Program-10

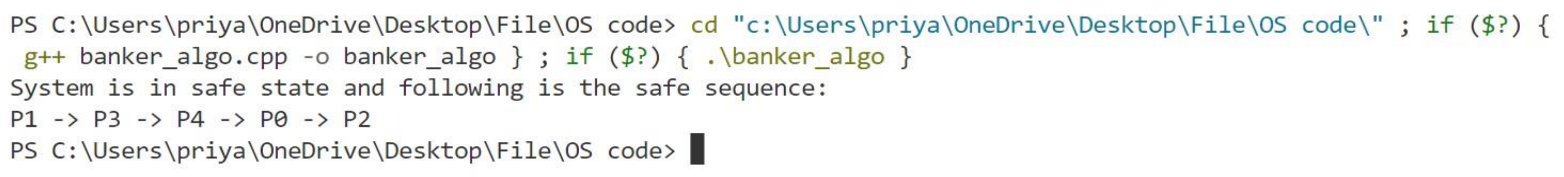
Q.Write a program for Banker's algorithm.

Source code:-

```
#include<stdio.h> int
main() {
   int i, j, k;
   int n = 5; //number of processes
int m = 3; //number of resources
   int allocation[5][3] = {{0, 1, 0},
                    {2, 0, 0},
                    {3, 0, 2},
                    {2, 1, 1},
                    {0, 0, 2}};

   int max[5][3] = {{7, 5, 3},
                  {3, 2, 2},
                  {9, 0, 2},
                  {2, 2, 2},
                  {4, 3, 3}};

   int available[3] = {3, 3, 2};

   int finish[n] = {0};
int ans[n] = {0};
   int idx =  0;

   int need[n][m];
   for(int i=0;i<n;i++)
   {
      for(int j=0;j<m; j++)
      {
         need[i][j] = max[i][j] - allocation[i][j];
      }
   }

   int y = 0;
   for (int k=0;k<5;k++)
   {
      for(int i=0;i<n;i++)
      {
         if(finish[i]==0)
         {
int flag = 0;
            for(int j=0;j<m;j++)
```

```
                {
                   if(need[i][j]>available[j])
                     {flag = 1;
                   }   }
                 if(flag==0)   {
                   ans[idx++] = i;
                   for(int y=0;y<m;y++)
                   {
                      available[y] += allocation[i][y];
                   }
                   finish[i] = 1;
                }   }   }
    }

    bool flag = true;
    for(int i=0; i<n; i++)
    {
       if(finish[i] == 0)
       {  flag = false;
          printf("System is in deadlock !!"); break;
       }
    }
    if(flag==true)
    {
       cout<<"System is in safe state and following is the safe sequence: "<<endl;
 for(int i=0;i<n-1;i++)
       {
          printf("%dP",ans[i]);
       }
       printf("%dP",ans[n-1]);
    }
 return 0;
 }
```

Output:-



```
PS C:\Users\priya\OneDrive\Desktop\File\OS code> cd "c:\Users\priya\OneDrive\Desktop\File\OS code\" ; if ($?) {
 g++ banker_algo.cpp -o banker_algo } ; if ($?) { .\banker_algo }
System is in safe state and following is the safe sequence:
P1 -> P3 -> P4 -> P0 -> P2
PS C:\Users\priya\OneDrive\Desktop\File\OS code>
```

Program-11

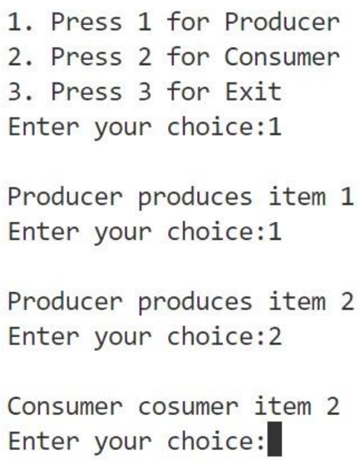
Q.Write a C program for Producer and consumer problem

Source Code:-

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

int mutex=1,full=0,empty=10,x=0;
void producer()
   {
       --mutex;
       ++full;          --
empty;
       x++;
       printf("\nProducer produces item %d",x);
       ++mutex;
   }
   void consumer()
   {
       --mutex;
       --full;
       ++empty;
       printf("\nConsumer cosumer item %d",x);
x--;
       ++mutex;
   }
int main() {
   int n,i;
   printf("\n1. Press 1 for Producer"
"\n2. Press 2 for Consumer"            "\n3.
Press 3 for Exit");
   for (i = 1; i > 0; i++) {

       printf("\nEnter your choice:");
scanf("%d", &n);


       switch (n) {
case 1:

           if    ((mutex    ==    1)
&&    (empty    !=    0))    {
producer();
```

```
            }
        else {
        }
        break;

            case 2:


            if ((mutex == 1)
        && (full != 0)) {
                consumer();
            }

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

Output:-



```
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:2

Consumer cosumer item 2
Enter your choice:
```

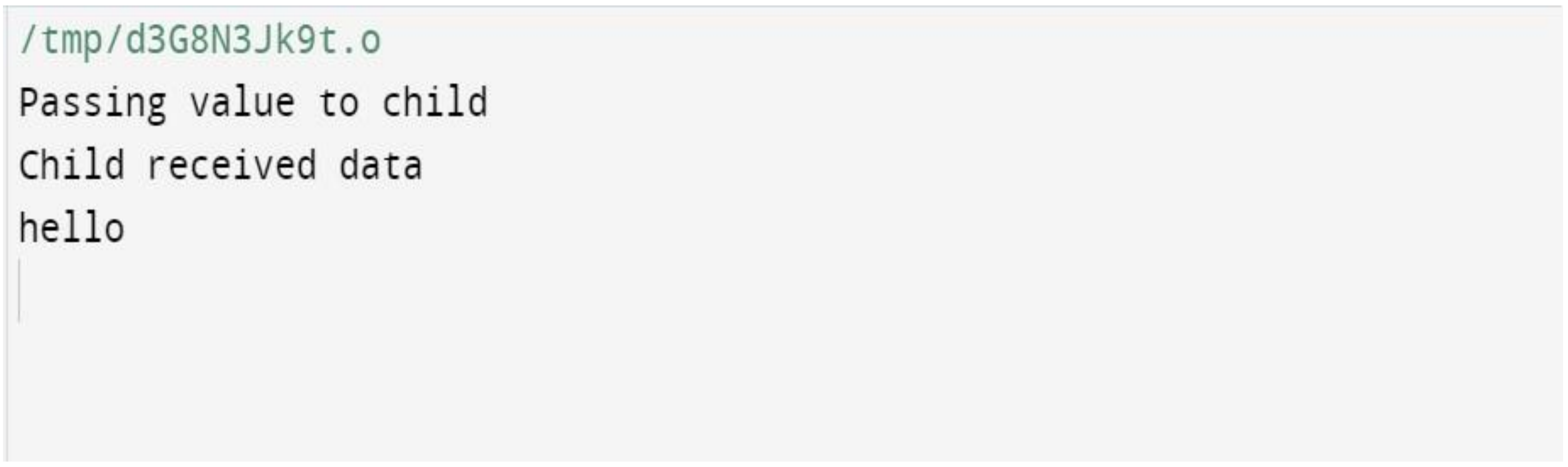
Program-12

Q.Write a C program for inter process communication using pipe function.

Source Code:-

```
#include<unistd.h>
#include <stdio.h> #include<sys/types.h>
int main()
{
   int fd[2],n;
char buffer[100];
   pid_t p;
pipe(fd);
p=fork();
if(p>0)
   {
      printf("Passing value to child\n");
write(fd[1],"hello\n",6);
   }
else
   {
      printf("Child received data\n");
n=read(fd[0],buffer,100);
      write(1,buffer,n);
   }
}
```

Output:-



```
/tmp/d3G8N3Jk9t.o
Passing value to child
Child received data
hello
```

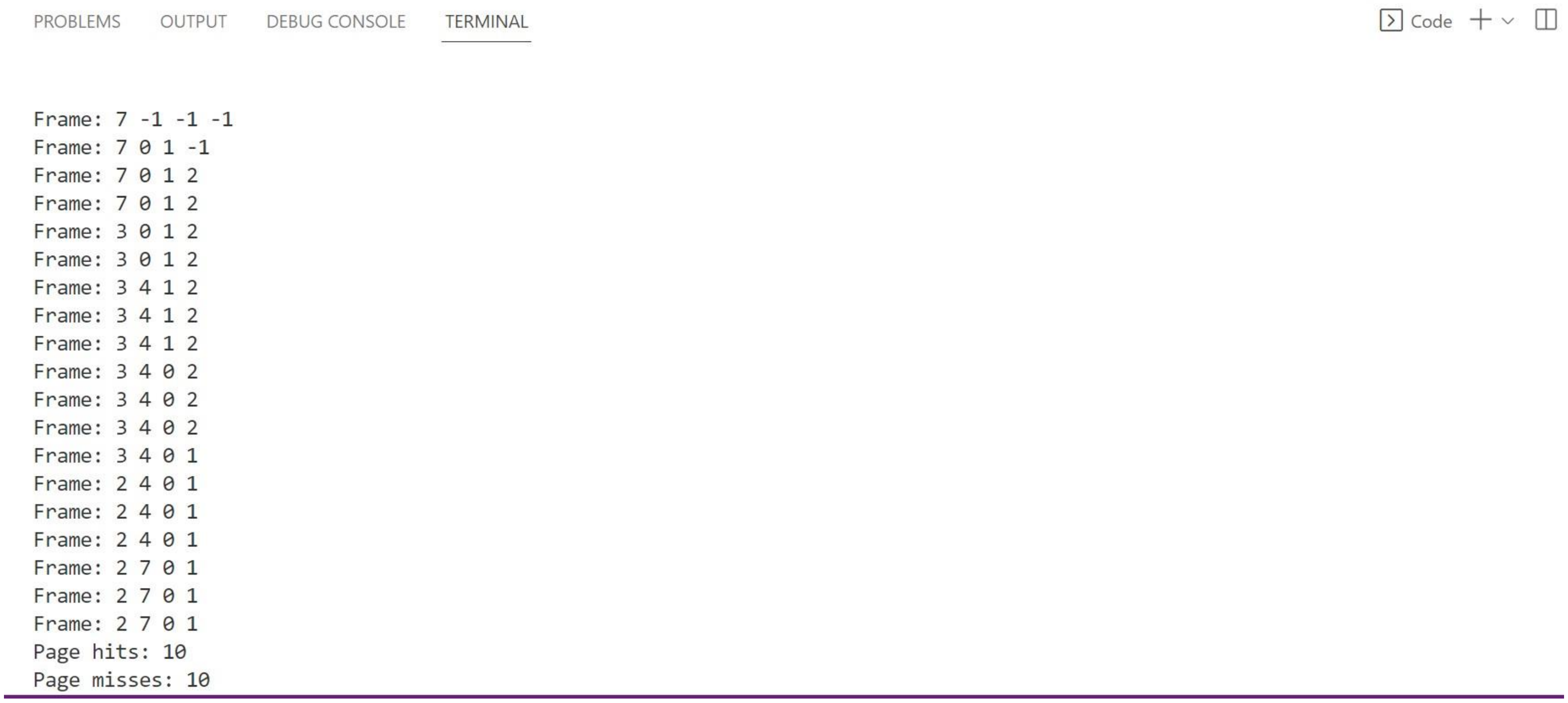
Program-13

Q.C program to implement FCFS page replacement policy.

Source Code:-

```
#include<stdio.h>
void fifo(int string[20],int n,int size)
{    int frames[n];    for
(int i=0;i<n;i++)
        frames[i]=-1;

    int index=-1;
int page_miss=0;
int page_hits=0;
    for (int i=0;i<size;i++)
    {
        int symbol=string[i];
int flag=0;

        for(int j=0;j<n;j++)
        {
            if (symbol==frames[j])
            {
flag=1;
break;
            }
        }

        if (flag==1)
        {
            printf("\nFrame: ",symbol);
for(intj=0;j<n;j++)
printf("%d",frames[j]);
page_hits+=1;
        }else {
   index=(index+1)%n;
frames[index]=symbol;
page_miss+=1;
 printf("\nFrame: ",symbol);
for (int j=0;j<n;j++)
printf("%d ",frames[j]);
        }
    }
    printf("\nPage hits: %d",page_hits);
printf("\nPage misses: %d",page_miss);
}



int main(void)
{
    int string[]={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1};
int no_frames=4;     int size=sizeof(string)/sizeof(int);
    fifo(string,no_frames,size);

}
```

Output:-



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Frame: 7 -1 -1 -1
Frame: 7 0 1 -1
Frame: 7 0 1 2
Frame: 7 0 1 2
Frame: 3 0 1 2
Frame: 3 0 1 2
Frame: 3 4 1 2
Frame: 3 4 1 2
Frame: 3 4 1 2
Frame: 3 4 0 2
Frame: 3 4 0 2
Frame: 3 4 0 2
Frame: 3 4 0 1
Frame: 2 4 0 1
Frame: 2 4 0 1
Frame: 2 4 0 1
Frame: 2 7 0 1
Frame: 2 7 0 1
Frame: 2 7 0 1
Page hits: 10
Page misses: 10
```

Program-14

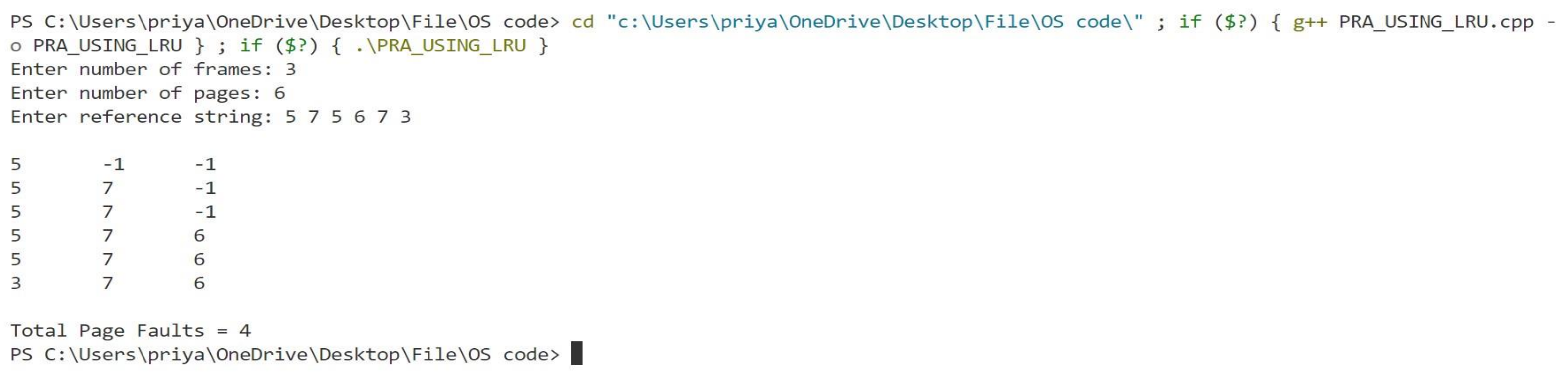
Q.C program to implement LRU page replacement policy.

Source Code:-

```
#include<stdio.h>  int
findLRU(int time[], int n){
    int i, minimum = time[0], pos = 0;

    for(i = 1; i < n; ++i){
if(time[i] < minimum){
            minimum = time[i];
            pos = i;
        }
}
    return pos;
}
int main()
{
    int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j,
            pos, faults = 0;
 printf("Enter number of frames: ");
scanf("%d", &no_of_frames);
printf("Enter number of pages: ");
scanf("%d", &no_of_pages);
printf("Enter 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]){
                counter++;
time[j] = counter;
flag1 = flag2 = 1;
break;
             } }
        if(flag1 == 0){
        for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
        counter++;
```

```
          faults++;
          frames[j] = pages[i];
          time[j] = counter;
                  flag2 = 1;
                 break;
               }
            }
          }
          if(flag2 == 0){
             pos = findLRU(time, no_of_frames);
    counter++;              faults++;
    frames[pos] = pages[i];
    time[pos] = counter;
          }
    printf("\n");
          for(j = 0; j < no_of_frames; ++j){
             printf("%d\t", frames[j]);
          }
       }
       printf("\n\nTotal Page Faults = %d", faults);
    return 0;
    }
```

Output:-



```
PS C:\Users\priya\OneDrive\Desktop\File\OS code> cd "c:\Users\priya\OneDrive\Desktop\File\OS code\" ; if ($?) { g++ PRA_USING_LRU.cpp -o PRA_USING_LRU } ; if ($?) { .\PRA_USING_LRU }
Enter number of frames: 3
Enter number of pages: 6
Enter reference string: 5 7 5 6 7 3

5       -1      -1
5       7       -1
5       7       -1
5       7       6
5       7       6
3       7       6

Total Page Faults = 4
PS C:\Users\priya\OneDrive\Desktop\File\OS code>
```

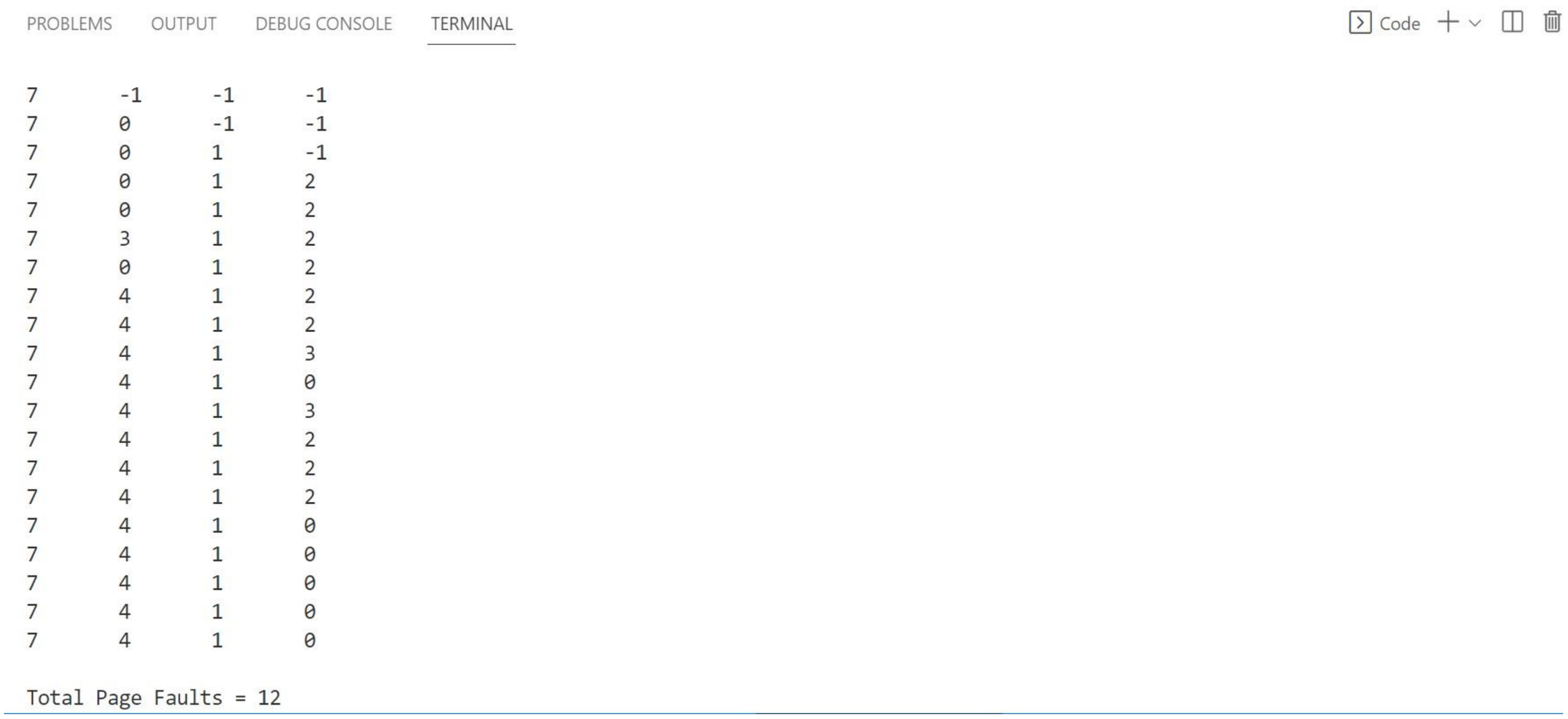
Program-15

Q.C program to implement MRU page replacement policy.

Source Code:-

```
#include<stdio.h>  int
findLRU(int time[], int n){
   int i, maximum = time[0], pos = 0;

   for(i = 1; i < n; ++i){
if(time[i] > maximum){
         maximum = time[i];
pos = i;
      }
}
   return pos;
} int
main()
{
   int no_of_frames, no_of_pages, frames[10], pages[30], counter = 0, time[10], flag1, flag2, i, j, pos,
faults = 0;     printf("Enter number of frames: ");     scanf("%d", &no_of_frames);     printf("Enter
number of pages: ");     scanf("%d", &no_of_pages);     printf("Enter 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]){
            counter++;
time[j] = counter;
flag1 = flag2 = 1;
break;
            } } if(flag1 == 0){     for(j = 0;
      j < no_of_frames; ++j){
            if(frames[j] == -1){
      counter++;           faults++;
      frames[j] = pages[i];
      time[j] = counter;
              flag2 = 1;
             break;
            }
            }
        }
        if(flag2 == 0){
            pos = findLRU(time, no_of_frames);
counter++;              faults++;
frames[pos] = pages[i];             time[pos] =
counter;
        }
printf("\n");
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }
    printf("\n\nTotal Page Faults = %d", faults);
return 0;
}
```

Output:-



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

Total Page Faults = 12
```

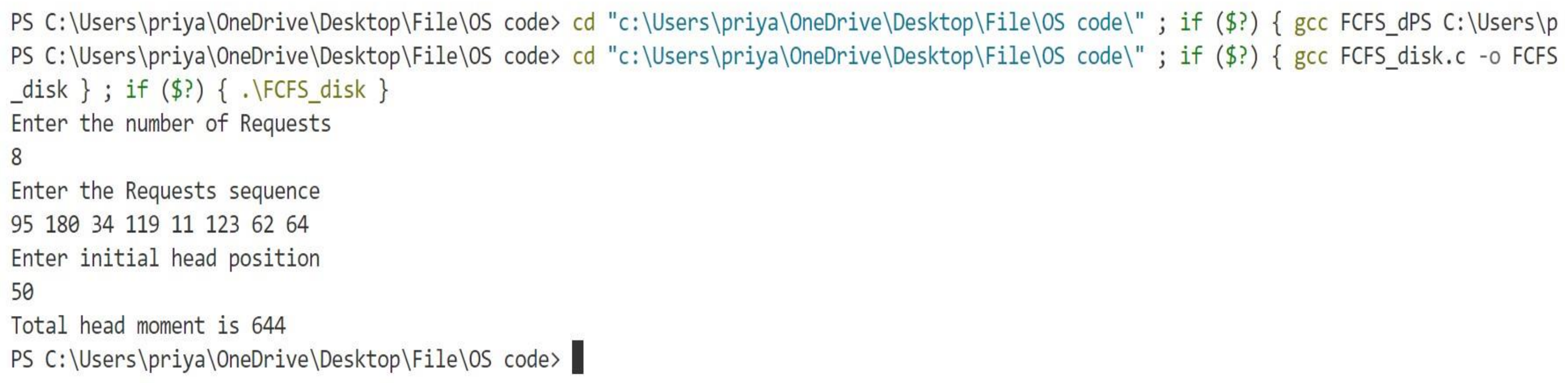
Program-16

Q.C program to implement FCFS disk scheduling algorithm.

Source Code:-

```
#include<stdio.h>
#include<stdlib.h> int
main()
{
Int    RQ[100],i,n,TotalHeadMoment=0,initial;
printf("Enter   the   number   of   Requests\n");
scanf("%d",&n);
   printf("Enter the Requests sequence\n");
for(i=0;i<n;i++)     scanf("%d",&RQ[i]);
   printf("Enter initial head position\n");
scanf("%d",&initial);
for(i=0;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
    }
   printf("Total head moment is %d",TotalHeadMoment);
return 0;
}
```

Output:-



```
PS C:\Users\priya\OneDrive\Desktop\File\OS code> cd "c:\Users\priya\OneDrive\Desktop\File\OS code\" ; if ($?) { gcc FCFS_dPS C:\Users\p
PS C:\Users\priya\OneDrive\Desktop\File\OS code> cd "c:\Users\priya\OneDrive\Desktop\File\OS code\" ; if ($?) { gcc FCFS_disk.c -o FCFS
_disk } ; if ($?) { .\FCFS_disk }
Enter the number of Requests
8
Enter the Requests sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Total head moment is 644
PS C:\Users\priya\OneDrive\Desktop\File\OS code> █
```

Program-17

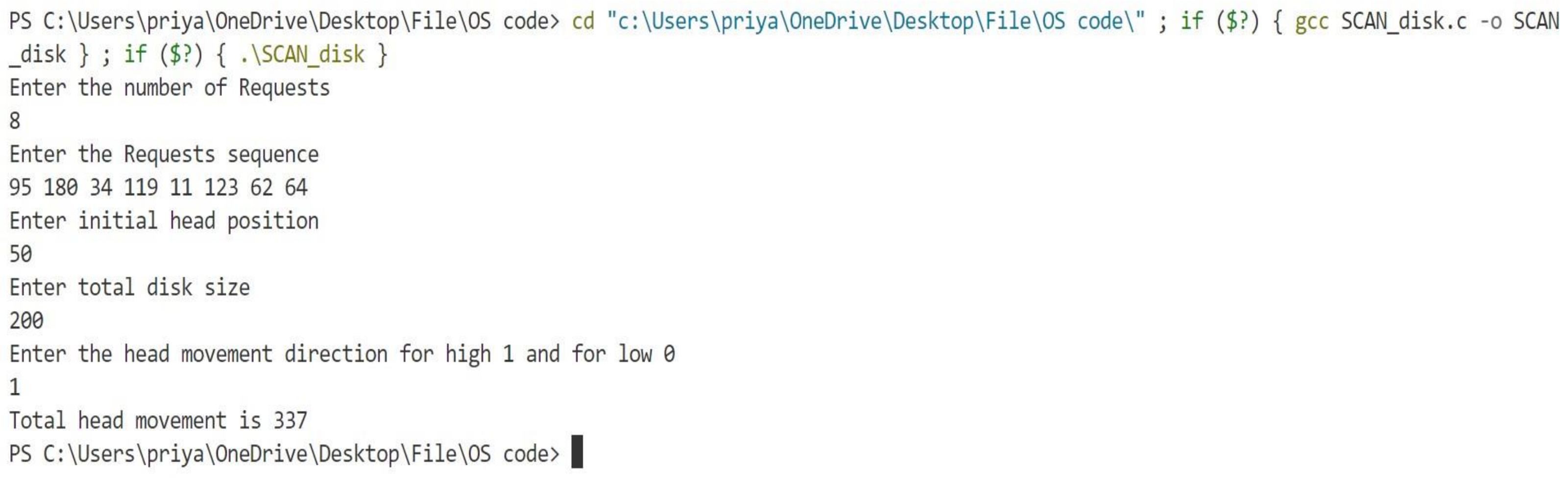
Q.C program to implement SCAN disk scheduling algorithm.

Source Code:-

```
#include<stdio.h>
#include<stdlib.h>
int main() {
   int RQ[100],i,j,n,TotalHeadMoment=0,initial,size,move;
printf("Enter        the        number        of        Requests\n");
scanf("%d",&n);
   printf("Enter the Requests sequence\n");
for(i=0;i<n;i++)     scanf("%d",&RQ[i]);
printf("Enter initial head position\n");
scanf("%d",&initial);     printf("Enter total
disk size\n");     scanf("%d",&size);
   printf("Enter the head movement direction for high 1 and for low 0\n");
scanf("%d",&move);     for(i=0;i<n;i++)
   {
      for(j=0;j<n-i-1;j++)
      {
         if(RQ[j]>RQ[j+1])
         {               int
temp;
temp=RQ[j];
RQ[j]=RQ[j+1];
            RQ[j+1]=temp;
         }

      }     }
int index;
   for(i=0;i<n;i++)
   {
      if(initial<RQ[i])
      {
index=i;
break;
      }
   }
   if(move==1)
   {
      for(i=index;i<n;i++)
      {
         TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];        }
```

```
        TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
initial = size-1;         for(i=index-1;i>=0;i--)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];

        }
}
else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];
        }
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
initial =0;
        for(i=index;i<n;i++)
        {
             TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
initial=RQ[i];

        }
    }
    printf("Total head movement is %d",TotalHeadMoment);
return 0;
}
```

Output:-



```
PS C:\Users\priya\OneDrive\Desktop\File\OS code> cd "c:\Users\priya\OneDrive\Desktop\File\OS code\" ; if ($?) { gcc SCAN_disk.c -o SCAN_disk } ; if ($?) { .\SCAN_disk }
Enter the number of Requests
8
Enter the Requests sequence
95 180 34 119 11 123 62 64
Enter initial head position
50
Enter total disk size
200
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 337
PS C:\Users\priya\OneDrive\Desktop\File\OS code>
```