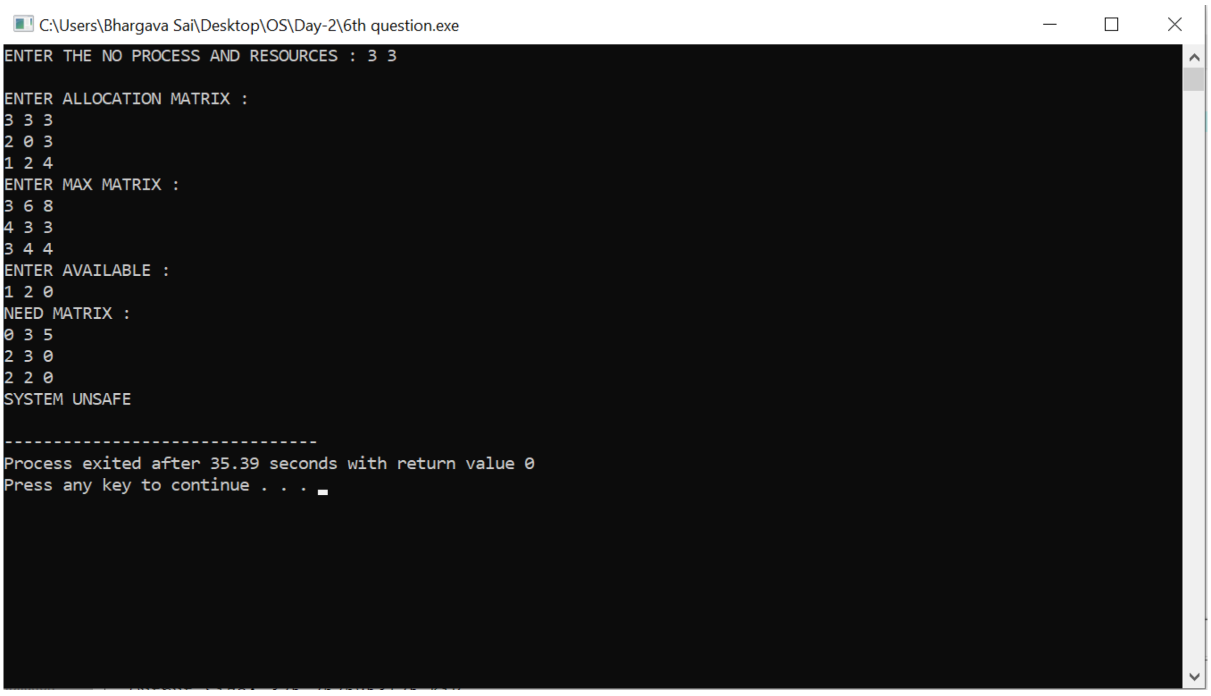
Day-2

```
6.#include<stdio.h>

#include<stdlib.h>

#include<stdbool.h>

int need[100][100],avai[100];

struct process

{

int pc;

bool status;

}

pno[100];

bool check(int i,int m)

{

int j;

for(j=0;j<m;j++)

{

if(need[i][j]>avai[j])

return 0;

}

return 1;

}

void disp(int n)

{

int i;

printf("SYSTEM IN SAFE STATE\nSAFE SEQUENCE = ");

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

{

printf("P%d ",pno[i].pc);

}

}

void banker(int alloc[][100],int max[][100],int n,int m)
```

```
{
int i,j,ck=0,ic=0,flag=0;
bool checker=0;
printf("NEED MATRIX :\n");
for(i=0;i<n;i++)
{
for(j=0;j<m;j++)
{
need[i][j]=max[i][j]-alloc[i][j];
printf("%d ",need[i][j]);
pno[i].status=0;
}
printf("\n");
}
while(ck<n)
{
if(check(ic,m)==1&&pno[ic].status==0)
{
int ac=0;
pno[ck].pc=ic;
ck++;
pno[ic].status=1;
checker=1;
for(ac=0;ac<m;ac++)
{
avai[ac]=avai[ac]+alloc[ic][ac];
}
}
if(ic==n-1&&checker==0)
{
printf("SYSTEM UNSAFE\n");
flag=1;
ck=n;
}
if(ic==n-1)
{
ic=-1;
checker =0;
}
ic++;
}
if(flag==0)
disp(n);
}
int main()
{
int alloc[100][100],max[100][100],n,m,i,j;
printf("ENTER THE NO PROCESS AND RESOURCES : ");
scanf("%d%d",&n,&m);
printf("\nENTER ALLOCATION MATRIX : \n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&alloc[i][j]);
printf("ENTER MAX MATRIX : \n");
for(i=0;i<n;i++)
for(j=0;j<m;j++)
scanf("%d",&max[i][j]);
printf("ENTER AVAILABLE :\n");
for(i=0;i<m;i++)
scanf("%d",&avai[i]);
banker(alloc,max,n,m);
return 0;
```

```
}
```

Output



```
ENTER THE NO PROCESS AND RESOURCES : 3 3

ENTER ALLOCATION MATRIX :
3 3 3
2 0 3
1 2 4
ENTER MAX MATRIX :
3 6 8
4 3 3
3 4 4
ENTER AVAILABLE :
1 2 0
NEED MATRIX :
0 3 5
2 3 0
2 2 0
SYSTEM UNSAFE

--------------------------------
Process exited after 35.39 seconds with return value 0
Press any key to continue . . .
```

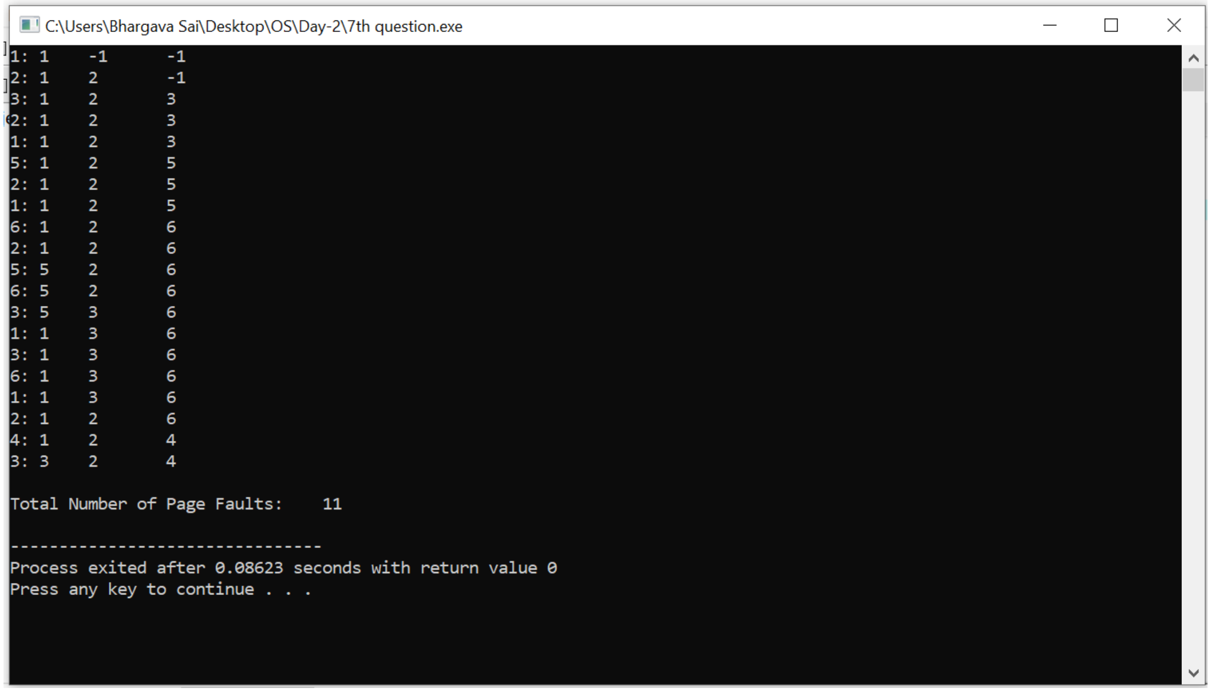
7. #include<stdio.h>

```
int main()
{
   int m, n, position, k, l;
   int a = 0, b = 0, page_fault = 0;

   int total_frames = 3;
   int frames[total_frames];
   int temp[total_frames];
   int pages[] = {1,2,3,2,1,5,2,1,6,2,5,6,3,1,3,6,1,2,4,3};
   int total_pages = sizeof(pages)/sizeof(pages[0]);

   for(m = 0; m < total_frames; m++){
        frames[m] = -1;
   }
```

```
for(n = 0; n < total_pages; n++)
{
  printf("%d: ", pages[n]);
    a = 0, b = 0;
    for(m = 0; m < total_frames; m++)
    {
      if(frames[m] == pages[n])
      {
          a = 1;
          b = 1;
          break;
      }
    }
    if(a == 0)
    {
      for(m = 0; m < total_frames; m++)
      {
        if(frames[m] == -1)
        {
          frames[m] = pages[n];
          b = 1;
          page_fault++;
          break;
        }
      }
    }
    if(b == 0)
    {
      for(m = 0; m < total_frames; m++)
      {
        temp[m] = 0;
```

```
            }
            for(k = n - 1, l = 1; l <= total_frames - 1; l++, k--)
            {
               for(m = 0; m < total_frames; m++)
               {
                  if(frames[m] == pages[k])
                  {
                     temp[m] = 1;
                  }
               }
            }
            for(m = 0; m < total_frames; m++)
            {
               if(temp[m] == 0)
                  position = m;
            }
            frames[position] = pages[n];
            page_fault++;
         }

         for(m = 0; m < total_frames; m++)
         {
            printf("%d\t", frames[m]);
         }
         printf("\n");
   }
   printf("\nTotal Number of Page Faults:\t%d\n", page_fault);

   return 0;
}
```

Output

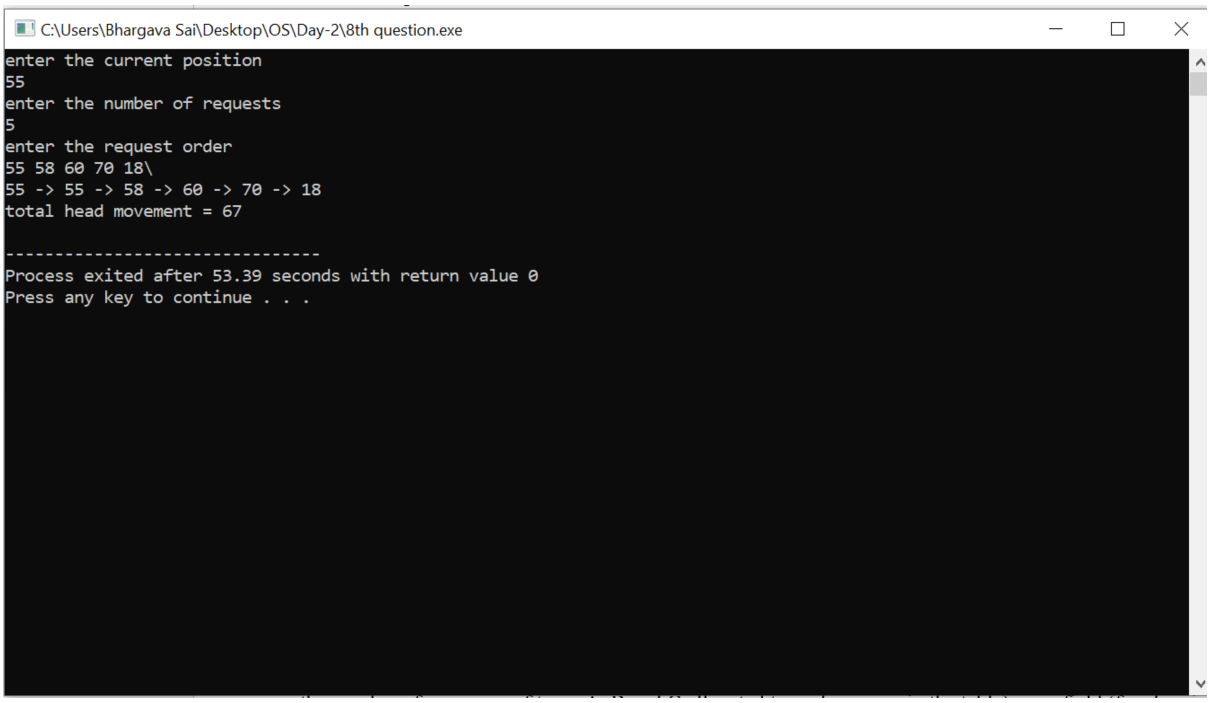

C:\Users\Bhargava Sai\Desktop\OS\Day-2\7th question.exe

```
1: 1    -1      -1
2: 1    2       -1
3: 1    2       3
2: 1    2       3
1: 1    2       3
5: 1    2       5
2: 1    2       5
1: 1    2       5
6: 1    2       6
2: 1    2       6
5: 5    2       6
6: 5    2       6
3: 5    3       6
1: 1    3       6
3: 1    3       6
6: 1    3       6
1: 1    3       6
2: 1    2       6
4: 1    2       4
3: 3    2       4

Total Number of Page Faults:    11

--------------------------------
Process exited after 0.08623 seconds with return value 0
Press any key to continue . . .
```

8. #include<math.h>

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
  int i,n,req[50],mov=0,cp;
  printf("enter the current position\n");
  scanf("%d",&cp);
  printf("enter the number of requests\n");
  scanf("%d",&n);
  printf("enter the request order\n");
  for(i=0;i<n;i++)
  {
    scanf("%d",&req[i]);
  }
  mov=mov+abs(cp-req[0]); // abs is used to calculate the absolute value
  printf("%d -> %d",cp,req[0]);
  for(i=1;i<n;i++)
  {
        mov=mov+abs(req[i]-req[i-1]);
        printf(" -> %d",req[i]);
    }
    printf("\n");
    printf("total head movement = %d\n",mov);
}
```

Output



C:\Users\Bhargava Sai\Desktop\OS\Day-2\8th question.exe

```
enter the current position
55
enter the number of requests
5
enter the request order
55 58 60 70 18\
55 -> 55 -> 58 -> 60 -> 70 -> 18
total head movement = 67

--------------------------------
Process exited after 53.39 seconds with return value 0
Press any key to continue . . .
```

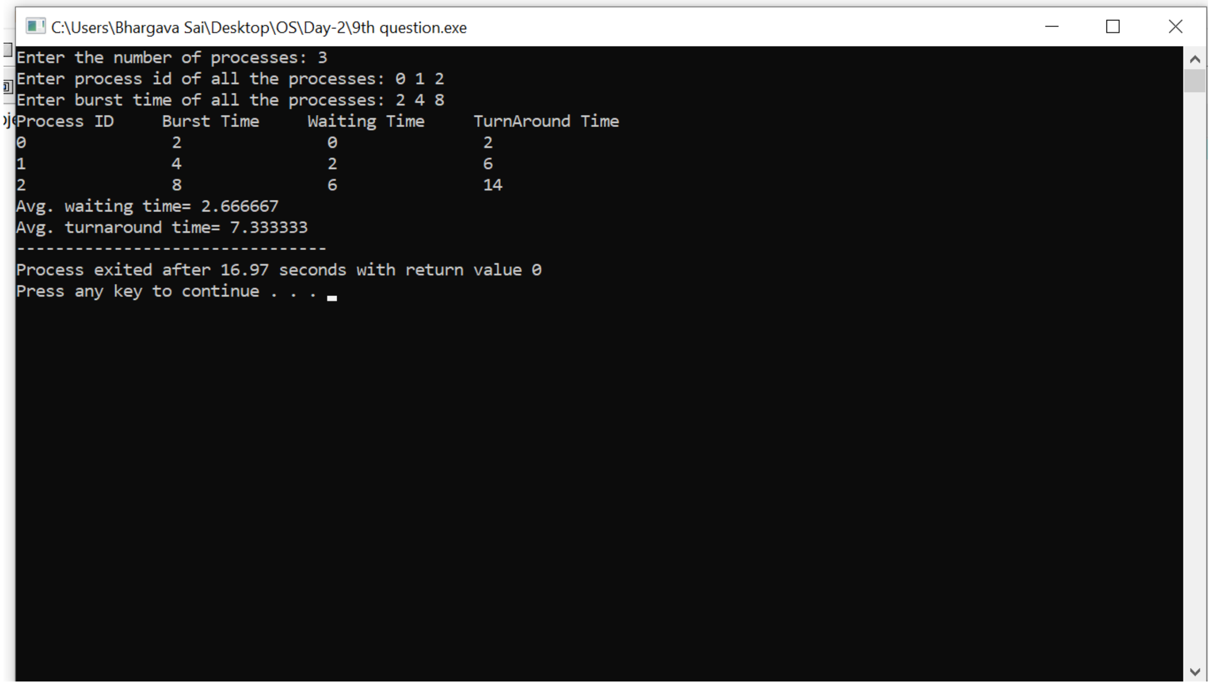
9. #include <stdio.h>

```
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 calculating waiting time of each process
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]);

    //calculating and printing turnaround time of each process
    printf("%d\t\t", bt[i]+wt[i]);
    printf("\n");
```

```
        //for calculating total waiting time
        twt += wt[i];

        //for calculating total turnaround time
        tat += (wt[i]+bt[i]);
    }
    float att,awt;

    //for calculating average waiting time
    awt = twt/n;

    //for calculating average turnaround time
    att = tat/n;
    printf("Avg. waiting time= %f\n",awt);
    printf("Avg. turnaround time= %f",att);
}
```

Output



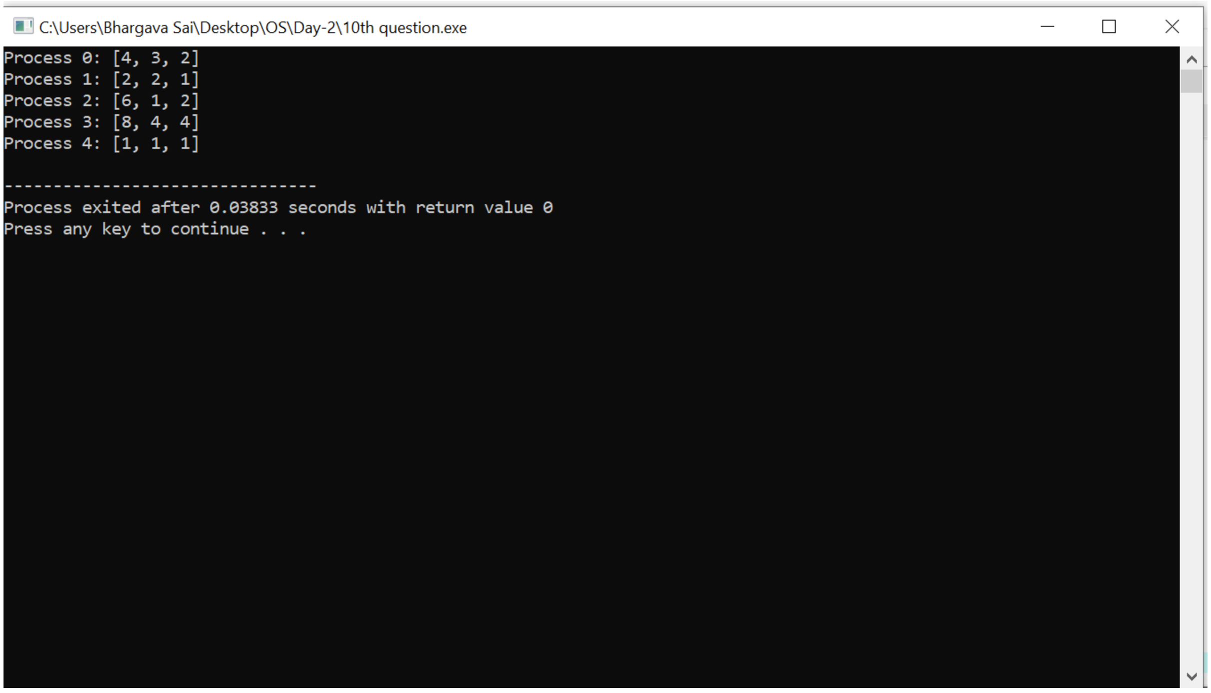
C:\Users\Bhargava Sai\Desktop\OS\Day-2\9th question.exe

```
Enter the number of processes: 3
Enter process id of all the processes: 0 1 2
Enter burst time of all the processes: 2 4 8
Process ID     Burst Time     Waiting Time     TurnAround Time
0               2              0                2
1               4              2                6
2               8              6                14
Avg. waiting time= 2.666667
Avg. turnaround time= 7.333333
--------------------------------
Process exited after 16.97 seconds with return value 0
Press any key to continue . . .
```

```
10. #include<stdio.h>
int main()
```

```
{
        int allocation[5][3]={
        {1,1,2},
        {2,1,2},
        {3,0,1},
        {0,2,0},
        {1,1,2}};
        int max[5][3]={
        {5,4,4},
        {4,3,3},
        {9,1,3},
        {8,6,4},
        {2,2,3}};
        int available[3]={3,2,1};
        int need[5][3]={0};
        for(int i=0;i<5;i++)
        {
                for(int j=0;j<3;j++)
                {
                        need[i][j]=max[i][j]-allocation[i][j];
                }
        }
        for(int i=0;i<5;i++)
        {
                printf("Process %d: [%d, %d, %d]\n",i,need[i][0],need[i][1],need[i][2]);
        }
        return 0;
}
```

Output



```
Process 0: [4, 3, 2]
Process 1: [2, 2, 1]
Process 2: [6, 1, 2]
Process 3: [8, 4, 4]
Process 4: [1, 1, 1]

--------------------------------
Process exited after 0.03833 seconds with return value 0
Press any key to continue . . .
```