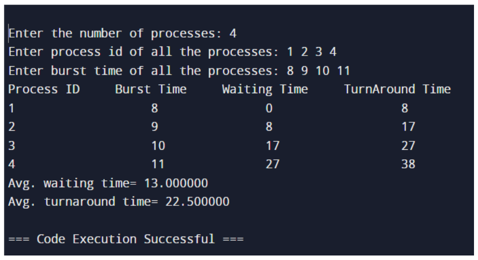
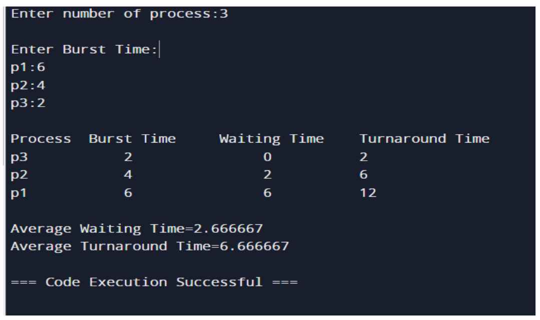
```c
#include <stdio.h>
int main()
{
    int pid[15];
    int bt[15];
    int n;
    printf("Enter the number of processes: ");
    scanf("%d",&n);
    printf("Enter process id of all the processes: ");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&pid[i]);
    }
    printf("Enter burst time of all the processes: ");
    for(int i=0;i<n;i++)
    {
   scanf("%d",&bt[i]);
    }
    int i, wt[n];
    wt[0]=0;
    for(i=1; i<n; i++)
    {
        wt[i]= bt[i-1]+ wt[i-1];
    }
    printf("Process ID     Burst Time     Waiting Time     TurnAround Time\n");
    float twt=0.0;
    float tat= 0.0;
    for(i=0; i<n; i++)
    {
        printf("%d\t\t", pid[i]);
        printf("%d\t\t", bt[i]);
        printf("%d\t\t", wt[i]);
        printf("%d\t\t", bt[i]+wt[i]);
        printf("\n");
       twt += wt[i];
       tat += (wt[i]+bt[i]);
    }
    float att,awt;
    awt = twt/n;
    att = tat/n;
    printf("Avg. waiting time= %f\n",awt);
    printf("Avg. turnaround time= %f",att);
}
```

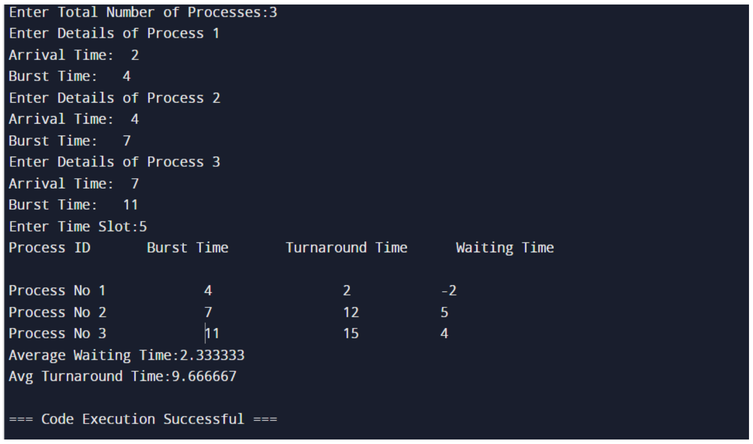


```
Enter the number of processes: 4
Enter process id of all the processes: 1 2 3 4
Enter burst time of all the processes: 8 9 10 11
Process ID     Burst Time     Waiting Time     TurnAround Time
1                   8               0                 8
2                   9               8                 17
3                   10              17                27
4                   11              27                38
Avg. waiting time= 13.000000
Avg. turnaround time= 22.500000

=== Code Execution Successful ===
#include<stdio.h>
int main()
{
    int bt[20],p[20],wt[20],tat[20],i,j,n,total=0,totalT=0,pos,temp;
    float avg_wt,avg_tat;
    printf("Enter number of process:");
    scanf("%d",&n);
    printf("\nEnter Burst Time:\n");
    for(i=0;i<n;i++)
    {
        printf("p%d:",i+1);
        scanf("%d",&bt[i]);
        p[i]=i+1;
    }
    for(i=0;i<n;i++)
    {
        pos=i;
        for(j=i+1;j<n;j++)
        {
   if(bt[j]<bt[pos])
                pos=j;
        }
        temp=bt[i];
        bt[i]=bt[pos];
        bt[pos]=temp;
        temp=p[i];
        p[i]=p[pos];
        p[pos]=temp;
    }
    wt[0]=0;
    for(i=1;i<n;i++)
    {
        wt[i]=0;
        for(j=0;j<i;j++)
        wt[i]total+=wt[i];
    }
    avg_wt=(float)total/n;
    printf("\nProcess\t Burst Time \tWaiting Time\tTurnaround Time");
    for(i=0;i<n;i++)
    {
        tat[i]=bt[i]+wt[i];
        totalT+=tat[i];
        printf("\np%d\t\t %d\t\t %d\t\t\t%d",p[i],bt[i],wt[i],tat[i]);
    }
   avg_tat=(float)totalT/n;
    printf("\n\nAverage Waiting Time=%f",avg_wt);
    printf("\nAverage Turnaround Time=%f",avg_tat);
}
```

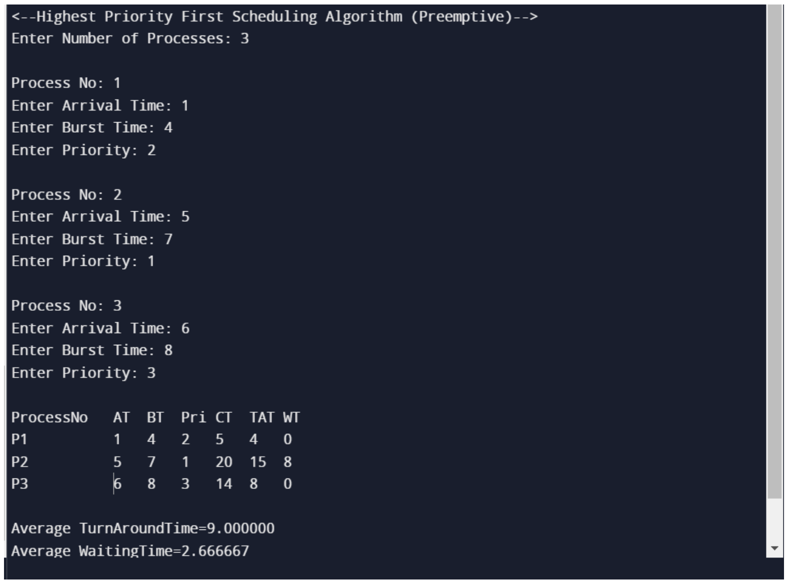


```
Enter number of process:3

Enter Burst Time:
p1:6
p2:4
p3:2

Process  Burst Time     Waiting Time    Turnaround Time
p3          2               0          2
p2          4               2          6
p1          6               6          12

Average Waiting Time=2.666667
Average Turnaround Time=6.666667

=== Code Execution Successful ===
#include<stdio.h>
#define MIN -9999;
struct proc
{
    int no,at,bt,rt,ct,wt,tat,pri,temp;
};
struct proc read(int i)
{
    struct proc p;
    printf("\nProcess No: %d\n",i);
    p.no=i;
    printf("Enter Arrival Time: ");
    scanf("%d",&p.at);
    printf("Enter Burst Time: ");
    scanf("%d",&p.bt);
    p.rt=p.bt;
    printf("Enter Priority: ");
    scanf("%d",&p.pri);
    p.temp=p.pri;
    return p;
}
void main()
{
    int i,n,c,remaining,max_val,max_index;
    struct proc p[10],temp;
    float avgtat=0,avgwt=0;
    printf("<--Highest Priority First Scheduling Algorithm (Preemptive)-->\n");
    printf("Enter Number of Processes: ");
    scanf("%d",&n);
    for(int i=0;i<n;i++)
        p[i]=read(i+1);
    remaining=n;
                    for(int i=0;i<n-1;i++)
        for(int j=0;j<n-i-1;j++)
            if(p[j].at>p[j+1].at)
            {
            temp=p[j];
            p[j]=p[j+1];
            p[j+1]=temp;
            }
    max_val=p[0].temp,max_index=0;
                    for(int j=0;j<n&&p[j].at<=p[0].at;j++)
                                    if(p[j].temp>max_val)
                                                    max_val=p[j].temp,max_index=j;
                    i=max_index;
                    c=p[i].ct=p[i].at+1;
                    p[i].rt--;
                    if(p[i].rt==0)
                    {
                                    p[i].temp=MIN;
                                    remaining--;
                    }
                    while(remaining>0)
                    {
                                    max_val=p[0].temp,max_index=0;
                    for(int j=0;j<n&&p[j].at<=c;j++)
                                    if(p[j].temp>max_val)
                                                    max_val=p[j].temp,max_index=j;
                    i=max_index;
                                    p[i].ct=c=c+1;
                                    p[i].rt--;
                                    if(p[i].rt==0)
                                    {
                                        p[i].temp=MIN;
                                        remaining--;
                                    }
                    }
                    printf("\nProcessNo\tAT\tBT\tPri\tCT\tTAT\tWT\n");
    for(int i=0;i<n;i++)
    {
        p[i].tat=p[i].ct-p[i].at;
        avgtat+=p[i].tat;
        p[i].wt=p[i].tat-p[i].bt;
        avgwt+=p[i].wt;
        printf("P%d\t\t%d\t%d\t%d\t%d\t%d\t%d\n",p[i].no,p[i].at,p[i].bt,p[i].pri,p[i].ct,p[i].tat,p[i].wt);
    }
    avgtat/=n,avgwt/=n;
    printf("\nAverage TurnAroundTime=%f\nAverage WaitingTime=%f",avgtat,avgwt);
}
```

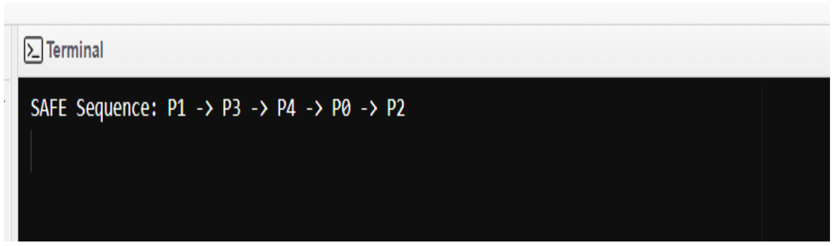
```
#include<stdio.h>
 int main()
{
    int  n;
    printf("Enter Total Number of Processes:");
    scanf("%d", &n);
    int wait_time = 0, ta_time = 0, arr_time[n], burst_time[n], temp_burst_time[n];
    int x = n;
    for(int i = 0; i < n; i++)
    {
        printf("Enter Details of Process %d \n", i + 1);
        printf("Arrival Time:  ");
        scanf("%d", &arr_time[i]);
        printf("Burst Time:   ");
        scanf("%d", &burst_time[i]);
        temp_burst_time[i] = burst_time[i];
    }
    int time_slot;
    printf("Enter Time Slot:");
    scanf("%d", &time_slot);
    int total = 0,  counter = 0,i;
    printf("Process ID       Burst Time       Turnaround Time      Waiting Time\n");
    for(total=0, i = 0; x!=0; )
    {
    if(temp_burst_time[i] <= time_slot && temp_burst_time[i] > 0)
        {
            total = total + temp_burst_time[i];
            temp_burst_time[i] = 0;
            counter=1;
        }
        else if(temp_burst_time[i] > 0)
        {
            temp_burst_time[i] = temp_burst_time[i] - time_slot;
            total  += time_slot;
        }
        if(temp_burst_time[i]==0 && counter==1)
        {
            x--;
            printf("\nProcess No %d  \t\t %d\t\t\t\t %d\t\t\t %d", i+1, burst_time[i],
                   total-arr_time[i], total-arr_time[i]-burst_time[i]);
            wait_time = wait_time+total-arr_time[i]-burst_time[i];
            ta_time += total -arr_time[i];
            counter =0;
        }
        if(i==n-1)
        {
            i=0;
        }
        else if(arr_time[i+1]<=total)
        {
            i++;
        }
        else
        {
            i=0;
        }
    }
    float average_wait_time = wait_time * 1.0 / n;
    float average_turnaround_time = ta_time * 1.0 / n;
    printf("\nAverage Waiting Time:%f", average_wait_time);
    printf("\nAvg Turnaround Time:%f", average_turnaround_time);
    return 0;
}
```



```
Enter Total Number of Processes:3
Enter Details of Process 1
Arrival Time:  2
Burst Time:   4
Enter Details of Process 2
Arrival Time:  4
Burst Time:   7
Enter Details of Process 3
Arrival Time:  7
Burst Time:   11
Enter Time Slot:5
Process ID       Burst Time       Turnaround Time      Waiting Time

Process No 1           4                 2           -2
Process No 2           7                 12          5
Process No 3           11                15          4
Average Waiting Time:2.333333
Avg Turnaround Time:9.666667

=== Code Execution Successful ===
```



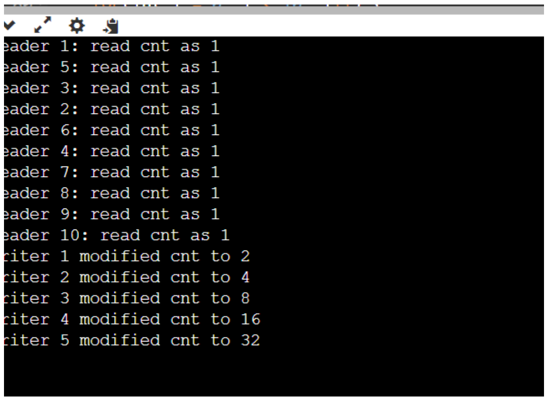
```
<--Highest Priority First Scheduling Algorithm (Preemptive)-->
Enter Number of Processes: 3

Process No: 1
Enter Arrival Time: 1
Enter Burst Time: 4
Enter Priority: 2

Process No: 2
Enter Arrival Time: 5
Enter Burst Time: 7
Enter Priority: 1

Process No: 3
Enter Arrival Time: 6
Enter Burst Time: 8
Enter Priority: 3

ProcessNo   AT  BT  Pri CT  TAT WT
P1          1   4   2   5   4   0
P2          5   7   1   20  15  8
P3          6   8   3   14  8   0

Average TurnAroundTime=9.000000
Average WaitingTime=2.666667
```



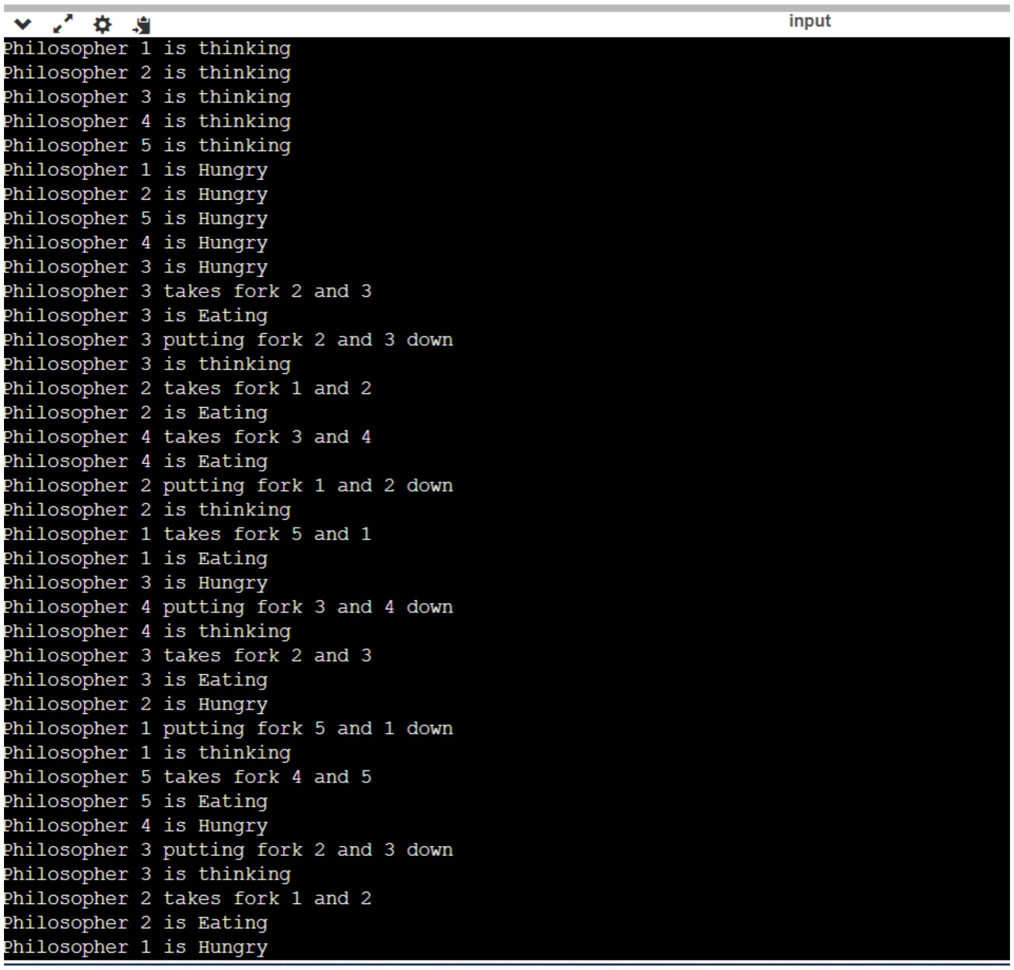
```
SAFE Sequence: P1 -> P3 -> P4 -> P0 -> P2
```

```c
#include <stdio.h>

int main() {
    int numProcesses = 5; // Number of processes
    int numResources = 3; // Number of resources

    int allocationMatrix[5][3] = {{0, 1, 0}, {2, 0, 0}, {3, 0, 2}, {2, 1, 1}, {0, 0, 2}}; // Allocation Matrix
    int maxMatrix[5][3] = {{7, 5, 3}, {3, 2, 2}, {9, 0, 2}, {2, 2, 2}, {4, 3, 3}};   // MAX Matrix
    int availableResources[3] = {3, 3, 2}; // Available Resources

    int isFinished[numProcesses], safeSequence[numProcesses], index = 0;
    for (int k = 0; k < numProcesses; k++) {
        isFinished[k] = 0;
    }

    int needMatrix[numProcesses][numResources];
    for (int i = 0; i < numProcesses; i++) {
        for (int j = 0; j < numResources; j++)
            needMatrix[i][j] = maxMatrix[i][j] - allocationMatrix[i][j];
    }

    for (int k = 0; k < numProcesses; k++) {
        for (int i = 0; i < numProcesses; i++) {
            if (isFinished[i] == 0) {
                int flag = 0;
                for (int j = 0; j < numResources; j++) {
                    if (needMatrix[i][j] > availableResources[j]) {
                        flag = 1;
                        break;
                    }
                }
                if (flag == 0) {
                    safeSequence[index++] = i;
                    for (int y = 0; y < numResources; y++)
                        availableResources[y] += allocationMatrix[i][y];
                    isFinished[i] = 1;
                }
            }
        }
    }

    int flag = 1;
    for (int i = 0; i < numProcesses; i++) {
        if (isFinished[i] == 0) {
            flag = 0;
            printf("The system is not safe.\n");
            break;
        }
    }

    if (flag == 1) {
        printf("SAFE Sequence: ");
        for (int i = 0; i < numProcesses - 1; i++)
            printf("P%d -> ", safeSequence[i]);
        printf("P%d\n", safeSequence[numProcesses - 1]);
    }

    return 0;
}
```

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;
void *writer(void *wno)
{
    sem_wait(&wrt);
    cnt = cnt*2;
    printf("Writer %d modified cnt to %d\n",(*((int *)wno)),cnt);
    sem_post(&wrt);
}
void *reader(void *rno)
{
    pthread_mutex_lock(&mutex);
    numreader++;
    if(numreader == 1) {
        sem_wait(&wrt);
    }
    pthread_mutex_unlock(&mutex);
    printf("Reader %d: read cnt as %d\n",*((int *)rno),cnt);
    pthread_mutex_lock(&mutex);
    numreader--;
    if(numreader == 0) {
        sem_post(&wrt);
    }
    pthread_mutex_unlock(&mutex);
}
int main()
{
    pthread_t read[10],write[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&wrt,0,1);
   int a[10] = {1,2,3,4,5,6,7,8,9,10};
   for(int i = 0; i < 10; i++) {
        pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
    }
    for(int i = 0; i < 10; i++) {
        pthread_join(read[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(write[i], NULL);
    }
    pthread_mutex_destroy(&mutex);
    sem_destroy(&wrt);
    return 0;
    }
```

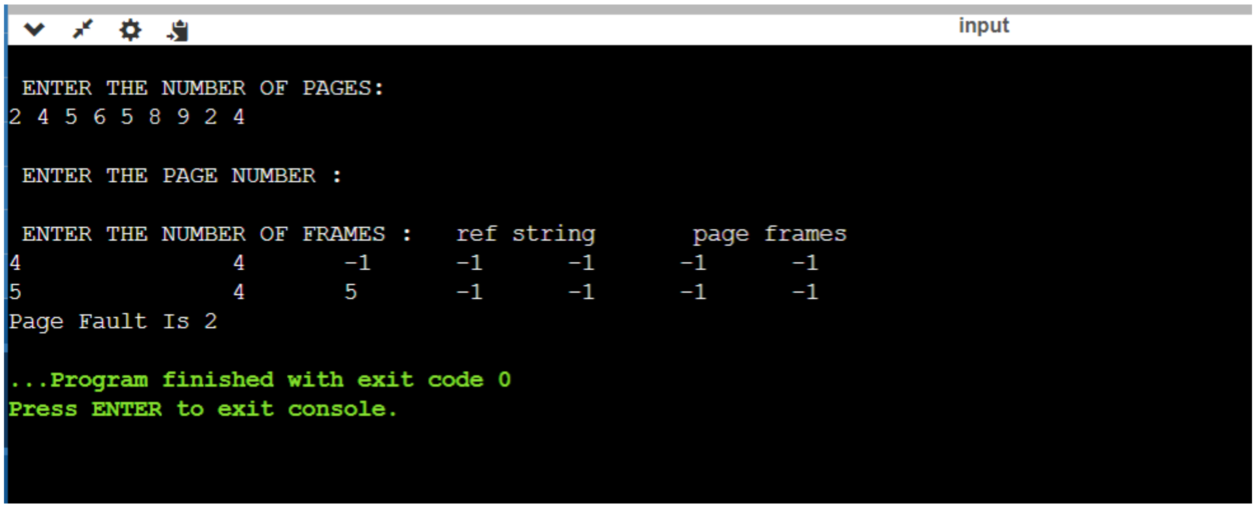


```
eader 1: read cnt as 1
eader 5: read cnt as 1
eader 3: read cnt as 1
eader 2: read cnt as 1
eader 6: read cnt as 1
eader 4: read cnt as 1
eader 7: read cnt as 1
eader 8: read cnt as 1
eader 9: read cnt as 1
eader 10: read cnt as 1
riter 1 modified cnt to 2
riter 2 modified cnt to 4
riter 3 modified cnt to 8
riter 4 modified cnt to 16
riter 5 modified cnt to 32
```



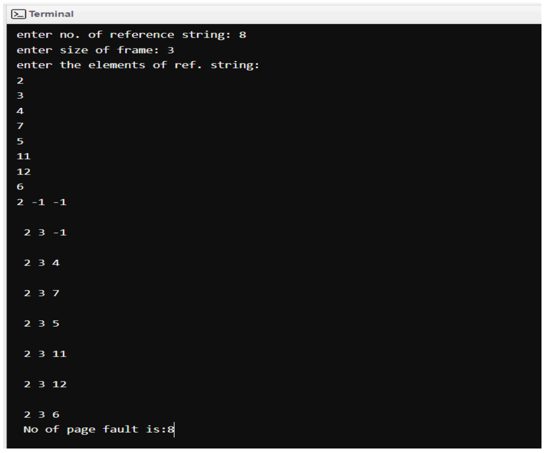
```
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 5 is thinking
Philosopher 1 is Hungry
Philosopher 2 is Hungry
Philosopher 5 is Hungry
Philosopher 4 is Hungry
Philosopher 3 is Hungry
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 4 takes fork 3 and 4
Philosopher 4 is Eating
Philosopher 2 putting fork 1 and 2 down
Philosopher 2 is thinking
Philosopher 1 takes fork 5 and 1
Philosopher 1 is Eating
Philosopher 3 is Hungry
Philosopher 4 putting fork 3 and 4 down
Philosopher 4 is thinking
Philosopher 3 takes fork 2 and 3
Philosopher 3 is Eating
Philosopher 2 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 4 is Hungry
Philosopher 3 putting fork 2 and 3 down
Philosopher 3 is thinking
Philosopher 2 takes fork 1 and 2
Philosopher 2 is Eating
Philosopher 1 is Hungry
```

```
#include<stdio.h>
int main()
{
int i,j,n,a[50],frame[10],no,k,avail,count=0;
          printf("\n ENTER THE NUMBER OF PAGES:\n");
scanf("%d",&n);
          printf("\n ENTER THE PAGE NUMBER :\n");
          for(i=1;i<=n;i++)
          scanf("%d",&a[i]);
          printf("\n ENTER THE NUMBER OF FRAMES :");
          scanf("%d",&no);
for(i=0;i<no;i++)
          frame[i]= -1;
                    j=0;
                    printf("\tref string\t page frames\n");
for(i=1;i<=n;i++)
                    {
                              printf("%d\t\t",a[i]);
                              avail=0;
                              for(k=0;k<no;k++)
if(frame[k]==a[i])
                                        avail=1;
                              if (avail==0)
                              {
                                        frame[j]=a[i];
                                        j=(j+1)%no;
                                        count++;
                                        for(k=0;k<no;k++)
                                        printf("%d\t",frame[k]);
}
                              printf("\n");
}
                    printf("Page Fault Is %d",count);
                    return 0;
}
```

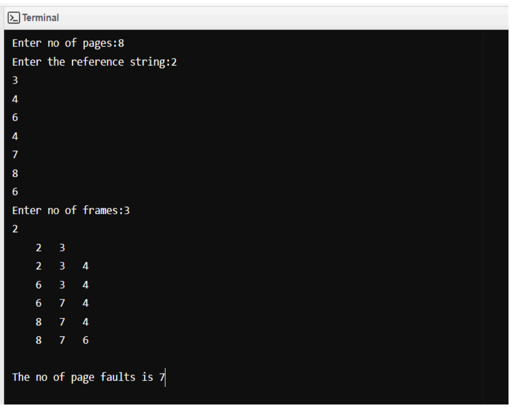


```
 ENTER THE NUMBER OF PAGES:
2 4 5 6 5 8 9 2 4

 ENTER THE PAGE NUMBER :

 ENTER THE NUMBER OF FRAMES :   ref string       page frames
4                4       -1      -1      -1      -1      -1
5                4       5       -1      -1      -1      -1
Page Fault Is 2

...Program finished with exit code 0
Press ENTER to exit console.
```

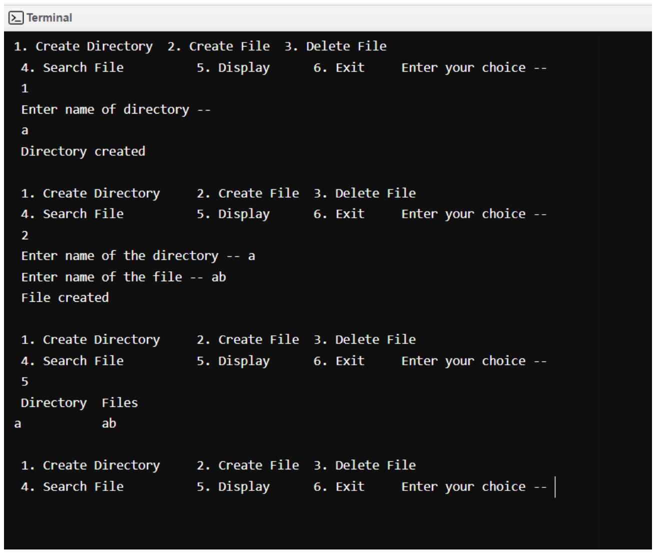
```
#include<stdio.h>
main()
{
    int i,j,k,l,m,n,p,c=0,s;
    int a[20],b[20],q,max;
    printf("enter no. of reference string: ");
    scanf("%d",&n);
    printf("enter size of frame: ");
    scanf("%d",&m);
    printf("enter the elements of ref. string: \n");
    for(i=0; i<n; i++)
        scanf("%d",&a[i]);
    for(j=0; j<m; j++)
        b[j]=-1; //initialize all frame elements with -1
    for(i=0; i<n; i++)
    {
        for(k=0; k<m; k++)
            if(b[k]==a[i])
                goto here;
        for(j=0; j<m; j++)
        {
            if(b[j]==-1)//check if element already present in frame,if true then no page fault.
            {
                b[j]=a[i];
                c++;
                goto here;
            }
        }
        if(j==m)
        {
            l=i+1,max=0;
            for(j=0; j<m; j++)
            {
                for(s=l; s<n; s++)
                {
                    if(a[s]==b[j])
                    {
                        if(s>max)
                        {
                            max=s;
                            p=j;
                        }
                        break;
                    }
                }
                if(s==n)
                {
                    max=s;
                    p=j;
                }
            }
        }
        b[p]=a[i];
        c++;
here:
        printf("\n\n");
        for(k=0; k<m; k++)
            printf(" %d",b[k]);
    }
    printf("\n No of page fault is:%d",c);
    return 0;
}
```



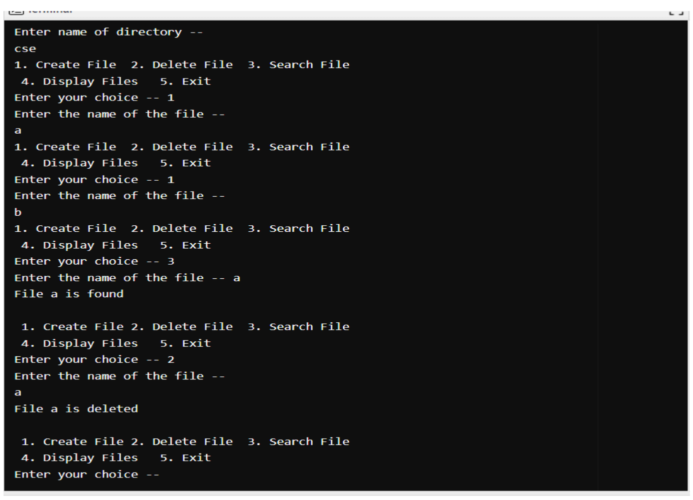
```
enter no. of reference string: 8
enter size of frame: 3
enter the elements of ref. string:
2
3
4
7
5
11
12
6
2 -1 -1

 2 3 -1

 2 3 4

 2 3 7

 2 3 5

 2 3 11

 2 3 12

 2 3 6
 No of page fault is:8
```



```
Enter no of pages:8
Enter the reference string:2
3
4
6
4
7
8
6
Enter no of frames:3
2
    2   3
    2   3   4
    6   3   4
    6   7   4
    8   7   4
    8   7   6


The no of page faults is 7
```



```
1. Create Directory  2. Create File  3. Delete File
 4. Search File          5. Display      6. Exit     Enter your choice --
 1
 Enter name of directory --
 a
 Directory created

 1. Create Directory     2. Create File  3. Delete File
 4. Search File          5. Display      6. Exit     Enter your choice --
 2
 Enter name of the directory -- a
 Enter name of the file -- ab
 File created

 1. Create Directory     2. Create File  3. Delete File
 4. Search File          5. Display      6. Exit     Enter your choice --
 5
 Directory  Files
a           ab

 1. Create Directory     2. Create File  3. Delete File
 4. Search File          5. Display      6. Exit     Enter your choice --
```



```
Enter name of directory --
cse
1. Create File  2. Delete File  3. Search File
 4. Display Files   5. Exit
Enter your choice -- 1
Enter the name of the file --
a
1. Create File  2. Delete File  3. Search File
 4. Display Files   5. Exit
Enter your choice -- 1
Enter the name of the file --
b
1. Create File  2. Delete File  3. Search File
 4. Display Files   5. Exit
Enter your choice -- 3
Enter the name of the file -- a
File a is found

 1. Create File 2. Delete File  3. Search File
 4. Display Files   5. Exit
Enter your choice -- 2
Enter the name of the file --
a
File a is deleted

 1. Create File 2. Delete File  3. Search File
 4. Display Files   5. Exit
Enter your choice --
```

```
#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];
printf("\n\t%d\n",q[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++;
                                                for(j=0;j<k;j++)
                                                printf("\t%d",q[j]);
                                                printf("\n");
                                    }
                                    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("\t%d",q[r]);
                                    }
                                    printf("\n");
                        }
            }
}
printf("\nThe no of page faults is %d",c);
}
```

```
#include<stdio.h>
struct
{
char dname[10],fname[10][10];
int fcnt;
}dir;
void main()
{
int i,ch;
char f[30];
dir.fcnt = 0;
printf("\nEnter name of directory -- ");
scanf("%s", dir.dname);
while(1)
{
printf("\n\n 1. Create File\t2. Delete File\t3. Search File \n 4. Display Files\t5. Exit\nEnter your choice -- ");
scanf("%d",&ch);
switch(ch)
{
case 1: printf("\n Enter the name of the file -- ");
scanf("%s",dir.fname[dir.fcnt]);
dir.fcnt++;
break;
case 2: printf("\n Enter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is deleted ",f);
strcpy(dir.fname[i],dir.fname[dir.fcnt-1]);
break;
}
}
if(i==dir.fcnt)
printf("File %s not found",f);
else
dir.fcnt--;
break;
case 3: printf("\n Enter the name of the file -- ");
scanf("%s",f);
for(i=0;i<dir.fcnt;i++)
{
if(strcmp(f, dir.fname[i])==0)
{
printf("File %s is found ", f);
break;
}
}
if(i==dir.fcnt)
printf("File %s not found",f);
break;
case 4: if(dir.fcnt==0)
printf("\n Directory Empty");
else
{
printf("\n The Files are -- ");
for(i=0;i<dir.fcnt;i++)
printf("\t%s",dir.fname[i]);
}
break;
default: exit(0);
}
}
return 0;
}
```