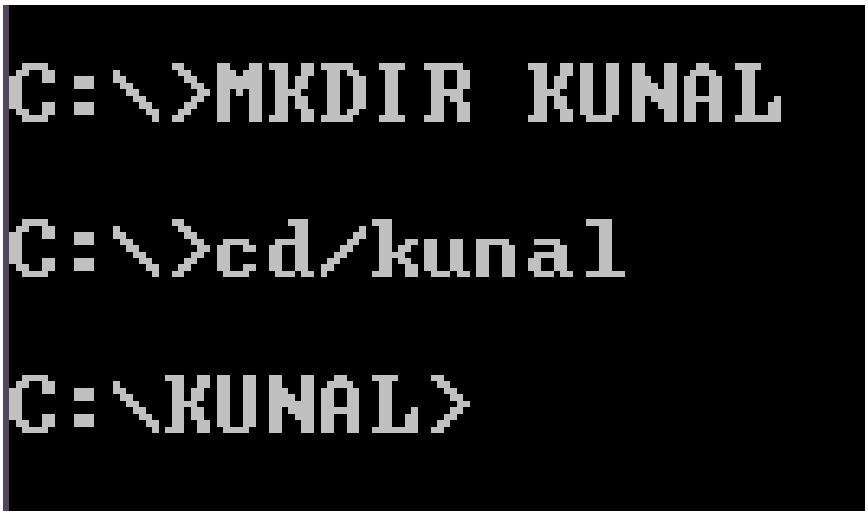
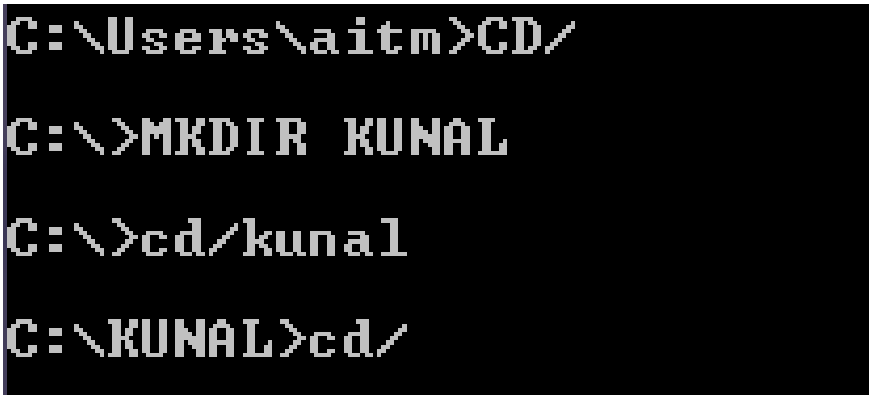
INDEX

1.WAP To Implement DOS Command:-

Program:

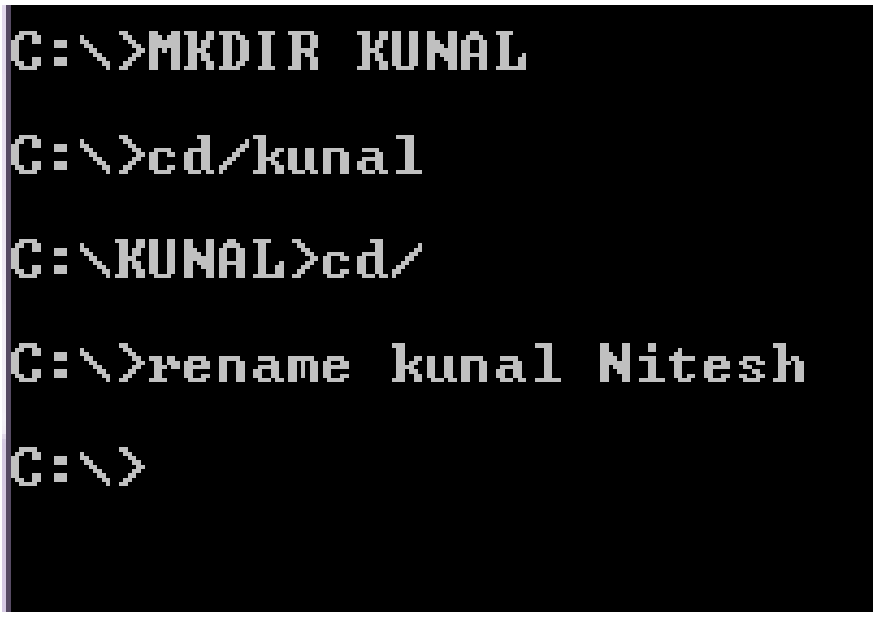
- **CD (To Change Directory)**
- **MKDIR - (To Create a New Directory)**
- **DEL (To Delete File)**
- **DIR (To Open List Of Files And Folder)**
- **RENAME (To rename the file)**





```
C:\>MKDIR KUNAL

C:\>cd/kunal

C:\KUNAL>cd/

C:\>rename kunal Nitesh

C:\>del Nitesh
C:\Nitesh\*, Are you sure (Y/N)? y

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is F05F-BBB5
```

```
C:\>MKDIR KUNAL

C:\>cd/kunal

C:\KUNAL>cd/

C:\>rename kunal Nitesh

C:\>del Nitesh
C:\Nitesh\*, Are you sure (Y/N)? y

C:\>dir
 Volume in drive C has no label.
 Volume Serial Number is F05F-BBB5

 Directory of C:\

11/11/2022  02:14 PM    <DIR>          .joggi
11/11/2022  02:24 PM    <DIR>          .manishjogi
11/11/2022  02:07 PM    <DIR>          .mannu
11/11/2022  02:03 PM    <DIR>          .rupesh
10/18/2022  10:32 AM    <DIR>          abc
12/01/2021  03:38 PM    <DIR>          aitm
04/11/2022  09:48 AM    <DIR>          aman
04/11/2022  09:56 AM    <DIR>          aman2
06/11/2009  03:12 AM                24 autoexec.bat
10/18/2022  11:11 AM    <DIR>          cde
12/01/2021  03:21 PM    <DIR>          class
12/01/2021  03:28 PM    <DIR>          computer
06/11/2009  03:12 AM                10 config.sys
04/11/2022  09:47 AM    <DIR>          huihui
12/01/2021  03:10 PM    <DIR>          india
11/11/2022  02:07 PM    <DIR>          manish
10/07/2022  02:18 PM    <DIR>          mohit
12/13/2022  11:31 AM    <DIR>          Nitesh
12/01/2021  03:30 PM    <DIR>          palwal
07/14/2009  08:07 AM    <DIR>          PerfLogs
12/01/2021  02:56 PM    <DIR>          pooja
07/11/2022  10:28 AM    <DIR>          Program Files
12/01/2021  03:03 PM    <DIR>          shalu
12/13/2022  10:26 AM    <DIR>          TurboC++
10/29/2021  12:23 PM    <DIR>          Users
04/27/2022  11:26 AM    <DIR>          Windows
               2 File(s)             34 bytes
              24 Dir(s)  32,687,947,776 bytes free

C:\>
```


C:\>MKDIR KUNAL
C:\>cd/kunal
C:\KUNAL>cd/
C:\>rename kunal Nitesh
C:\>

2.WAP To Implement FCFS Scheduling :-

Program:

```
#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);
return 0;
}
```

Output:-

```
Enter the number of processes: 3
Enter process id of all the processes: 1 2 3
Enter burst time of all the processes: 5 11 11
Process ID     Burst Time     Waiting Time     TurnAround Time
1              5              0                5
2              11             5                16
3              11             16               27
Avg. waiting time= 7.000000
Avg. turnaround time= 16.000000
```

3.WAP To Implement Of SJF Scheduling:-

Program:

```
#include<stdio.h>
int main() {
  int time, burst_time[10], at[10], sum_burst_time = 0, smallest, n, i;
  int sumt = 0, sumw = 0;
  printf("enter the no of processes : ");
  scanf("%d", & n);
  for (i = 0; i < n; i++) {
    printf("the arrival time for process P%d : ", i + 1);
    scanf("%d", & at[i]);
    printf("the burst time for process P%d : ", i + 1);
    scanf("%d", & burst_time[i]);
    sum_burst_time += burst_time[i];
  }
  burst_time[9] = 9999;
  for (time = 0; time < sum_burst_time;) {
    smallest = 9;
    for (i = 0; i < n; i++) {
      if (at[i] <= time && burst_time[i] > 0 && burst_time[i] <
burst_time[smallest])
        smallest = i;
    }
    printf("P[%d]\t|\t%d\t|\t%d\n", smallest + 1, time + burst_time[smallest] -
at[smallest], time - at[smallest]);
    sumt += time + burst_time[smallest] - at[smallest];
    sumw += time - at[smallest];
    time += burst_time[smallest];
    burst_time[smallest] = 0;
  }
  printf("\n\n average waiting time = %f", sumw * 1.0 / n);
  printf("\n\n average turnaround time = %f", sumt * 1.0 / n);
  return 0;
}
```

Output:-

```
enter the no of processes: 2
the arrival time for process P1: 10
the burst time for process P1: 5
the arrival time for process P2: 6

the burst time for process P2 : 3
P[10] |         -22765         |         -32764
 the average waiting time = -16382.000000
 the average turnaround time = -11382.500000
```

4.WAP To Implement Of Priority Scheduling Algorithm In C:-

Program:

```
#include<stdio.h>
 // structure representing a structure
struct priority_scheduling
{
// name of the process
  char process_name;
// time required for execution
  int burst_time;
 // waiting time of a process
 int waiting_time;
// total time of execution
  int turn_around_time;
// priority of the process
  int priority;
};
int main()
 {
// total number of processes
  int  number_of_process;
// total waiting and turnaround time
  int total = 0;
// temporary structure for swapping
  struct priority_scheduling temp_process;
// ASCII numbers are used to represent the name of the process
  int ASCII_number = 65;
// swapping position
  int position;
 // average waiting time of the process
  float average_waiting_time;
// average turnaround time of the process
  float average_turnaround_time;
 printf("Enter the total number of Processes: ");
  // get the total number of the process as input
  scanf("%d", & number_of_process);
// initializing the structure array
  struct priority_scheduling process[number_of_process];
```

```
 printf("\nPlease Enter the  Burst Time and Priority of each process:\n");
 // get burst time and priority of all process
  for (int i = 0; i < number_of_process; i++)
 {
// assign names consecutively using ASCII number
process[i].process_name = (char) ASCII_number;
printf("\nEnter the details of the process %c \n", process[i].process_name);
    printf("Enter the burst time: ");
    scanf("%d", & process[i].burst_time);
    printf("Enter the priority: ");
    scanf("%d", & process[i].priority);
// increment the ASCII number to get the next alphabet
    ASCII_number++;
}
// swap process according to high priority
  for (int i = 0; i < number_of_process; i++) {
position = I;
 for (int j = i + 1; j < number_of_process; j++)
{
// check if priority is higher for swapping
      if (process[j].priority > process[position].priority)
        position = j;
    }
    // swapping of lower priority process with the higher priority process
    temp_process = process[i];
    process[i] = process[position];
    process[position] = temp_process;
  }
  // First process will not have to wait and hence has a waiting time of 0
 process[0].waiting_time = 0;
  for (int i = 1; i < number_of_process; i++)
 {
    process[i].waiting_time = 0;
    for (int j = 0; j < i; j++)
 {
      // calculate waiting time
      process[i].waiting_time += process[j].burst_time;
    }
// calculate total waiting time
    total += process[i].waiting_time;
```

```
  }
  // calculate average waiting time
 average_waiting_time = (float) total / (float) number_of_process;
// assigning total as 0 for next calculations
  total = 0;
  printf("\n\nProcess_name \t Burst Time \t Waiting Time \t Turnaround
Time\n");
  printf("_____\n");

  for (int i = 0; i < number_of_process; i++)
 {
// calculating the turn around time of the processes
    process[i].turn_around_time = process[i].burst_time +
process[i].waiting_time;
    // calculating the total turnaround time.
    total += process[i].turn_around_time;
// printing all the values
    printf("\t %c \t\t %d \t\t %d \t\t %d", process[i].process_name,
process[i].burst_time, process[i].waiting_time,
process[i].turn_around_time);
    printf("\n_____\n");
  }
  // calculating the average turn_around time
  average_turnaround_time = (float) total / (float) number_of_process;
  // average waiting time
  printf("\n\n Average Waiting Time : %f", average_waiting_time);
 // average turnaround time
  printf("\n Average Turnaround Time: %f\n", average_turnaround_time);
 return 0;
}
```

Output:-

```
Enter the total number of Processes :3
Please Enter the Burst Time and Priority of each process:
Enter the details of the process A
Enter the burst time: 5
Enter the priority: 2
Enter the details of the process B
Enter the burst time: 6
Enter the priority: 1

Enter the details of the process C
Enter the burst time: 7
Enter the priority: 3
Process_name        Burst Time   Waiting Time        Turnaround Time
-------------------------------------------------------
   C            7              0              7
-------------------------------------------------------
   A            5              7              12
-------------------------------------------------------
   B            6              12             18
-------------------------------------------------------
Average Waiting Time: 6.333333
Average Turnaround Time: 12.333333
```

5.WAP To Implement Of Round Robin Scheduling :-

Program:

```
#include<stdio.h>
int main()
{
 int cnt,j,n,t,remain,flag=0,tq;
  int wt=0,tat=0,at[10],bt[10],rt[10];
  printf("Enter Total Process:\t ");
  scanf("%d",&n);
  remain=n;
  for(cnt=0;cnt<n;cnt++)
  {
    printf("Enter Arrival Time and Burst Time for Process Process
Number %d :",cnt+1);
    scanf("%d",&at[cnt]);
    scanf("%d",&bt[cnt]);
    rt[cnt]=bt[cnt];
  }
  printf("Enter Time Quantum:\t");
  scanf("%d",&tq);
  printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
  for(t=0,cnt=0;remain!=0;)
  {
    if(rt[cnt]<=tq && rt[cnt]>0)
    {
      t+=rt[cnt];
      rt[cnt]=0;
      flag=1;
    }
    else if(rt[cnt]>0)
    {
      rt[cnt]-=tq;
      t+=tq;
    }
    if(rt[cnt]==0 && flag==1)
    {
      remain--;
      printf("P[%d]\t|\t%d\t|\t%d\n",cnt+1,t-at[cnt],t-at[cnt]-bt[cnt]);
      wt+=t-at[cnt]-bt[cnt];
```

```
      tat+=t-at[cnt];
      flag=0;
    }
    if(cnt==n-1)
      cnt=0;
    else if(at[cnt+1]<=t)
      cnt++;
    else
      cnt=0;
  }
  printf("\nAverage Waiting Time= %f\n",wt*1.0/n);
  printf("Avg Turnaround Time = %f",tat*1.0/n);

  return 0;
}
```

Output:-

```
Enter Total Process:    4
Enter Arrival Time and Burst Time for Process Process Number 1 :0 5
Enter Arrival Time and Burst Time for Process Process Number 2 :1 4
Enter Arrival Time and Burst Time for Process Process Number 3 :2 2
Enter Arrival Time and Burst Time for Process Process Number 4 :4 1
Enter Time Quantum:  2
Process |Turnaround Time|Waiting Time

P[3]   |      4       |      2
P[4]   |      3       |      2
P[2]   |      10      |      6
P[1]   |      12      |      7
Average Waiting Time= 4.250000
Avg Turnaround Time = 7.250000
```

6.WAP to implement BanKer's algorithm in C:-

Program:

```
#include <stdio.h>
int main()
{
        // P0, P1, P2, P3, P4 are the Process names here
        int n, m, i, j, k;
        n = 5; // Number of processes
        m = 3; // Number of resources
        int alloc[5][3] = { { 0, 1, 0 }, // P0 // Allocation Matrix
                                        { 2, 0, 0 }, // P1
                                        { 3, 0, 2 }, // P2
                                        { 2, 1, 1 }, // P3
                                        { 0, 0, 2 } }; // P4
        int max[5][3] = { { 7, 5, 3 }, // P0 // MAX Matrix
                                    { 3, 2, 2 }, // P1
                                    { 9, 0, 2 }, // P2
                                    { 2, 2, 2 }, // P3
                                    { 4, 3, 3 } }; // P4
        int avail[3] = { 3, 3, 2 }; // Available Resources
       int f[n], ans[n], ind = 0;
        for (k = 0; k < n; k++) {
                f[k] = 0;
        }
        int need[n][m];
        for (i = 0; i < n; i++) {
                for (j = 0; j < m; j++)
                        need[i][j] = max[i][j] - alloc[i][j];
        }
        int y = 0;
        for (k = 0; k < 5; k++) {
                for (i = 0; i < n; i++) {
                        if (f[i] == 0) {

                                int flag = 0;
                                for (j = 0; j < m; j++) {
                                        if (need[i][j] > avail[j]){
                                                flag = 1;
                                                break;
```

```
                                        }
                                }

                                if (flag == 0) {
                                        ans[ind++] = i;
                                        for (y = 0; y < m; y++)
                                                avail[y] += alloc[i][y];
                                        f[i] = 1;
                                }
                        }
                }
        }
        int flag = 1;
        for(int i=0;i<n;i++)
        {
        if(f[i]==0)
        {
                flag=0;
                printf("The following system is not safe");
                break;
        }
        }
        if(flag==1)
        {
        printf("Following is the SAFE Sequence\n");
        for (i = 0; i < n - 1; i++)
       printf(" P%d ->", ans[i]);
        printf(" P%d", ans[n - 1]);
        }
return (0);
}
```

Output:-

Following is the SAFE Sequence

P1 -> P3 -> P4 -> P0 -> P2

7.WAP To Implement FCFS Page Replacement Algorithm In C:-

Program:

```
#include<  stdio.h>
#include< conio.h>
int fsize;
int frm[15];
void display();
void main()
{
 int pg[100],nPage,i,j,pf=0,top=-1,temp,flag=0;
 clrscr();
 printf("\n Enter frame size:");
 scanf("%d",&fsize);
 printf("\n Enter number of pages:");
 scanf("%d",&nPage);
for(i=0;i< nPage;i++)
 {
 printf("\n Enter page[%d]:",i+1);
  scanf("%d",&pg[i]);
 }
   for(i=0;i< fsize;i++)
    frm[i]=-1;
printf("\n page | \t Frame content ");
printf("\n_____");
for(j=0;j< nPage;j++)
{
 flag=0;
  for(i=0;i< fsize;i++)
  {
   if(frm[i]==pg[j])
   {
    flag=1;
    break;
   }
  }
 if(flag==0)
  {
```

```
   if(top==fsize-1)
   {
    top=-1;
   }
   pf++;
   top++;
   frm[top]=pg[j];
  }
 printf("\n %d    |",pg[j]);
 display();
}
  printf("\n_____");
 printf("\n total page fault:%d",pf);
getch();
}
void display()
{
 int i;
 for(i=0;i< fsize;i++)
  printf("\t %d",frm[i]);
}
```

OUTPUT:-

```
Enter frame size:3
 Enter number of pages:12
Enter page[1]:1
 Enter page[2]:2
 Enter page[3]:3
 Enter page[4]:4
 Enter page[5]:1
 Enter page[6]:2
 Enter page[7]:5
 Enter page[8]:1
 Enter page[9]:2
 Enter page[10]:3
 Enter page[11]:4
 Enter page[12]:5
 page |        Frame content
-------------------------
 1   | 1      -1     -1

 2   | 1      2      -1
 3   | 1      2      3
 4   | 4      2      3
 1   | 4      1      3
 2   | 4      1      2
 5   | 5      1      2
 1   | 5      1      2
 2   | 5      1      2
 3   | 5      3      2
 4   | 5      3      4
 5   | 5      3      4
-------------------------
 total page fault: 9
```

8.WAP To Implement Optimal Page Replacement Algorithm In C:-

Program:

```
#include<stdio.h>
#include<conio.h>
main()
{
   int fr[5],i,j,k,t[5],p=1,flag=0,page[25],psz,nf,t1,u[5];
   clrscr();
   printf("enter the number of frames:");
   scanf("%d",&nf);
   printf("\n enter the page size");
   scanf("%d",&psz);

   printf("\nenter the page sequence:");
   for(i=1; i<=psz; i++)
      scanf("%d",&page[i]);

   for(i=1; i<=nf; i++)
      fr[i]=-1;
   for(i=1; i<=psz; i++)
   {
      if(full(fr,nf)==1)
        break;
      else
      {
         flag=0;
         for(j=1; j<=nf; j++)
         {
           if(page[i]==fr[j])
            {
               flag=1;
               printf("         \t%d:\t",page[i]);
               break;
            }
         }
         if(flag==0)
         {
            fr[p]=page[i];
```

```
            printf("          \t%d:\t",page[i]);
            p++;
        }

        for(j=1; j<=nf; j++)
            printf(" %d  ",fr[j]);
        printf("\n");
    }
}
p=0;
for(; i<=psz; i++)
{
    flag=0;
    for(j=1; j<=nf; j++)
    {
        if(page[i]==fr[j])
        {
            flag=1;
            break;
        }
    }
   if(flag==0)
    {
        p++;
        for(j=1; j<=nf; j++)
        {
            for(k=i+1; k<=psz; k++)
            {
                if(fr[j]==page[k])
                {
                    u[j]=k;
                    break;
                }
                else
                    u[j]=21;
            }
        }
        for(j=1; j<=nf; j++)
            t[j]=u[j];
        for(j=1; j<=nf; j++)
```

```
            {
               for(k=j+1; k<=nf; k++)
               {
                  if(t[j]<t[k])
                  {
                     t1=t[j];
                     t[j]=t[k];
                     t[k]=t1;
                  }
               }
            }
            for(j=1; j<=nf; j++)
            {
               if(t[1]==u[j])
               {
                  fr[j]=page[i];
                  u[j]=i;
               }
            }
            printf("page fault\t");
         }
         else
            printf("          \t");
         printf("%d:\t",page[i]);
         for(j=1; j<=nf; j++)
            printf(" %d ",fr[j]);
         printf("\n");
      }
      printf("\ntotal page faults:  %d",p+3);
   // getch();
   }
   int full(int a[],int n)
   {
      int k;
      for(k=1; k<=n; k++)
      {
         if(a[k]==-1)
            return 0;
      }
      return 1;  }
```

Output:-

```
enter the number of frames:5
enter the page size2
enter the page sequence:1
2
  1:      1   -1   -1   -1   -1
  2:      1   2   -1   -1   -1
total page faults:  3
```

9.WAP To Implement SSTF Disk Scheduling Algorithm In C:-

Program:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
   int RQ[100],i,n,TotalHeadMoment=0,initial,count=0;
   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);
  // logic for sstf disk scheduling
 /* loop will execute until all process is completed*/
   while(count!=n)
   {
      int min=1000,d,index;
      for(i=0;i<n;i++)
      {
        d=abs(RQ[i]-initial);
        if(min>d)
        {
           Min = d;
          index = i;
        }
   }
      Total Head Moment=Total Head Moment +min;
      Initial = RQ [index];
      // 1000 is for max
      // you can use any number
      RQ [index] =  1000;
      Count ++
  }
Printf (" Total head movement is %d", Total Head Moment);
   return 0;
}
```

Output:-

```
Enter the number of Request
8
Enter Request Sequence
95 180 34 119 11 123 62 64
Enter initial head Position
50
Total head movement is 236
```

10. WAP To Implement Scan Disk Scheduling Algorithm In C:-

Program:

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
int main()
{
   int i,j,k,n,m,sum=0,x,y,h;
   cout<<"Enter the size of disc \n";
   cin>>m;
   cout<<"Enter number of requests \n";
   cin>>n;
   cout<<"Enter the requests \n";
   vector<int>a(n),b;
   for(i=0;i<n;i++)
   {
      cin>>a[i];
   }
   for(i=0;i<n;i++)
   {
      if(a[i]>m)
      {
         cout<<"Error, unknown position"<<a[i]<<"\n";
         return 0;
      }
   }

   cout<<"Enter the head position \n";
   cin>>h;
   int temp=h;
   a.push_back(h);
   a.push_back(m);
   a.push_back(0);
   sort(a.begin(),a.end());
   for(i=0;i<a.size();i++)
```

```
    {
       if(h==a[i])
       break;
    }
    k=i;
    if(k<n/2)
    {
       for(i=k;i<a.size();i++)
       {
          b.push_back(a[i]);
       }
    }
    else
    {
       for(i=k;i>=0;i--)
       {
          b.push_back(a[i]);
       }
       for(i=k+1;i<a.size();i++)
       {
          b.push_back(a[i]);
       }
    }
    temp=b[0];
    cout<<temp;
    for(i=1;i<b.size();i++)
    {
       cout<<" -> "<<b[i];
       sum+=abs(b[i]-temp);
       temp=b[i];
    }
    cout<<"\n";
    cout<<"Total head movement = "<<sum<<"\n";
    cout<<"Average head movement = "<<(float)sum/n<<"\n";
    return 0;
}
```

Output:

```
Enter the size of disc
300
Enter number of requests
9
Enter the requests
43 65 22 43 21 11 76 88 10
Enter the head position
40
40 -> 22 -> 21 -> 11 -> 10 -> 0 -> 43 -> 43 -> 65 -> 76 -> 88 -> 300
Total head movement = 340
Average head movement = 37.7778
```