
4
enter arrival time
1 2 3 4
enter burst time
8 7 6 5
process  arrival time    bursttime       waitingtime      turnaroundtime
P[1]     1               8                0                8
P[2]     2               7                5                10
P[3]     3               6                11               17
P[4]     4               5                18               25


        GANTT CHART :
| P1 | | P2 | | P4 | | P1 | | P3 |
0    1      5     10     17     26


Average waiting time = 8.500000
Average Turnaround time = 15.0000001210315127@CSELinx:~/OS$ 
```

3. SJF without preumption

SJFNOPREUMP.java

```
import java.util.*;
class SJFNOPREUMP
{
 public static void main(String args[])
 {
  System.out.println("Enter the number of processes");
  Scanner s = new Scanner(System.in);
  int n = s.nextInt();
  System.out.println("Enter the burst times for the "+n+" processes respectively");
  int b[] = new int [n];
  int p[] = new int [n];
  int btime[] = new int [n];
  int i,j;
  for(i=0;i<n;i++)
     b[i] = s.nextInt();
  for(i=0;i<n;i++)
     btime[i] = b[i];
  Arrays.sort(b);
  /*The processes in sorted order*/
  int []porder;
  porder = new int [n];
  for(i=0;i<n;i++)
   for(j=0;j<n;j++)
     if(b[i] == btime[j])
        porder[i] = j;
  System.out.println("GANTT Chart");
  System.out.println("===========");
  for(i=0;i<n;i++)
    System.out.print("P"+porder[i]+" ");
  System.out.println("");
  System.out.println("===========");
  int []ct;
  int []tat;
  ct = new int [n];
  for(i=0;i<n;i++)
   {
    if(i==0)
     ct[i] = b[i];
    else
     ct[i] = ct[i-1] + b[i];
   }
  tat = ct;
  int wt[] = new int [n];
  for(i=0;i<n;i++)
   wt[i] = tat[i] - b[i];
  System.out.println("Process Arrivaltime Bursttime Conpletiontime
Turnaroundtime Waitingtime");
  for(i=0;i<n;i++)
   System.out.println("P"+porder[i]+"       "+"0"+"            "+b[i]+"         "+ct[i]+"
"+tat[i]+"               "+wt[i]);
 }
}
```

Output:



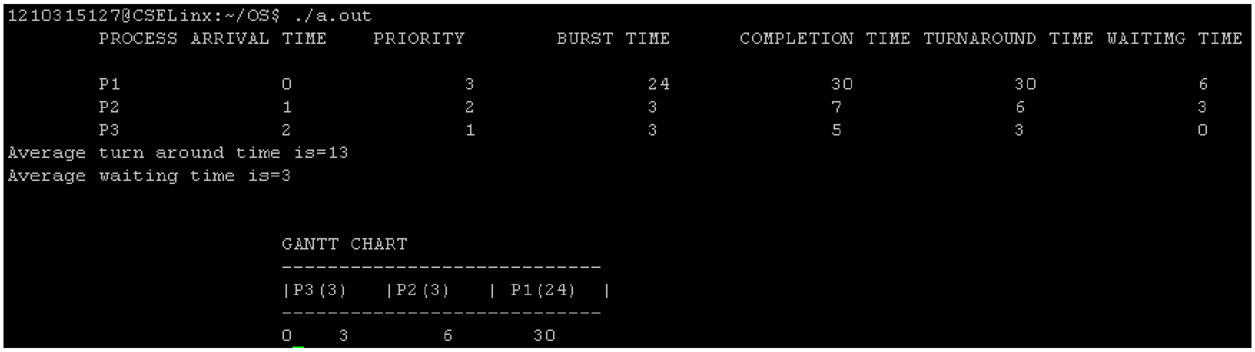
```
1210315127@CSELinx:~/OS$ java SJFNOPREUMP
Enter the number of processes
4
Enter the burst times for the 4 processes respectively
6 8 7 3
GANTT Chart
===========
P3 P0 P2 P1
===========
Process Arrivaltime Bursttime Conpletiontime Turnaroundtime Waitingtime
P3       0             3         3            3               0
P0       0             6         9            9               3
P2       0             7         16            16               9
P1       0             8         24            24               16
```

4. Priority with preemption

```
#include<iostream>
#include<iomanip>
using namespace std;
int main()
{
int a[3]={0,1,2};
int b[3]={24,3,3};
int priority[3]={3,2,1};
int c[3]={30,7,5};
int t[7],w[7],temp,i,j;
int p[7]={1,2,3};
    for(i=0;i<3;i++)
    {
        t[i]=c[i]-a[i];
        w[i]=t[i]-b[i];
    }
    for(i=0;i<2;i++)
    {
        for(j=i+1;j<3;j++)
        {
            if(priority[i]>priority[j])
            {
                temp=b[i];
                b[i]=b[j];
                b[j]=temp;
                temp=c[i];
                c[i]=c[j];
                c[j]=temp;
                temp=t[i];
                t[i]=t[j];
                t[j]=temp;
                temp=w[i];
                w[i]=w[j];
                w[j]=temp;
```

```
                    temp=p[i];
                    p[i]=p[j];
                    p[j]=temp;
               }
          }
     }
 int k;
   cout<<"\tPROCESS\tARRIVAL TIME\tPRIORITY\tBURST TIME\tCOMPLETION
TIME\tTURNAROUND TIME\tWAITIMG TIME\n\n";
          for(i=2;i>=0;i--)
          {

cout<<"\tP"<<p[i]<<"\t\t"<<k<<"\t\t"<<priority[k]<<"\t\t"<<b[i]<<"\t\t"<<c[i]<<"\t
\t"<<t[i]<<"\t\t"<<w[i];
               k++;
               cout<<"\n";
          }
       int sum,sum1;
          sum=0;sum1=0;
          sum=t[0]+t[1]+t[2];
      float avg=sum/3;
          sum1=w[0]+w[1]+w[2];
      float avg1=sum1/3;
     cout<<"Average turn around time is="<<avg<<endl;
     cout<<"Average waiting time is="<<avg1<<endl;
cout<<"\n\n\t\t\tGANTT CHART"<<endl;
cout<<"\t\t\t----------------------------"<<endl;
cout<<"\t\t\t|P3(3)   |P2(3)   | P1(24)  |"<<endl;
cout<<"\t\t\t----------------------------"<<endl;
cout<<"\t\t\t0    3       6      30"<<endl;
return 0;
}
```

Output:



```
1210315127@CSELinx:~/OS$ ./a.out
        PROCESS ARRIVAL TIME    PRIORITY        BURST TIME      COMPLETION TIME TURNAROUND TIME WAITIMG TIME

        P1              0               3               24              30              30              6
        P2              1               2               3               7               6               3
        P3              2               1               3               5               3               0
Average turn around time is=13
Average waiting time is=3


                        GANTT CHART
                        ----------------------------
                        |P3(3)   |P2(3)    | P1(24)  |
                        ----------------------------
                        0    3        6       30
```

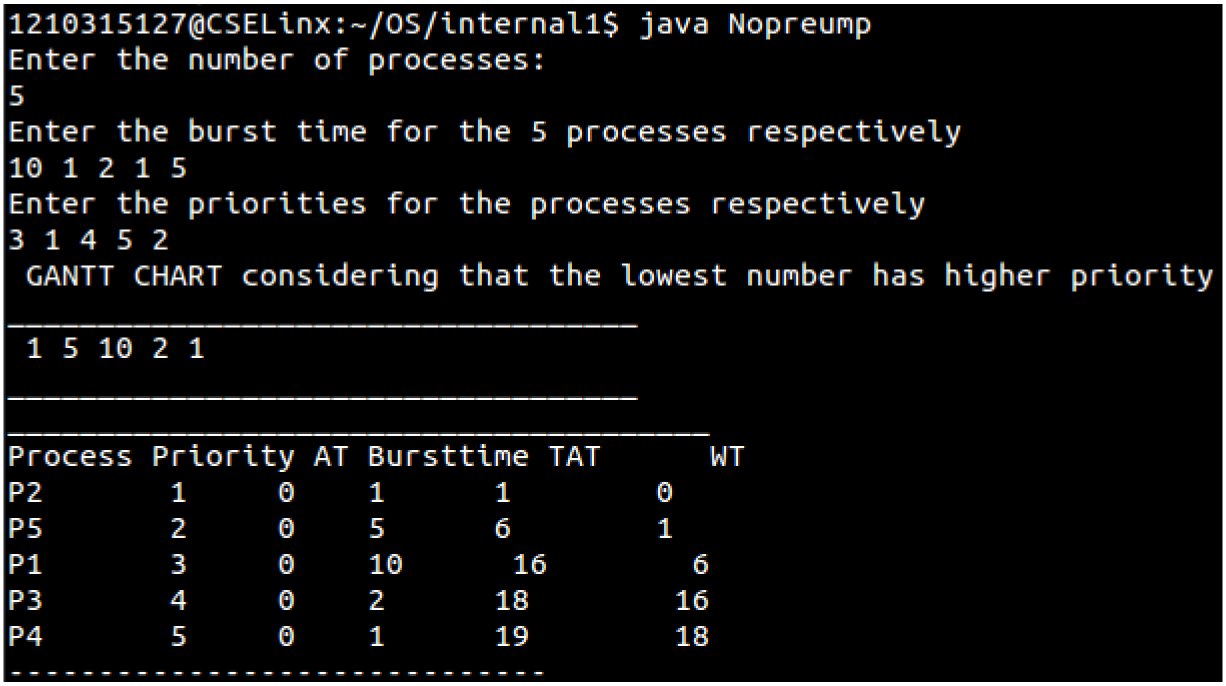
5. Priority without preumption

Nopreump.java

```
import java.util.*;
class Nopreump
{
   public static void main(String args[])
   {
      int i,j,n,temp;
      System.out.println("Enter the number of processes:");
      Scanner s = new Scanner(System.in);
      n = s.nextInt();
      int []b;
      int []p;
      b = new int[n];
      p = new int[n];
      System.out.println("Enter the burst time for the "+n+" processes respectively");
      for(i=0;i<n;i++)
      {
            b[i] = s.nextInt();
      }
      System.out.println("Enter the priorities for the processes respectively");
      for(i=0;i<n;i++)
      {
            p[i] = s.nextInt();
      }
      System.out.println(" GANTT CHART considering that the lowest number has higher priority");
      System.out.println("________________________________________");
      int []l;
      l = new int[n];
      int []m;
      m = new int[n];
      for(i=0;i<n;i++)
            l[i] = i+1;
```

```
for(i=0;i<n;i++)
{
  for( j=0;j<n;j++)
      if(l[i]==p[j])
            {
                  System.out.print(" "+b[j]);
                  m[i] = b[j];
            }
}
System.out.println("");
System.out.println("____________________________________________");
int at = 0;
int []ct;
int []tat;
tat = new int[n];
int []wt;
wt = new int[n];

ct = new int[n];

int []process;
process = new int[n];
for(i=0;i<n;i++)
{
  for( j=0;j<n;j++)
      if(l[i]==p[j])
            {
            process[i] = l[j];
            }
}



System.out.println("__________________________________________________");
    System.out.println("Process Priority AT Bursttime TAT      WT");
    for(i=0;i<n;i++)
    {
            if(i==0)
             tat[i] = m[i];
             else
             {
                   tat[i]= tat[i-1]+ m[i];
             }
            wt[i] = tat[i] -m[i];
            System.out.println("P"+process[i]+"      "+(i+1)+"     "+at+"    "+m[i]+"
"+tat[i]+"       "+wt[i]);
          }
             System.out.println("-----------------------------");

   }
}
```

Output:

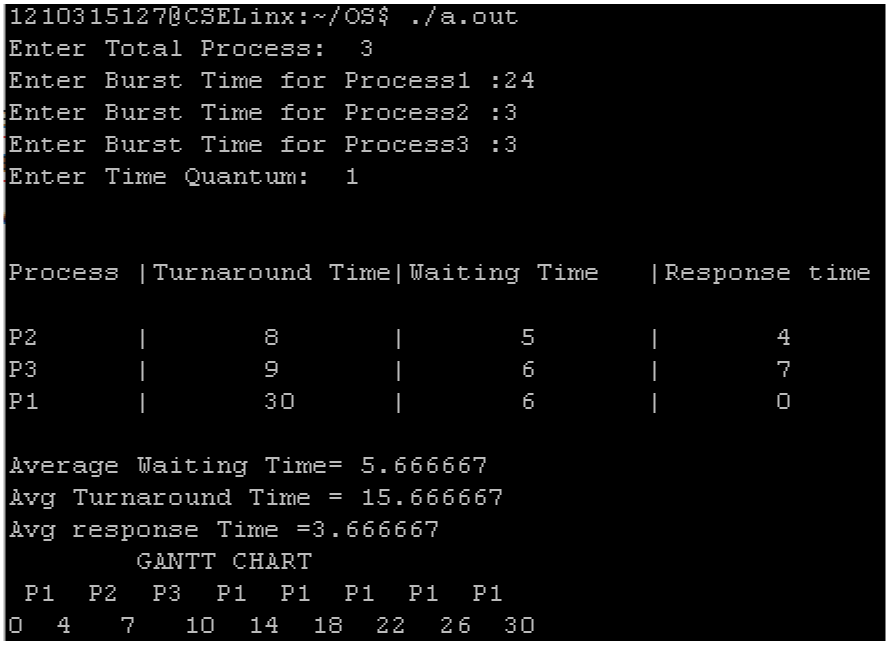


```
1210315127@CSELinx:~/OS/internal1$ java Nopreump
Enter the number of processes:
5
Enter the burst time for the 5 processes respectively
10 1 2 1 5
Enter the priorities for the processes respectively
3 1 4 5 2
 GANTT CHART considering that the lowest number has higher priority
___________________________________
 1 5 10 2 1

___________________________________
_______________________________________
Process Priority AT Bursttime TAT      WT
P2       1     0    1      1        0
P5       2     0    5      6        1
P1       3     0    10      16        6
P3       4     0    2      18        16
P4       5     0    1      19        18
------------------------------
```

6. Round robin

```
#include<stdio.h>
int main()
{
  int count,j,n,time,remain,flag=0,time_quantum;
  int wait_time=0,turnaround_time=0,bt[10],rt[10];
  int r[10]={0,4,7};  int at[10]={0,0,0};
  printf("Enter Total Process:  ");
  scanf("%d",&n); remain=n;
  for(count=0;count<n;count++)
  {
    printf("Enter Burst Time for Process%d :",count+1);
    scanf("%d",&bt[count]);
    rt[count]=bt[count];
  }
  printf("Enter Time Quantum:  ");
scanf("%d",&time_quantum);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\t|Response time\n\n");
  for(time=0,count=0;remain!=0;)
  {
    if(rt[count]<=time_quantum && rt[count]>0)
    {
      time+=rt[count];   rt[count]=0;   flag=1;
    }
    else if(rt[count]>0)
    {
      rt[count]-=time_quantum;time+=time_quantum;
    }
    if(rt[count]==0 && flag==1)
    {
      remain--;
      printf("P%d\t|\t%d\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count],r[count]);
      wait_time+=time-at[count]-bt[count];
      turnaround_time+=time-at[count];   flag=0;
    }
    if(count==n-1)    count=0;
    else if(at[count+1]<=time)   count++;
    else   count=0;
  }
  float sum=0.00;  sum=r[0]+r[1]+r[2];
  printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
 printf("Avg Turnaround Time = %f\n",turnaround_time*1.0/n);
printf("Avg response Time =%f\n",sum*1.0/n);
 printf("\tGANTT CHART\n");
printf(" P1  P2  P3  P1  P1  P1  P1  P1\n");
 printf("0  4   7   10  14  18  22  26  30\n");
  return 0;
}
```

Output:



```
1210315127@CSELinx:~/OS$ ./a.out
Enter Total Process:  3
Enter Burst Time for Process1 :24
Enter Burst Time for Process2 :3
Enter Burst Time for Process3 :3
Enter Time Quantum:  1


Process |Turnaround Time|Waiting Time   |Response time

P2      |       8       |       5       |       4
P3      |       9       |       6       |       7
P1      |       30      |       6       |       0

Average Waiting Time= 5.666667
Avg Turnaround Time = 15.666667
Avg response Time =3.666667
        GANTT CHART
 P1  P2  P3  P1  P1  P1  P1  P1
0  4   7   10  14  18  22  26  30
```

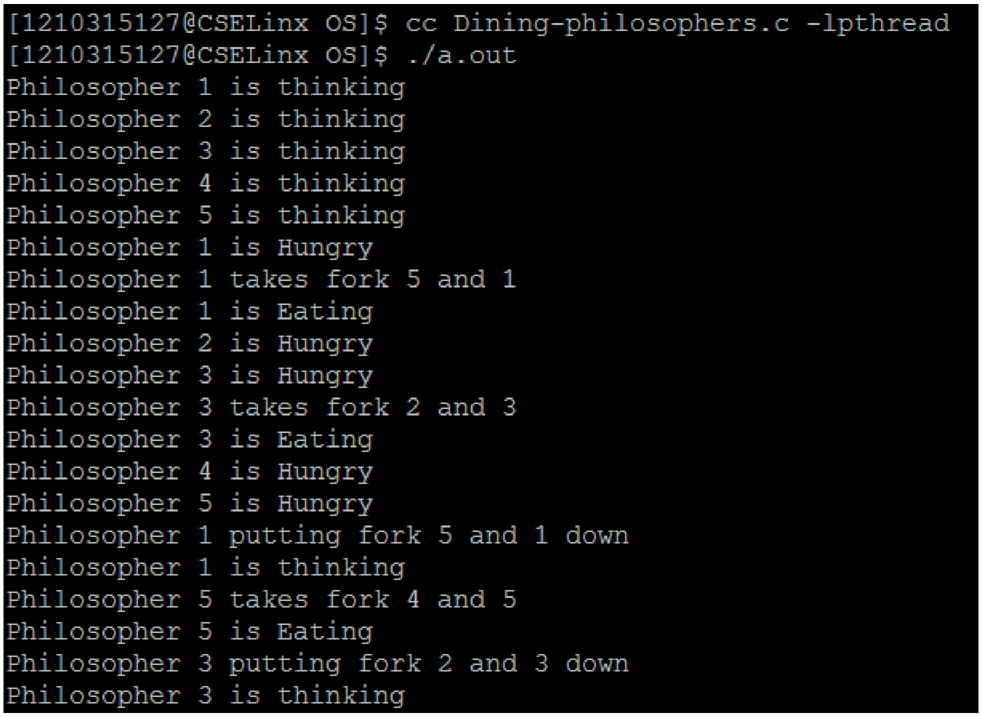
7. Dining Philosophers

```
#include<stdio.h>
#include<semaphore.h>
#include<pthread.h>
#define N 5
#define THINKING 0
#define HUNGRY 1
#define EATING 2
#define LEFT (ph_num+4)%N
#define RIGHT (ph_num+1)%N
sem_t mutex;
sem_t S[N];
void * philospher(void *num);
void take_fork(int);
void put_fork(int);
void test(int);
int state[N];
int phil_num[N]={0,1,2,3,4};
int  main()
{
   int i;
   pthread_t thread_id[N];
   sem_init(&mutex,0,1);
   for(i=0;i<N;i++)
      sem_init(&S[i],0,0);
   for(i=0;i<N;i++)
   {
      pthread_create(&thread_id[i],NULL,philospher,&phil_num[i]);
      printf("Philosopher %d is thinking\n",i+1);
   }
   for(i=0;i<N;i++)
      pthread_join(thread_id[i],NULL);
  return 0;
}
void *philospher(void *num)
```

```
{
   while(1)
   {
      int *i = num;
      sleep(1);
      take_fork(*i);
      sleep(0);
      put_fork(*i);
   }
}
void take_fork(int ph_num)
{
   sem_wait(&mutex);
   state[ph_num] = HUNGRY;
   printf("Philosopher %d is Hungry\n",ph_num+1);
   test(ph_num);
   sem_post(&mutex);
   sem_wait(&S[ph_num]);
   sleep(1);
}
void test(int ph_num)
{
   if (state[ph_num] == HUNGRY && state[LEFT] != EATING && state[RIGHT] !=
EATING)
   {
      state[ph_num] = EATING;
      sleep(2);
      printf("Philosopher %d takes fork %d and
%d\n",ph_num+1,LEFT+1,ph_num+1);
      printf("Philosopher %d is Eating\n",ph_num+1);
      sem_post(&S[ph_num]);
   }
}

void put_fork(int ph_num)
{
   sem_wait(&mutex);
```

```
    state[ph_num] = THINKING;
    printf("Philosopher %d putting fork %d and %d
down\n",ph_num+1,LEFT+1,ph_num+1);
    printf("Philosopher %d is thinking\n",ph_num+1);
    test(LEFT);
    test(RIGHT);
    sem_post(&mutex);
}
```

Output:



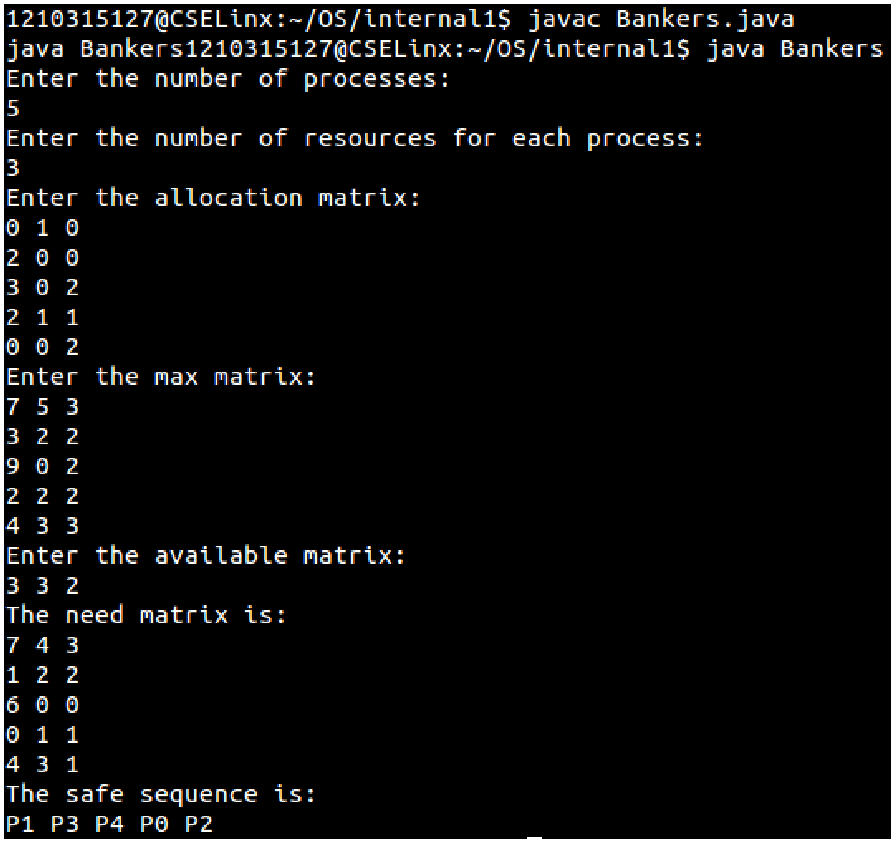
```
[1210315127@CSELinx OS]$ cc Dining-philosophers.c -lpthread
[1210315127@CSELinx OS]$ ./a.out
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 1 is Hungry
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 2 is Hungry
Philosopher 3 is Hungry
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 4 is Hungry
Philosopher 5 is Hungry
Philosopher 1 putting fork 5 and 1 down
Philosopher 1 is thinking
Philosopher 5 takes fork 4 and 5
Philosopher 5 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
```

8. Banker's Safety algorithm implementation

Bankers.java

```
import java.util.Scanner;
class Bankers
{
 public static void main(String args[])
 {
  Scanner s = new Scanner(System.in);
  System.out.println("Enter the number of processes:");
  int n = s.nextInt(); /* n is the number of processes */
  System.out.println("Enter the number of resources for each process:");
  int r = s.nextInt(); /* r is the number of resources required for each process */
  System.out.println("Enter the allocation matrix:");
  int i,j;
  int [][]allocation;
  allocation = new int [n][r];
  for(i=0;i<n;i++)
   for(j=0;j<r;j++)
    allocation[i][j] = s.nextInt();
  System.out.println("Enter the max matrix:");
  int [][]max;
  max = new int [n][r];
  for(i=0;i<n;i++)
   for(j=0;j<r;j++)
    max[i][j] = s.nextInt();
  System.out.println("Enter the available matrix:");
  int []available;
  available = new int [r];
  for(i=0;i<r;i++)
   available[i] = s.nextInt();
  System.out.println("The need matrix is:");
  int [][]need;
  need = new int [n][r];
  for(i=0;i<n;i++)
   for(j=0;j<r;j++)
    need[i][j] = max[i][j] - allocation[i][j];
  for(i=0;i<n;i++)
   {
    for(j=0;j<r;j++)
     System.out.print(need[i][j]+" ");
    System.out.println("");
   }
  int count = 0, l = 0;
  int condition=0;
  int []p;
  p = new int [n];
  int []q;
  q = new int [n];
  System.out.println("The safe sequence is:");
  for(i=0;i<n;i++)
   {
     for(j=0;j<r;j++)
     {
      if(need[i][j] <= available[j])
       condition++;
     }
     if(condition == 3)
     {
       p[i] = i;
       for(j=0;j<r;j++)
        available[j] = available[j] + allocation[i][j];
       System.out.print("P"+p[i]+" ");
       condition = 0;
     }
     else
     {
       q[count] = i;
       condition = 0;
       count++;
       l = count;
     }
   }
  for(count=0;count<l;count++)
   {
    for(j=0;j<r;j++)
    {
     if(need[count][j] <= available[j])
      condition++;
    }
    if(condition == 3)
     {      for(j=0;j<r;j++)
        available[j] = available[j] + allocation[count][j];
       System.out.print("P"+q[count]+" ");
       condition = 0;
     }
   }
  System.out.println("");
 }}
```

Output:



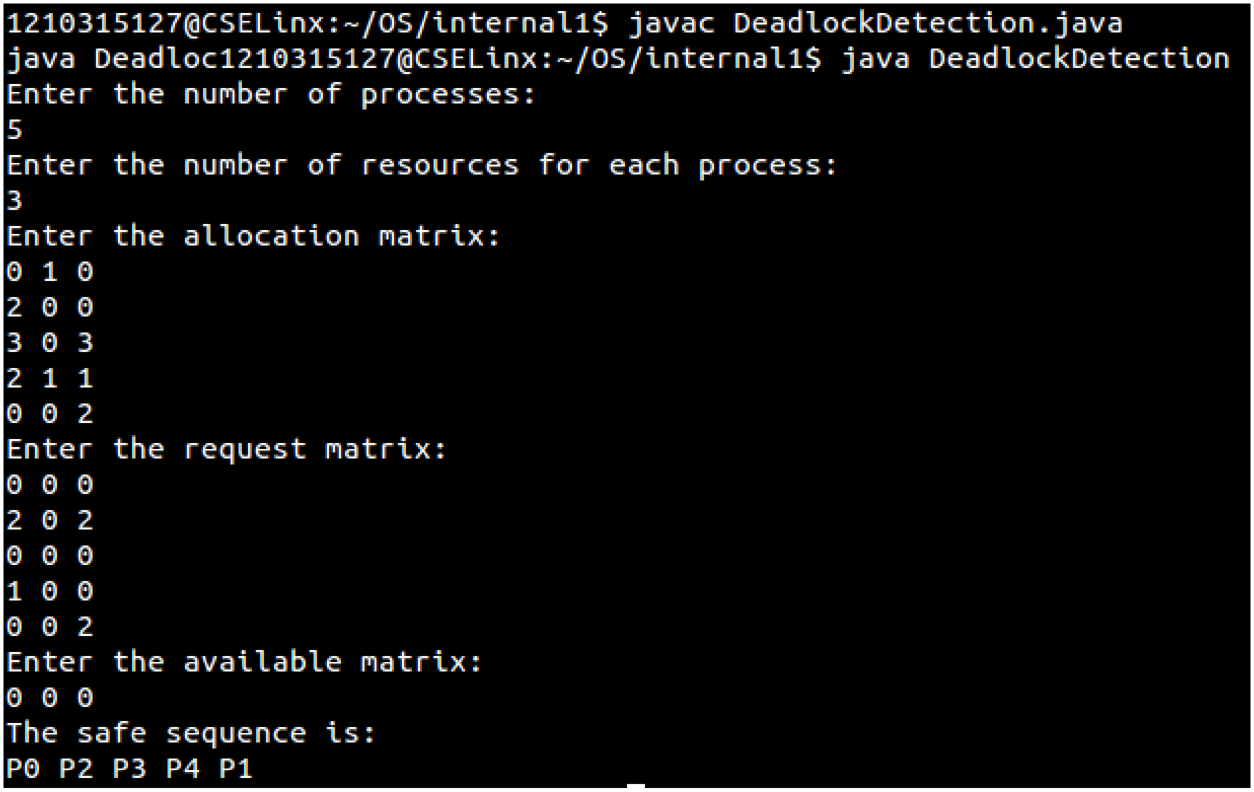
```
1210315127@CSELinx:~/OS/internal1$ javac Bankers.java
java Bankers1210315127@CSELinx:~/OS/internal1$ java Bankers
Enter the number of processes:
5
Enter the number of resources for each process:
3
Enter the allocation matrix:
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the max matrix:
7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter the available matrix:
3 3 2
The need matrix is:
7 4 3
1 2 2
6 0 0
0 1 1
4 3 1
The safe sequence is:
P1 P3 P4 P0 P2
```

9. Deadlock Detection Algorithm implementation

```
DeadlockDetection.java
import java.util.Scanner;
class DeadlockDetection
{
 public static void main(String args[])
 {
  Scanner s = new Scanner(System.in);
  System.out.println("Enter the number of processes:");
  int n = s.nextInt(); /* n is the number of processes */
  System.out.println("Enter the number of resources for each process:");
  int r = s.nextInt(); /* r is the number of resources required for each process */
  System.out.println("Enter the allocation matrix:");
  int i,j;
  int [][]allocation;
  allocation = new int [n][r];
  for(i=0;i<n;i++)
   for(j=0;j<r;j++)
    allocation[i][j] = s.nextInt();
  System.out.println("Enter the request matrix:");
  int [][]request;
  request = new int [n][r];
  for(i=0;i<n;i++)
   for(j=0;j<r;j++)
    request[i][j] = s.nextInt();
  System.out.println("Enter the available matrix:");
  int []available;
  available = new int [r];
  for(i=0;i<r;i++)
   available[i] = s.nextInt();
  int count = 0, l = 0;
  int condition=0, f = 0;
  int []p;
  int increment =0;
```

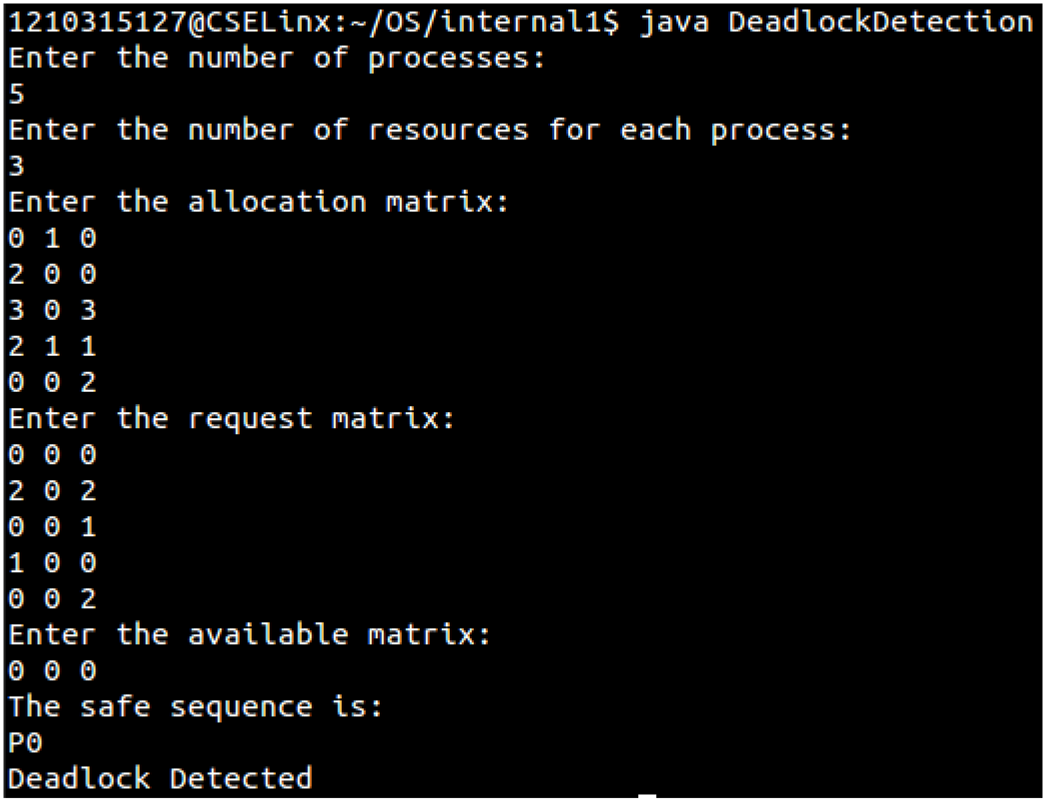
```
p = new int [n];
int []q;
q = new int [n];
System.out.println("The safe sequence is:");
for(i=0;i<n;i++)
 {
   for(j=0;j<r;j++)
   {
    if(request[i][j] <= available[j])
     {
     condition++;
     }
   }
   if(condition == 3)
   {
     p[i] = i;
     for(j=0;j<r;j++)
      available[j] = available[j] + allocation[i][j];
     System.out.print("P"+p[i]+" ");
     condition = 0;
     increment++;
   }
   else
   {
     q[count] = i;
     f = q[count];
     condition = 0;
     count++;
     l = count;
   }
 }
for(count=0;count<l;count++)
 {
  for(j=0;j<r;j++)
  {
   if(request[f][j] <= available[j])
    {
     condition++;
     }
   }
   if(condition == 3)
    {
      for(j=0;j<r;j++)
       available[j] = available[j] + allocation[count][j];
      System.out.print("P"+q[count]+" ");
      condition = 0;
      increment++;
    }
  }
 System.out.println("");
 if(increment<5)
  System.out.println("Deadlock Detected");
}
}
```

Output 1:



```
1210315127@CSELinx:~/OS/internal1$ javac DeadlockDetection.java
java Deadloc1210315127@CSELinx:~/OS/internal1$ java DeadlockDetection
Enter the number of processes:
5
Enter the number of resources for each process:
3
Enter the allocation matrix:
0 1 0
2 0 0
3 0 3
2 1 1
0 0 2
Enter the request matrix:
0 0 0
2 0 2
0 0 0
1 0 0
0 0 2
Enter the available matrix:
0 0 0
The safe sequence is:
P0 P2 P3 P4 P1
```

Output 2:

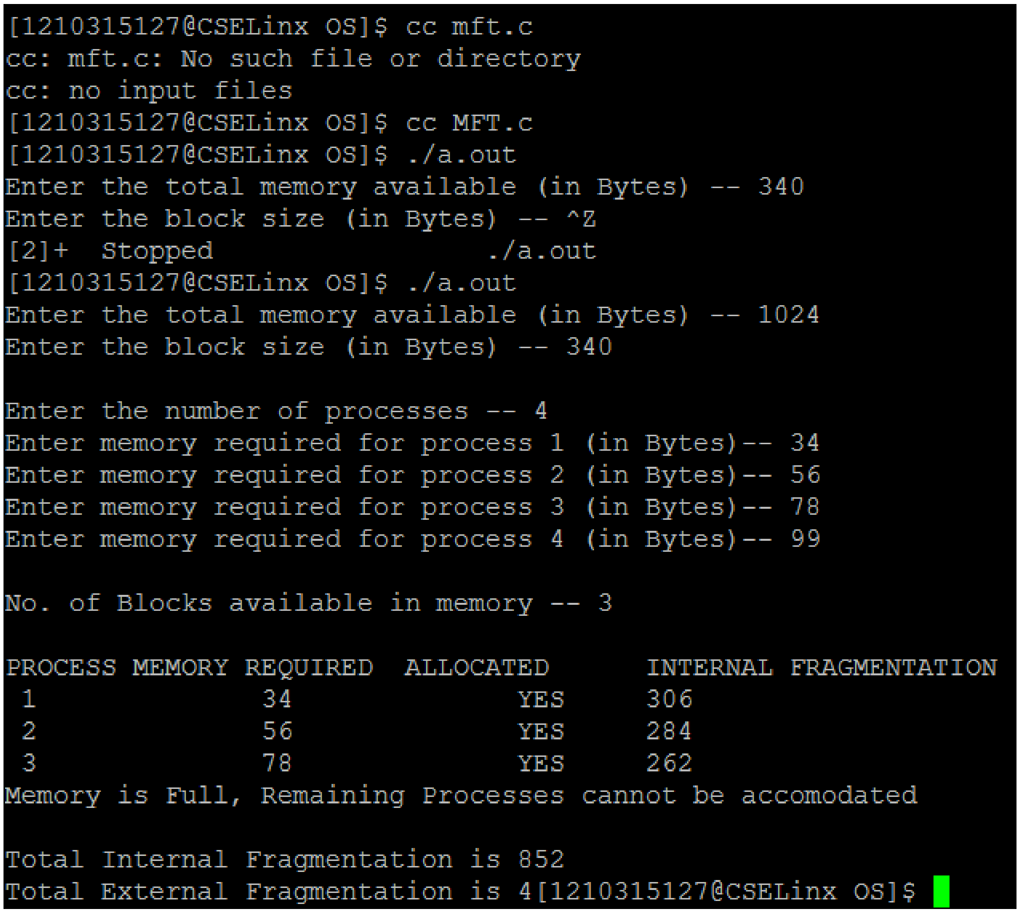


```
1210315127@CSELinx:~/OS/internal1$ java DeadlockDetection
Enter the number of processes:
5
Enter the number of resources for each process:
3
Enter the allocation matrix:
0 1 0
2 0 0
3 0 3
2 1 1
0 0 2
Enter the request matrix:
0 0 0
2 0 2
0 0 1
1 0 0
0 0 2
Enter the available matrix:
0 0 0
The safe sequence is:
P0
Deadlock Detected
```

10. MFT

```
#include<stdio.h>
main()
{
int ms, bs, nob, ef,n, mp[10],tif=0;
int i,p=0;
printf("Enter the total memory available (in Bytes) -- ");
scanf("%d",&ms);
printf("Enter the block size (in Bytes) -- ");
scanf("%d", &bs);
nob=ms/bs;
ef=ms - nob*bs;
printf("\nEnter the number of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter memory required for process %d (in Bytes)-- ",i+1);
scanf("%d",&mp[i]);
}
printf("\nNo. of Blocks available in memory -- %d",nob);
printf("\n\nPROCESS\tMEMORY REQUIRED\t ALLOCATED\tINTERNAL
FRAGMENTATION");
for(i=0;i<n && p<nob;i++)
{
printf("\n %d\t\t%d",i+1,mp[i]);
if(mp[i] > bs)
printf("\t\tNO\t\t---");
else
{
printf("\t\tYES\t%d",bs-mp[i]);
tif = tif + bs-mp[i];
p++;
}
}
if(i<n)
printf("\nMemory is Full, Remaining Processes cannot be accomodated");
printf("\n\nTotal Internal Fragmentation is %d",tif);
printf("\nTotal External Fragmentation is %d",ef);
}
```

Output:



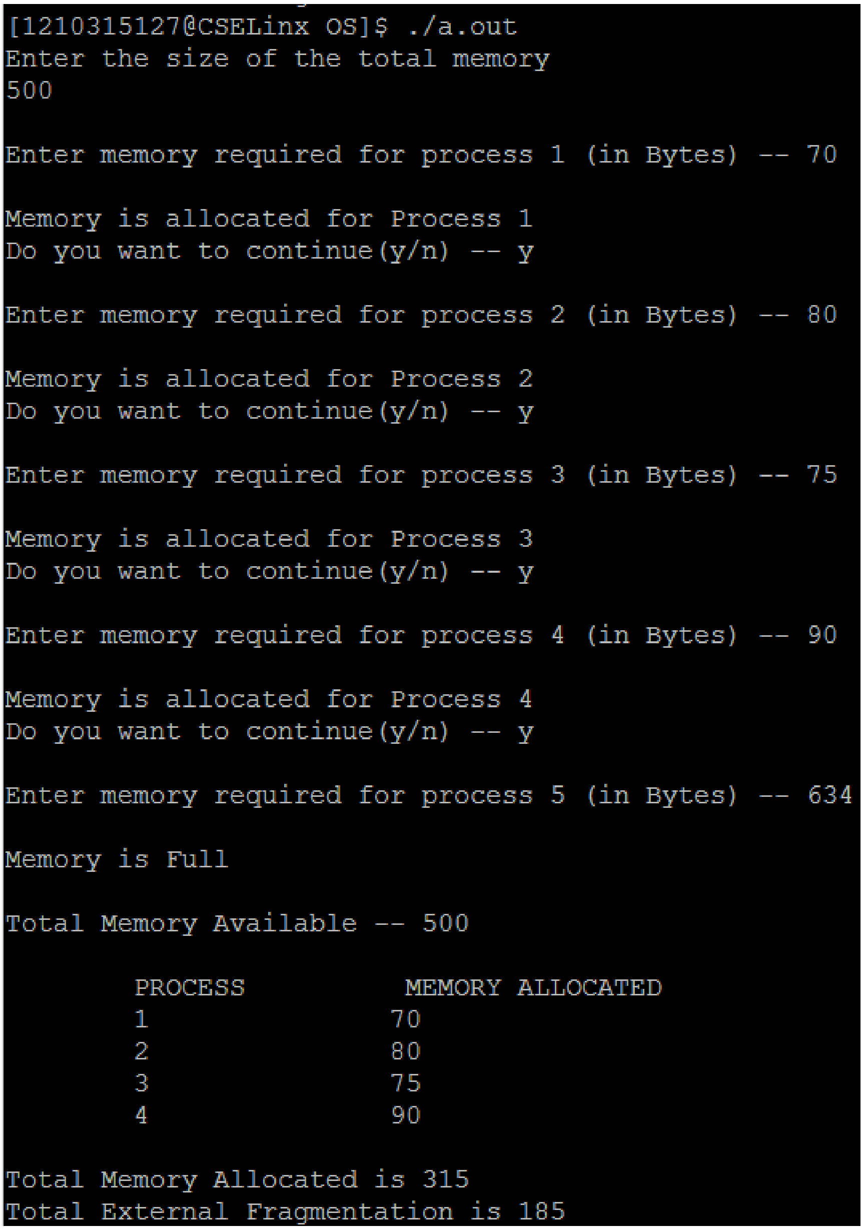
```
[1210315127@CSELinx OS]$ cc mft.c
cc: mft.c: No such file or directory
cc: no input files
[1210315127@CSELinx OS]$ cc MFT.c
[1210315127@CSELinx OS]$ ./a.out
Enter the total memory available (in Bytes) -- 340
Enter the block size (in Bytes) -- ^Z
[2]+  Stopped                 ./a.out
[1210315127@CSELinx OS]$ ./a.out
Enter the total memory available (in Bytes) -- 1024
Enter the block size (in Bytes) -- 340

Enter the number of processes -- 4
Enter memory required for process 1 (in Bytes)-- 34
Enter memory required for process 2 (in Bytes)-- 56
Enter memory required for process 3 (in Bytes)-- 78
Enter memory required for process 4 (in Bytes)-- 99

No. of Blocks available in memory -- 3

PROCESS MEMORY REQUIRED  ALLOCATED      INTERNAL FRAGMENTATION
 1              34              YES     306
 2              56              YES     284
 3              78              YES     262
Memory is Full, Remaining Processes cannot be accomodated

Total Internal Fragmentation is 852
Total External Fragmentation is 4[1210315127@CSELinx OS]$
```

11. MVT

```
#include<stdio.h>
#include<string.h>
int i=0;
int main()
{
int tot,sz,b[20],ch,size[20];
char p[20][20],pro[10];
int j,k,status[10],tot1;
printf("\n Enter total Memory ::");
scanf("%d",&tot);
tot1=tot;
printf("size of os\n");
scanf("%d",&sz);
tot1=tot1-sz;
do
{
printf("\n\t\tMVT");
printf("\n------------------------------------------------------------\n");
printf("\n\t\t1.Allocation\n\t\t2.Deletion\n\t\t3.Display\n\t\t4.Exit");
printf("\n------------------------------------------------------------\n");
printf("\n Enter Ur choice ::");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n Enter the process name %d::",i+1);
scanf("%s",&p[i]);
printf("\n Enter memory size for process%d::",i+1);
scanf("%d",&size[i]);
if(tot1 >=size[i])
{
b[i]=size[i];
tot1=tot1-size[i];
status[i]=1;
```

```
i++;
printf("\n Process is allocated ::");
}
else
printf("\n Memory size is not Available ::");
break;
case 2:
printf("\n Enter a process U want to delete ::");
scanf("%s",&pro);
j=0;
while(j<i)
{
if(strcmp(p[j],pro)==0)
{
status[j]=0;
tot1+=b[j];
b[j]-=size[j];
}
j++;
}
printf("\n Process is deleted ::");
break;
case 3:
printf("\n\n Total Memory size ::%d",tot);
printf("\n\n Memory size of the operating system::%d",sz);
printf("\n\n No.of processes   ::%d",i);
printf("\n---------------------------------------------------------------------------\n");
printf("\npname\tAllocated Memory\tStatus\n");
printf("\n---------------------------------------------------------------------------\n");
for(k=0;k<i;k++)
{
if(b[k]!=0)
{
printf("\n%s\t\t%d\t\t",p[k],b[k]);
if(status[k]==1)
printf("Full");
else
printf("Available");
}
}
printf("\n--------------------------------------------------------------------------\n");
printf("\n Total Available Space ::%d",tot1);
break;
case 4:
return 0;
default:
printf("\n Wrong Choice ::");
}
}while(1);
}
```

Output:



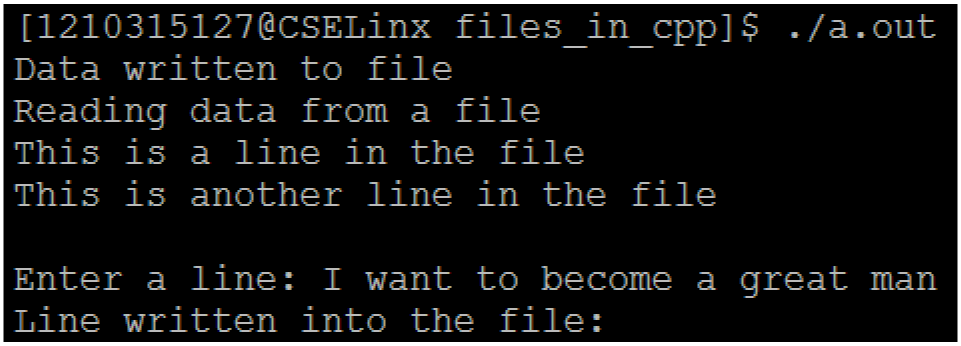
```
[1210315127@CSELinx OS]$ ./a.out
Enter the size of the total memory
500

Enter memory required for process 1 (in Bytes) -- 70

Memory is allocated for Process 1
Do you want to continue(y/n) -- y

Enter memory required for process 2 (in Bytes) -- 80

Memory is allocated for Process 2
Do you want to continue(y/n) -- y

Enter memory required for process 3 (in Bytes) -- 75

Memory is allocated for Process 3
Do you want to continue(y/n) -- y

Enter memory required for process 4 (in Bytes) -- 90

Memory is allocated for Process 4
Do you want to continue(y/n) -- y

Enter memory required for process 5 (in Bytes) -- 634

Memory is Full

Total Memory Available -- 500

        PROCESS          MEMORY ALLOCATED
        1               70
        2               80
        3               75
        4               90

Total Memory Allocated is 315
Total External Fragmentation is 185
```

12. Write a program to perform file operations

Fileoperations.cpp

```
#include<iostream>
#include<fstream>
using namespace std;

class File
{
  char data[100];
  char line[100];
  public:
   void read(), write(), append();
};

void File::read()
{
 ofstream ofile;
 ofile.open("file.txt");
 ofile<<"This is a line in the file"<<endl;
 ofile<<"This is another line in the file"<<endl;
 cout<<"Data written to file"<<endl;
 ofile.close();
}

void File::write()
{
 ifstream ifile;
 ifile.open("file.txt");
 cout<<"Reading data from a file"<<endl;
 while(!ifile.eof())
 {
  ifile.getline(data,100);
  cout<<data<<endl;
 }
 ifile.close();
}

void File::append()
{
 fstream file;
 file.open("file.txt",ios::out|ios::app);
 if(file.fail())
  cout<<"Error opening file"<<endl;
 else
 {
  cout<<"Enter a line: ";
  cin.getline(line,100);
  file<<line<<endl;
  cout<<"Line written into the file:"<<endl;
 }
}

int main()
{
 File f;
 f.read();
 f.write();
 f.append();
}
```

Output:

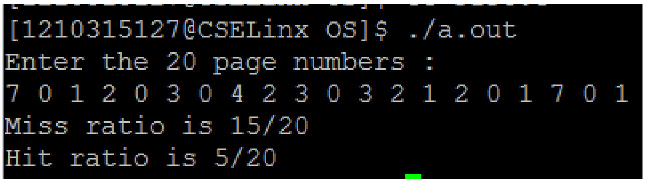


```
[1210315127@CSELinx files_in_cpp]$ ./a.out
Data written to file
Reading data from a file
This is a line in the file
This is another line in the file

Enter a line: I want to become a great man
Line written into the file:
```

13.Page replacement algorithm- FIFO

```
#include<stdio.h>
int main()
{
int i,j=0,n=20,a[50],mm[3],k,available,miss=0;
printf("Enter the 20 page numbers :\n");
for(i=0;i<n;i++)
 scanf("%d",&a[i]);
for(i=0;i<3;i++)
 mm[i]= -1 ;
for(i=0;i<n;i++)
{
 available=0;
 for(k=0;k<3;k++)
 if(mm[k]==a[i])
   available=1;
 if(available==0)
 {
  mm[j]=a[i];
  j=(j+1)%3;
  miss++;
 }
}
printf("Miss ratio is %d/20\n",miss);
printf("Hit ratio is %d/20\n",20-miss);
return 0;
}
```

Output:



```
[1210315127@CSELinx OS]$ ./a.out
Enter the 20 page numbers :
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Miss ratio is 15/20
Hit ratio is 5/20
```

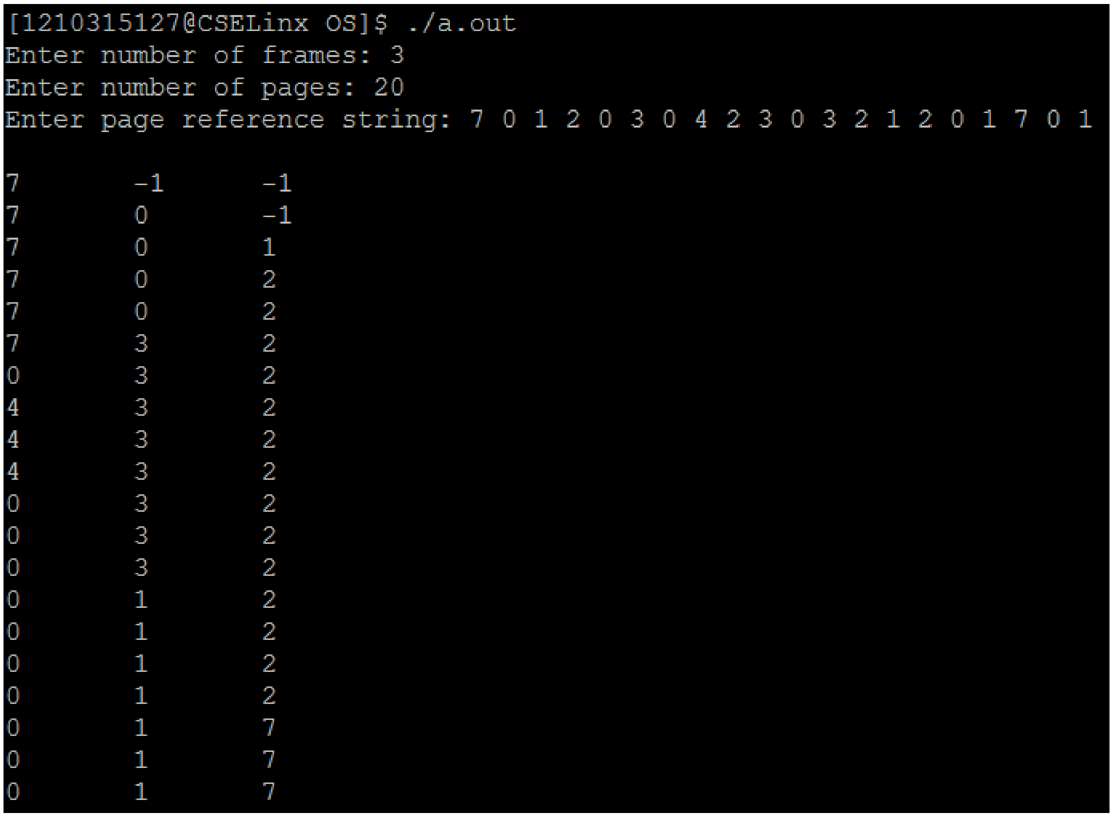
14. Page replacement algorithm – Optimal

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

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

    printf("Enter page reference string: ");

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

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

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

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

        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
```

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

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

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

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

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

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

        for(j = 1; j < no_of_frames; ++j){
            if(temp[j] > max){
                max = temp[j];
```

```
                        pos = j;
                    }
                }
            }
            frames[pos] = pages[i];
            faults++;
        }
        printf("\n");
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }
    printf("\n\nTotal Page Faults = %d", faults);
}
```

Output:

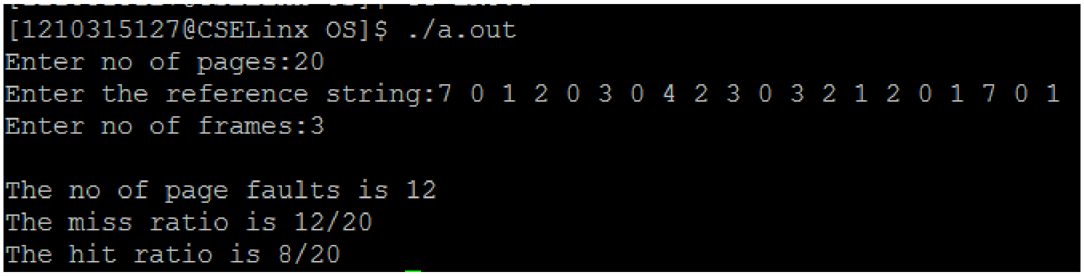


```
[12103151270CSELinx OS]$ ./a.out
Enter number of frames: 3
Enter number of pages: 20
Enter page reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

7       -1      -1
7       0       -1
7       0       1
7       0       2
7       0       2
7       3       2
0       3       2
4       3       2
4       3       2
4       3       2
0       3       2
0       3       2
0       3       2
0       1       2
0       1       2
0       1       2
0       1       2
0       1       7
0       1       7
0       1       7
```

15. Page replacement algorithm - LRU

```
#include<stdio.h>
main()
{
 int q[20],p[50],c=0,c1,d,f,i,j,k=0,n,r,t,b[20],c2[20];
 printf("Enter no of pages:");
 scanf("%d",&n);
 printf("Enter the reference string:");
 for(i=0;i<n;i++)
 scanf("%d",&p[i]);
 printf("Enter no of frames:");
 scanf("%d",&f);
 q[k]=p[k];  c++;  k++;
 for(i=1;i<n;i++)
 {
  c1=0;
  for(j=0;j<f;j++)
  {   if(p[i]!=q[j])
    c1++;
  }
  if(c1==f)
  {
   c++;
   if(k<f)
   {
    q[k]=p[i];
    k++;
   }
   else
   {
    for(r=0;r<f;r++)
    {
     c2[r]=0;
     for(j=i-1;j<n;j--)
     {
      if(q[r]!=p[j])
       c2[r]++;
      else
       break;
     }
    }
    for(r=0;r<f;r++)
     b[r]=c2[r];
    for(r=0;r<f;r++)
     for(j=r;j<f;j++)
      if(b[r]<b[j])
      {
       t=b[r];
       b[r]=b[j];
       b[j]=t;
      }
    for(r=0;r<f;r++)
    {
     if(c2[r]==b[0])
      q[r]=p[i];
    }
   }
  }
 }
 printf("\nThe no of page faults is %d\n",c);
 printf("The miss ratio is %d/20\n",c);
 printf("The hit ratio is %d/20\n",20-c);
}
```

Output:

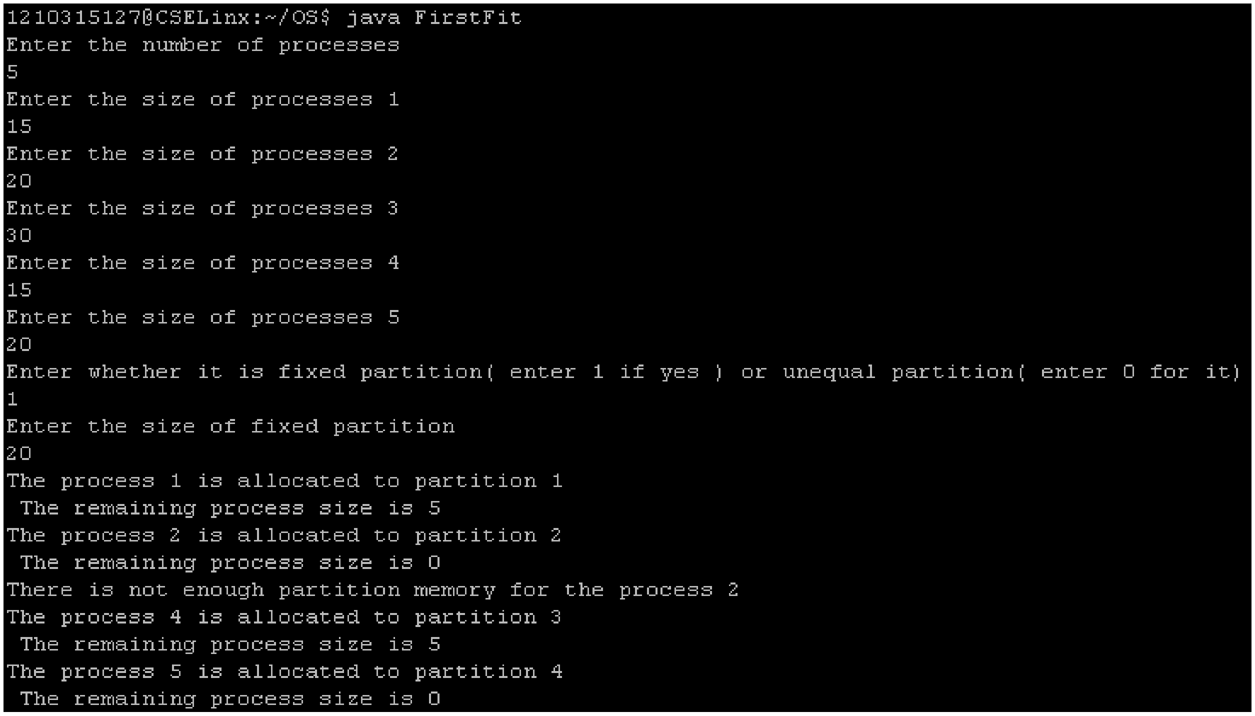


```
[1210315127@CSELinx OS]$ ./a.out
Enter no of pages:20
Enter the reference string:7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter no of frames:3

The no of page faults is 12
The miss ratio is 12/20
The hit ratio is 8/20
```

16. First Fit algorithm

```
import java.util.*;
class FirstFit
{
 public static void main(String args[])
 {
  int p,m;
  System.out.println("Enter the number of processes");
  Scanner s = new Scanner(System.in);
  p = s.nextInt();
  m = p;
  int i;
  int pr[] = new int[p];
  int mr[] = new int[m];
  for(i=0;i<p;i++)
  {
   System.out.println("Enter the size of processes "+(i+1));
   pr[i] = s.nextInt();
  }
  System.out.println("Enter whether it is fixed partition( enter 1 if yes ) or unequal
partition( enter 0 for it)");
  int f;
  f = s.nextInt();
  if(f == 0)
  {
   for(i=0;i<p;i++)
   {
    System.out.println("Enter the size of the partition "+(i+1));
        mr[i] = s.nextInt();
   }  }
  else
  {
   System.out.println("Enter the size of fixed partition");
   mr[0] = s.nextInt();
   for(i=0;i<p;i++)
   {
    mr[i] = mr[0];
   }
  }
 int j;
 for(i=0;i<p;i++)
 {  for(j=0;j<m;j++)
  {
   if(mr[j]>=pr[i])
   {
    mr[j] -= pr[i];
    System.out.println("The process "+(i+1)+" is allocated to partition "+(j+1)+"
\n"+" The remaining process size is "+mr[j]);
    break;
   }
  }
  if(j==m)
  {
   System.out.println("There is not enough partition memory for the process "+i);
  }  } }}
```

Output:



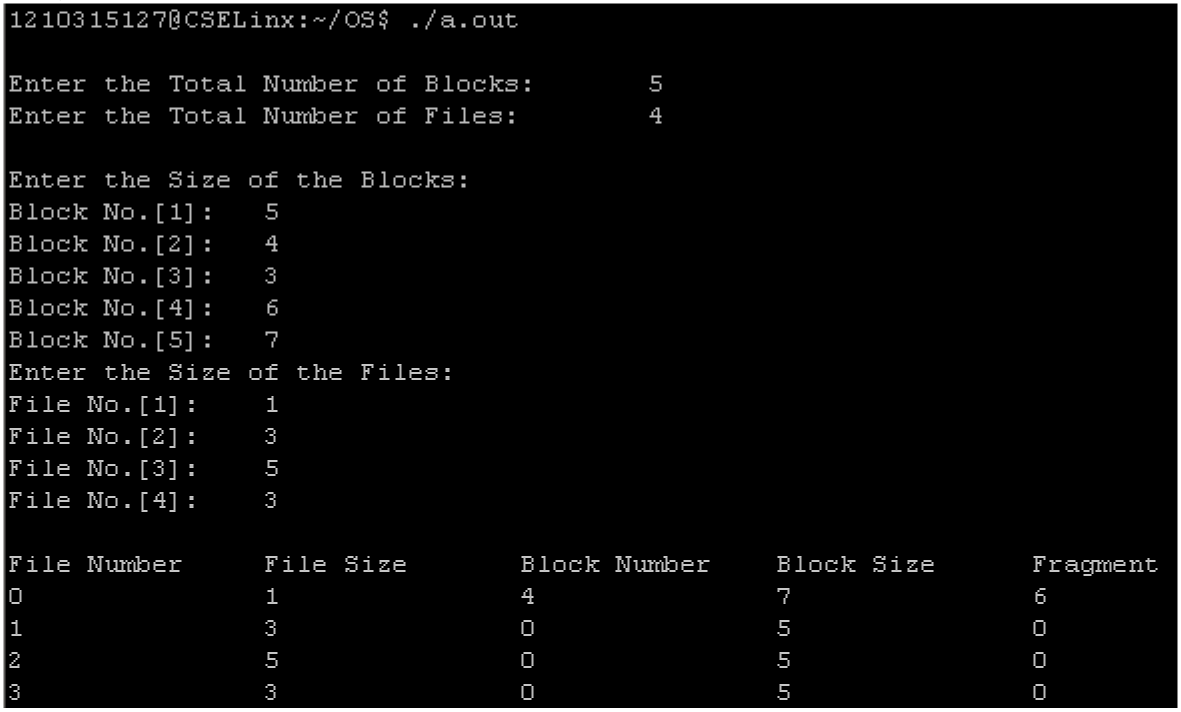
```
1210315127@CSELinx:~/OS$ java FirstFit
Enter the number of processes
5
Enter the size of processes 1
15
Enter the size of processes 2
20
Enter the size of processes 3
30
Enter the size of processes 4
15
Enter the size of processes 5
20
Enter whether it is fixed partition( enter 1 if yes ) or unequal partition( enter 0 for it)
1
Enter the size of fixed partition
20
The process 1 is allocated to partition 1
 The remaining process size is 5
The process 2 is allocated to partition 2
 The remaining process size is 0
There is not enough partition memory for the process 2
The process 4 is allocated to partition 3
 The remaining process size is 5
The process 5 is allocated to partition 4
 The remaining process size is 0
```

17. Worst Fit algorithm

```
#include<stdio.h>

int main()
{
	int fragments[10], blocks[10], files[10];
	int m, n, number_of_blocks, number_of_files, temp, top = 0;
	static int block_arr[10], file_arr[10];
	printf("\nEnter the Total Number of Blocks:\t");
	scanf("%d",&number_of_blocks);
	printf("Enter the Total Number of Files:\t");
	scanf("%d",&number_of_files);
	printf("\nEnter the Size of the Blocks:\n");
	for(m = 0; m < number_of_blocks; m++)
	{
		printf("Block No.[%d]:\t", m + 1);
		scanf("%d", &blocks[m]);
	}
	printf("Enter the Size of the Files:\n");
	for(m = 0; m < number_of_files; m++)
	{
		printf("File No.[%d]:\t", m + 1);
		scanf("%d", &files[m]);
	}
	for(m = 0; m < number_of_files; m++)
	{
		for(n = 0; n < number_of_blocks; n++)
		{
			if(block_arr[n] != 1)
			{
				temp = blocks[n] - files[m];
				if(temp >= 0)
				{
					if(top < temp)
					{
```

```
                        file_arr[m] = n;
                        top = temp;
                    }
                }
            }
            fragments[m] = top;
            block_arr[file_arr[m]] = 1;
            top = 0;
        }
    }
    printf("\nFile Number\tFile Size\tBlock Number\tBlock Size\tFragment");
    for(m = 0; m < number_of_files; m++)
    {
        printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d", m, files[m], file_arr[m],
blocks[file_arr[m]], fragments[m]);
    }
    printf("\n");
    return 0;
}
```

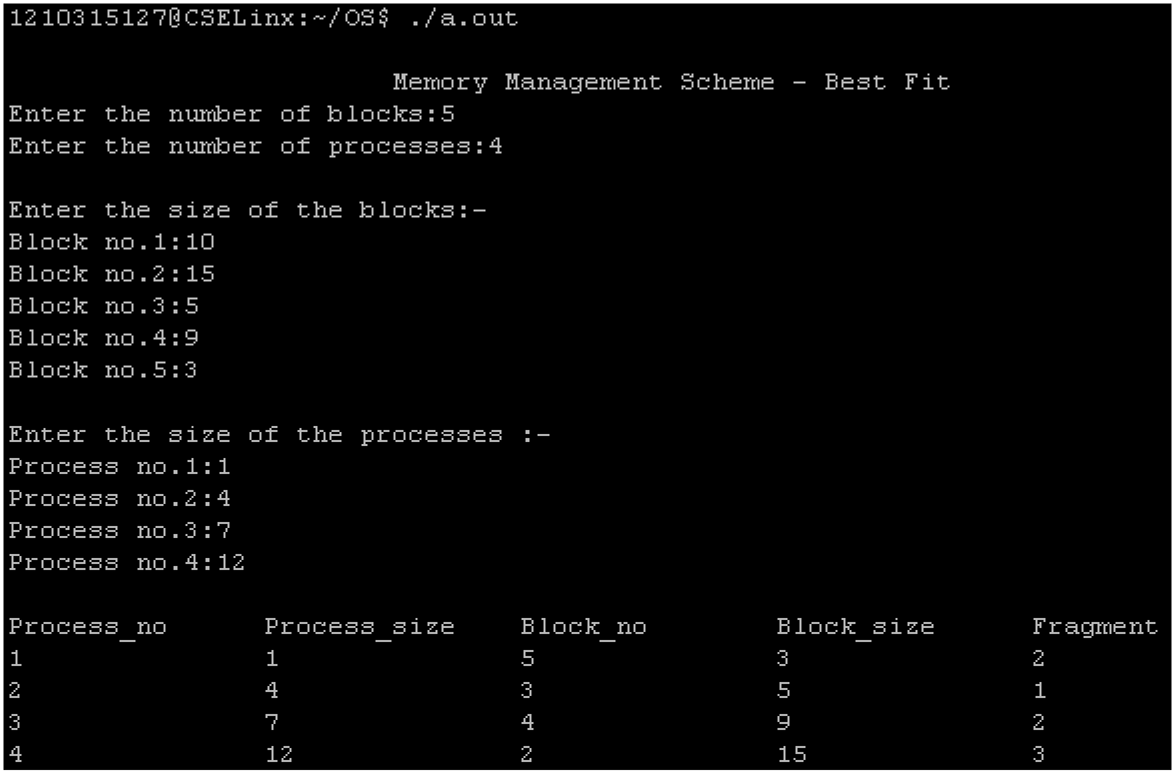
Output:



```
1210315127@CSELinx:~/OS$ ./a.out

Enter the Total Number of Blocks:       5
Enter the Total Number of Files:        4

Enter the Size of the Blocks:
Block No.[1]:   5
Block No.[2]:   4
Block No.[3]:   3
Block No.[4]:   6
Block No.[5]:   7
Enter the Size of the Files:
File No.[1]:    1
File No.[2]:    3
File No.[3]:    5
File No.[4]:    3

File Number     File Size       Block Number    Block Size      Fragment
0               1               4               7               6
1               3               0               5               0
2               5               0               5               0
3               3               0               5               0
```

18. Best fit algorithm

```
#include<stdio.h>

void main()
{
   int fragment[20],b[20],p[20],i,j,nb,np,temp,lowest=9999;
   static int barray[20],parray[20];

   printf("\n\t\t\tMemory Management Scheme - Best Fit");
   printf("\nEnter the number of blocks:");
   scanf("%d",&nb);
   printf("Enter the number of processes:");
   scanf("%d",&np);

   printf("\nEnter the size of the blocks:-\n");
   for(i=1;i<=nb;i++)
   {
      printf("Block no.%d:",i);
      scanf("%d",&b[i]);
   }

   printf("\nEnter the size of the processes :-\n");
   for(i=1;i<=np;i++)
   {
      printf("Process no.%d:",i);
      scanf("%d",&p[i]);
   }

   for(i=1;i<=np;i++)
   {
      for(j=1;j<=nb;j++)
      {
         if(barray[j]!=1)
         {
            temp=b[j]-p[i];
            if(temp>=0)
               if(lowest>temp)
               {
                  parray[i]=j;
                  lowest=temp;
               }
         }
      }

      fragment[i]=lowest;
      barray[parray[i]]=1;
      lowest=10000;
   }

   printf("\nProcess_no\tProcess_size\tBlock_no\tBlock_size\tFragment");
   for(i=1;i<=np && parray[i]!=0;i++)

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



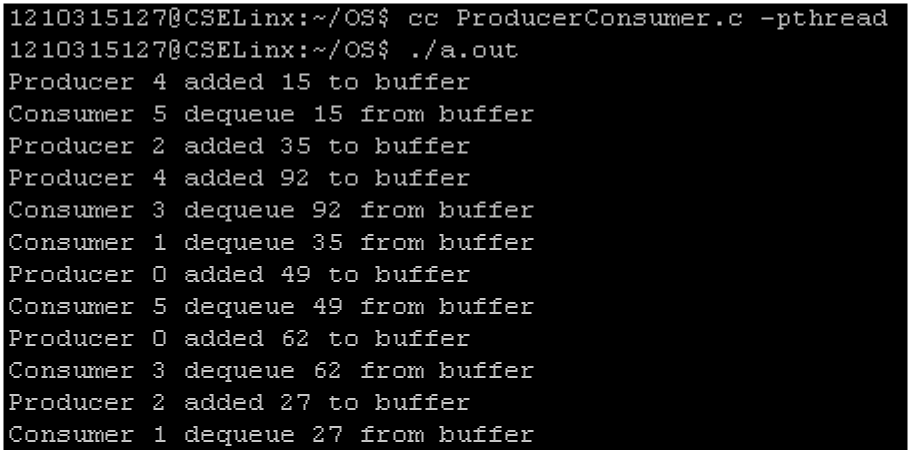
```
1210315127@CSELinx:~/OS$ ./a.out

                        Memory Management Scheme - Best Fit
Enter the number of blocks:5
Enter the number of processes:4

Enter the size of the blocks:-
Block no.1:10
Block no.2:15
Block no.3:5
Block no.4:9
Block no.5:3

Enter the size of the processes :-
Process no.1:1
Process no.2:4
Process no.3:7
Process no.4:12

Process_no      Process_size    Block_no        Block_size      Fragment
1               1               5               3               2
2               4               3               5               1
3               7               4               9               2
4               12              2               15              3
```

19.Producer-Consumer problem

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <semaphore.h>
#define SIZE 5
#define NUMB_THREADS 6
#define PRODUCER_LOOPS 2
typedef int buffer_t;
buffer_t buffer[SIZE];
int buffer_index;
pthread_mutex_t buffer_mutex;
sem_t full_sem;
sem_t empty_sem;
void insertbuffer(buffer_t value) {
    if (buffer_index < SIZE) {
        buffer[buffer_index++] = value;
    } else {
        printf("Buffer overflow\n");
    }
}
buffer_t dequeuebuffer() {
    if (buffer_index > 0) {
        return buffer[--buffer_index]; // buffer_index-- would be error!
    } else {
        printf("Buffer underflow\n");
    }
    return 0;
}
void *producer(void *thread_n) {
    int thread_numb = *(int *)thread_n;
    buffer_t value;
    int i=0;
    while (i++ < PRODUCER_LOOPS) {
        sleep(rand() % 10);
```

```
      value = rand() % 100;
      sem_wait(&full_sem); // sem=0: wait. sem>0: go and decrement it
      pthread_mutex_lock(&buffer_mutex);
      insertbuffer(value);
      pthread_mutex_unlock(&buffer_mutex);
      sem_post(&empty_sem); // post (increment) emptybuffer semaphore
      printf("Producer %d added %d to buffer\n", thread_numb, value);
   }
   pthread_exit(0);
}
void *consumer(void *thread_n) {
   int thread_numb = *(int *)thread_n;
   buffer_t value;
   int i=0;
   while (i++ < PRODUCER_LOOPS) {
      sem_wait(&empty_sem);
      pthread_mutex_lock(&buffer_mutex);
      value = dequeuebuffer(value);
      pthread_mutex_unlock(&buffer_mutex);
      sem_post(&full_sem); // post (increment) fullbuffer semaphore
      printf("Consumer %d dequeue %d from buffer\n", thread_numb, value);
  }
   pthread_exit(0);
}
int main(int argc, int **argv) {
   buffer_index = 0;

   pthread_mutex_init(&buffer_mutex, NULL);
   sem_init(&full_sem, // sem_t *sem
          0, // int pshared. 0 = shared between threads of process,  1 = shared
between processes
          SIZE); // unsigned int value. Initial value
   sem_init(&empty_sem, 0,0);
   pthread_t thread[NUMB_THREADS];
   int thread_numb[NUMB_THREADS];
   int i;
   for (i = 0; i < NUMB_THREADS; ) {
     thread_numb[i] = i;
     pthread_create(thread + i, // pthread_t *t
                 NULL, // const pthread_attr_t *attr
                 producer, // void *(*start_routine) (void *)
                 thread_numb + i);  // void *arg
     i++;
     thread_numb[i] = i;
    // playing a bit with thread and thread_numb pointers...
       pthread_create(&thread[i], // pthread_t *t
       NULL, // const pthread_attr_t *attr
        consumer, // void *(*start_routine) (void *)
       &thread_numb[i]);  // void *arg
         i++;
      }
   for (i = 0; i < NUMB_THREADS; i++)
    pthread_join(thread[i], NULL);
    pthread_mutex_destroy(&buffer_mutex);
    sem_destroy(&full_sem);
    sem_destroy(&empty_sem);
     return 0;
  }
```

Output:



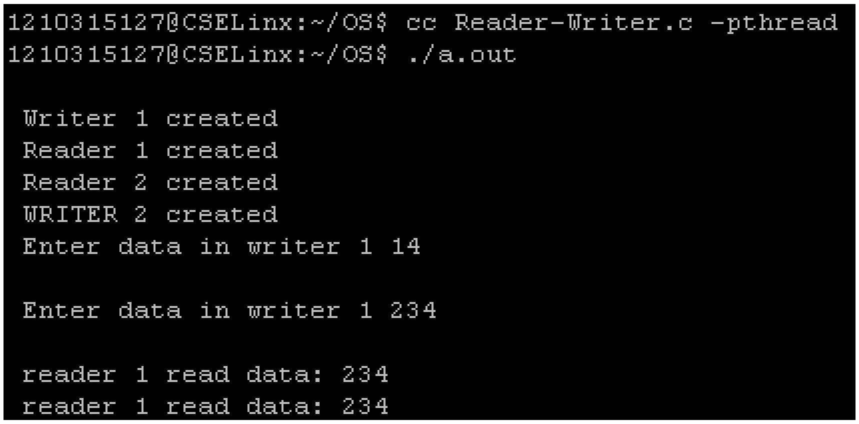
```
1210315127@CSELinx:~/OS$ cc ProducerConsumer.c -pthread
1210315127@CSELinx:~/OS$ ./a.out
Producer 4 added 15 to buffer
Consumer 5 dequeue 15 from buffer
Producer 2 added 35 to buffer
Producer 4 added 92 to buffer
Consumer 3 dequeue 92 from buffer
Consumer 1 dequeue 35 from buffer
Producer 0 added 49 to buffer
Consumer 5 dequeue 49 from buffer
Producer 0 added 62 to buffer
Consumer 3 dequeue 62 from buffer
Producer 2 added 27 to buffer
Consumer 1 dequeue 27 from buffer
```

20. Reader - Writer problem

```
#include<semaphore.h>
#include<pthread.h>
#include<stdio.h>
int rc=0,wc=0,val;
pthread_mutex_t mutex1,mwrite,mread,rallow;
pthread_t tr1,tr2,tw1,tw2;
pthread_attr_t tr1attr,tr2attr,tw1attr,tw2attr;
void *writer();
void *reader();
int main()
{
 pthread_mutex_init(&mwrite,NULL);
 pthread_mutex_init(&mread,NULL);
 pthread_mutex_init(&rallow,NULL);
 pthread_mutex_init(&mutex1,NULL);
 pthread_attr_init(&tw1attr);
 pthread_attr_init(&tr1attr);
 pthread_attr_init(&tr2attr);
 pthread_attr_init(&tw2attr);
 printf("\n Writer 1 created");
 pthread_create(&tw1,&tw1attr,writer,NULL);
 printf("\n Reader 1 created");
 pthread_create(&tr1,&tr1attr,reader,NULL);
 printf("\n Reader 2 created");
 pthread_create(&tr2,&tr2attr,reader,NULL);
 printf("\n WRITER 2 created");
 pthread_create(&tw2,&tw2attr,writer,NULL);
 pthread_join(tw1,NULL);
 pthread_join(tr1,NULL);
 pthread_join(tr2,NULL);
 pthread_join(tw2,NULL);
 printf("\n");
 return 0;
}
```

```
void *writer()
{
 pthread_mutex_lock(&mwrite);
 wc++;
 if(wc==1)
 pthread_mutex_lock(&rallow);
 pthread_mutex_unlock(&mwrite);
 pthread_mutex_lock(&mutex1);
 printf("\n Enter data in writer %d ",wc);
 scanf("%d",&val);
 pthread_mutex_unlock(&mutex1);
 pthread_mutex_lock(&mwrite);
 wc--;
 if(wc==0)
  pthread_mutex_unlock(&rallow);
 pthread_mutex_unlock(&mwrite);
 pthread_exit(0);
}
void *reader()
{
 pthread_mutex_lock(&rallow);
 pthread_mutex_lock(&mread);
 rc++;
 if(rc==1)
 pthread_mutex_lock(&mutex1);
 pthread_mutex_unlock(&mread);
 pthread_mutex_unlock(&rallow);
 printf("\n reader %d read data: %d ",rc,val);
 pthread_mutex_lock(&mread);
 rc--;
 if(rc==0)
  pthread_mutex_unlock(&mutex1);
 pthread_mutex_unlock(&mread);
 pthread_exit(0);
}
```

Output:



```
1210315127@CSELinx:~/OS$ cc Reader-Writer.c -pthread
1210315127@CSELinx:~/OS$ ./a.out

 Writer 1 created
 Reader 1 created
 Reader 2 created
 WRITER 2 created
 Enter data in writer 1 14

 Enter data in writer 1 234

 reader 1 read data: 234
 reader 1 read data: 234
```