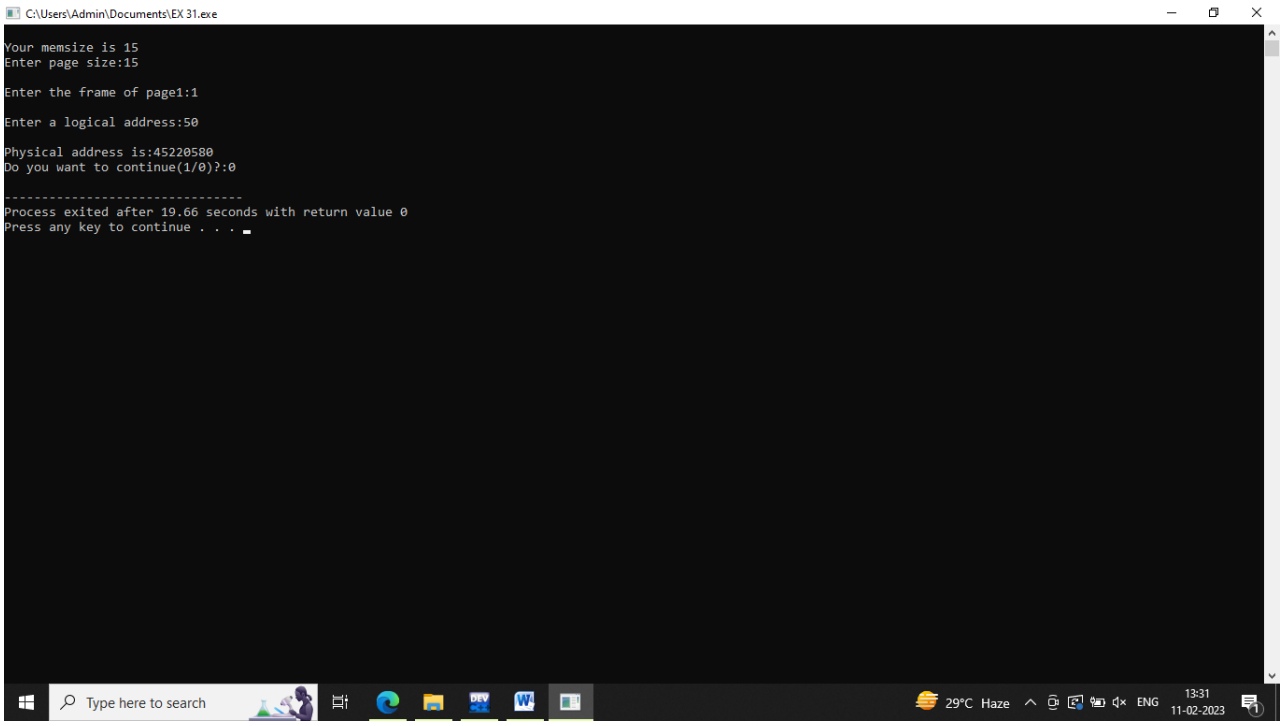
31. Construct a C program to simulate the First in First Out paging technique of memory management.

```
#include<stdio.h>

int main()

{

int memsize=15;

int pagesize,nofpage;

int p[100];

int frameno,offset;

int logadd,phyadd;

int i;

int choice=0;

printf("\nYour memsize is %d ",memsize);

printf("\nEnter page size:");

scanf("%d",&pagesize);


nofpage=memsize/pagesize;


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

{

printf("\nEnter the frame of page%d:",i+1);

scanf("%d",&p[i]);

}


do

{

printf("\nEnter a logical address:");

scanf("%d",&logadd);
```

```
frameno=logadd/pagesize;

offset=logadd%pagesize;

phyadd=(p[frameno]*pagesize)+offset;

printf("\nPhysical address is:%d",phyadd);

printf("\nDo you want to continue(1/0)?:");

scanf("%d",&choice);

}while(choice==1);

}
```



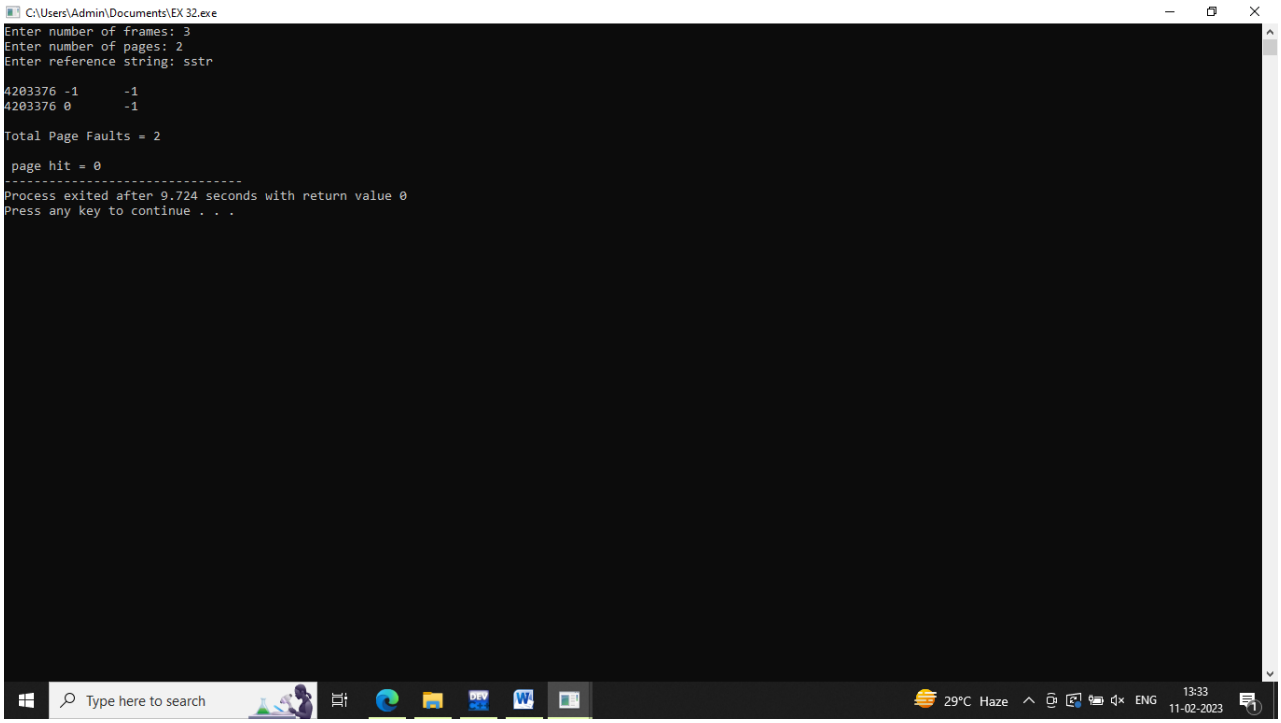
32. Construct a C program to simulate the Least Recently Used paging technique of memory management.

```
#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,page_hit;
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);
page_hit=no_of_pages-faults;
```

```
printf("\n\n page hit = %d",page_hit);

   return 0;

}
```



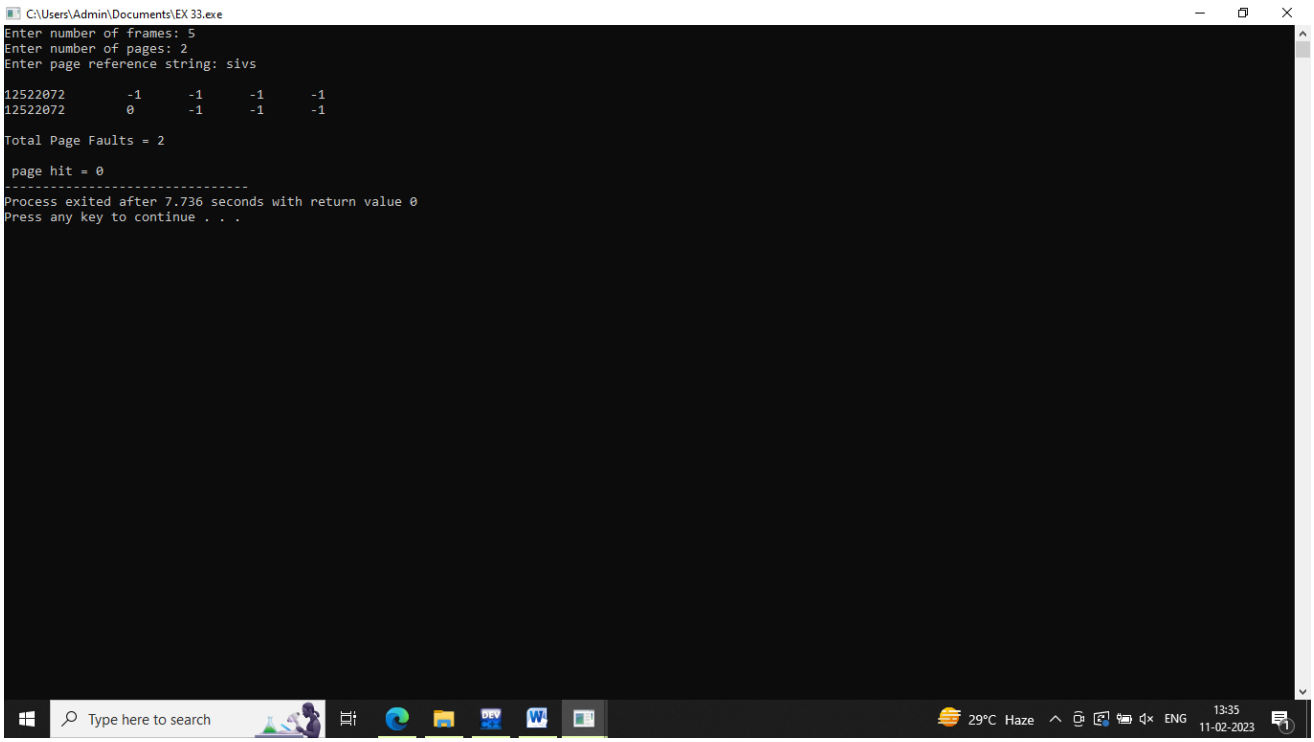
33. Construct a C program to simulate the optimal paging technique of memory management

```
#include<stdio.h>

int main()

{

   int no_of_frames, no_of_pages,page_hit, 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;
 break;
 }
 }
}
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;
 }
 }
}
```

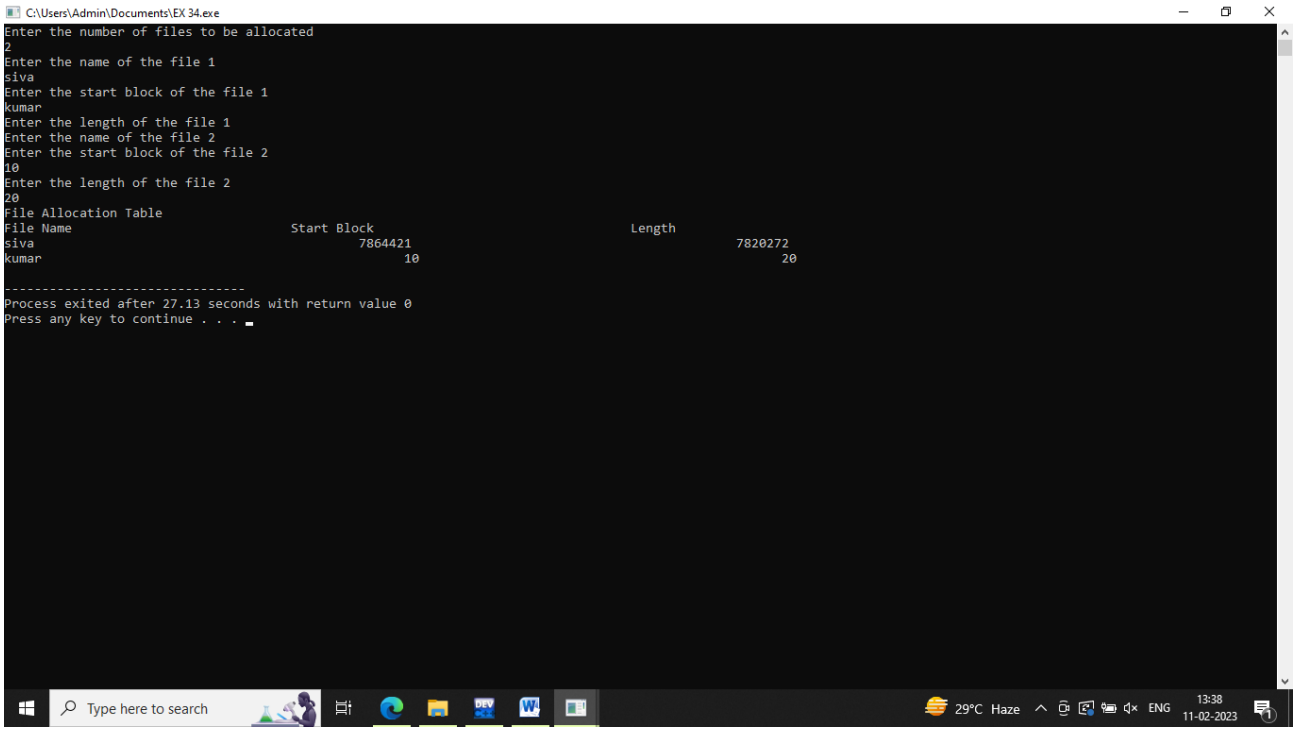
```
frames[pos] = pages[i];
faults++;
        }
        printf("\n");
        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }
    printf("\n\nTotal Page Faults = %d", faults);
    page_hit=no_of_pages-faults;
    printf("\n\n page hit = %d",page_hit);
    return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



34. Consider a file system where the records of the file are stored one after another both physically and logically. A record of the file can only be accessed by reading all the previous records. Design a C program to simulate the file allocation strategy.

```
#include<stdio.h>
int main()
{
   char name[10][30];
   int start[10],length[10],num;
   printf("Enter the number of files to be allocated\n");
   scanf("%d",&num);
   int count=0,k,j;
   for(int i=0;i<num;i++)
   {
      printf("Enter the name of the file %d\n",i+1);
      scanf("%s",&name[i][0]);
      printf("Enter the start block of the file %d\n",i+1);
      scanf("%d",&start[i]);
      printf("Enter the length of the file %d\n",i+1);
      scanf("%d",&length[i]);
      for(j=0,k=1;j<num && k<num;j++,k++)
      {
         if(start[j+1]<=start[j] || start[j+1]>=length[j])
         {


         }
         else
         {
```

```
            count++;
          }
        }
        if(count==1)
        {
          printf("%s cannot be allocated disk space\n",name[i]);
        }
    }
    printf("File Allocation Table\n");
    printf("%s%40s%40s\n","File Name","Start Block","Length");
    printf("%s%50d%50d\n",name[0],start[0],length[0]);
    for(int i=0,j=1;i<num && j<num;i++,j++)
    {
      if(start[i+1]<=start[i] || start[i+1]>=length[i])
          {
            printf("%s%50d%50d\n",name[j],start[j],length[j]);
          }
    }
    return 0;
}
```

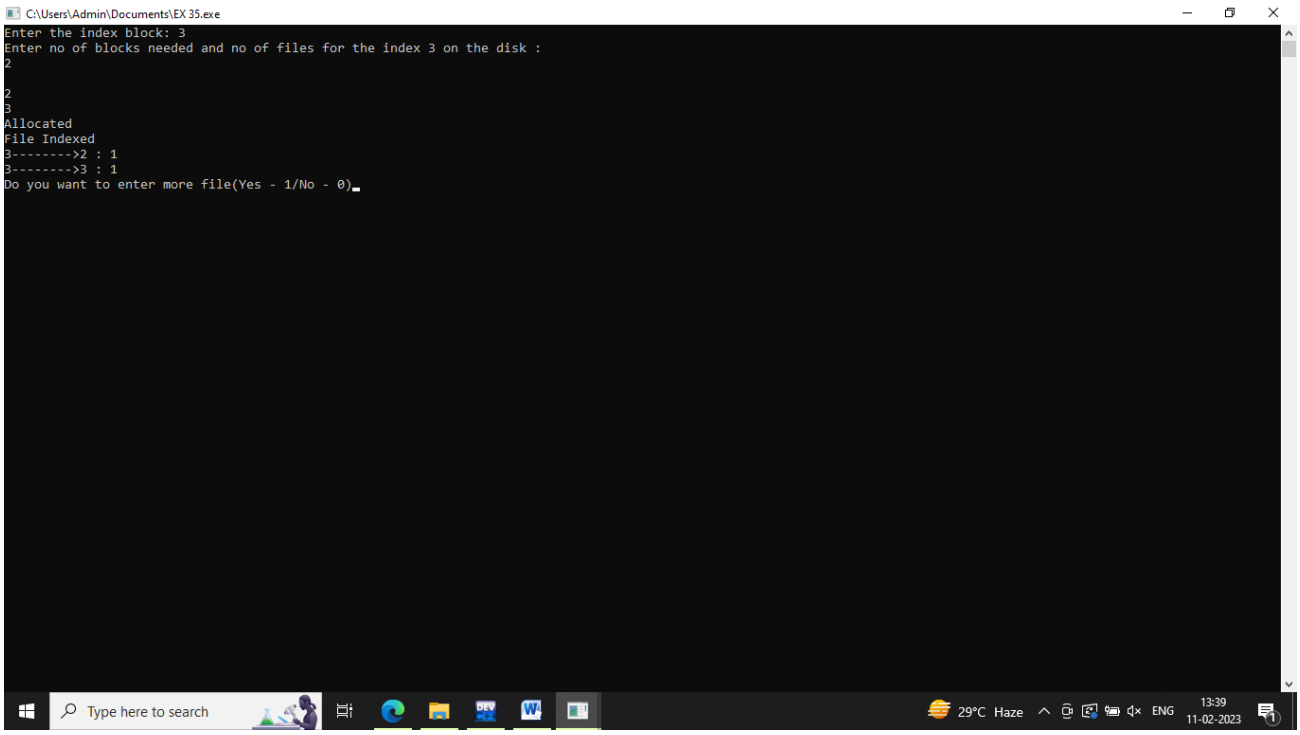


35. Consider a file system that brings all the file pointers together into an index block. The ithentry in the index block points to the ith block of the file. Design a C program to simulate the file allocation strategy.

```
#include<stdio.h>

#include<stdlib.h>

int main()

{

int f[50], index[50],i, n, st, len, j, c, k, ind,count=0;

for(i=0;i<50;i++)

f[i]=0;

x:printf("Enter the index block: ");

scanf("%d",&ind);

if(f[ind]!=1)

{

printf("Enter no of blocks needed and no of files for the index %d on the disk : \n", ind);
```

```
scanf("%d",&n);
}
else
{
printf("%d index is already allocated \n",ind);
goto x;
}
y: count=0;
for(i=0;i<n;i++)
{
scanf("%d", &index[i]);
if(f[index[i]]==0)
count++;
}
if(count==n)
{
for(j=0;j<n;j++)
f[index[j]]=1;
printf("Allocated\n");
printf("File Indexed\n");
for(k=0;k<n;k++)
printf("%d-------->%d : %d\n",ind,index[k],f[index[k]]);
}
else
{
printf("File in the index is already allocated \n");
```

```
printf("Enter another file indexed");

goto y;

}

printf("Do you want to enter more file(Yes - 1/No - 0)");

scanf("%d", &c);

if(c==1)

goto x;

else

exit(0);

}
```

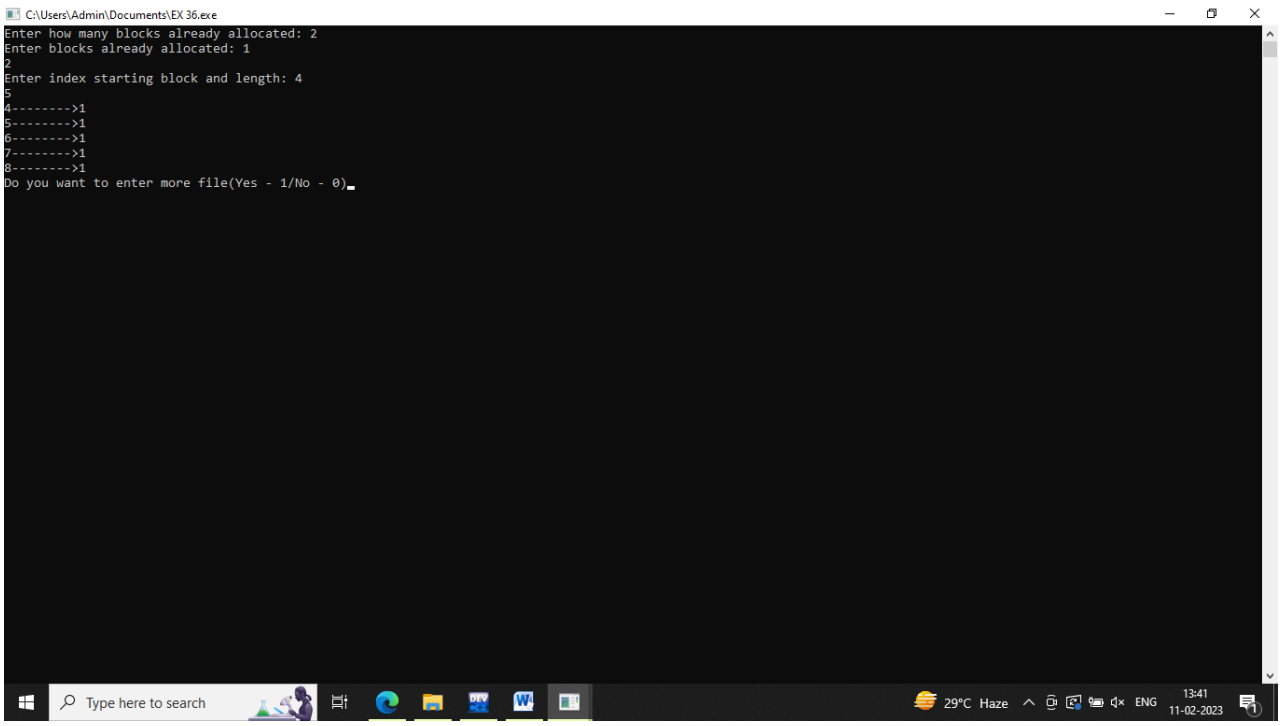


36. With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. Each block contains a pointer to the next block. Design a C program to simulate the file allocation strategy.

```
#include<stdio.h>

#include<stdlib.h>
```

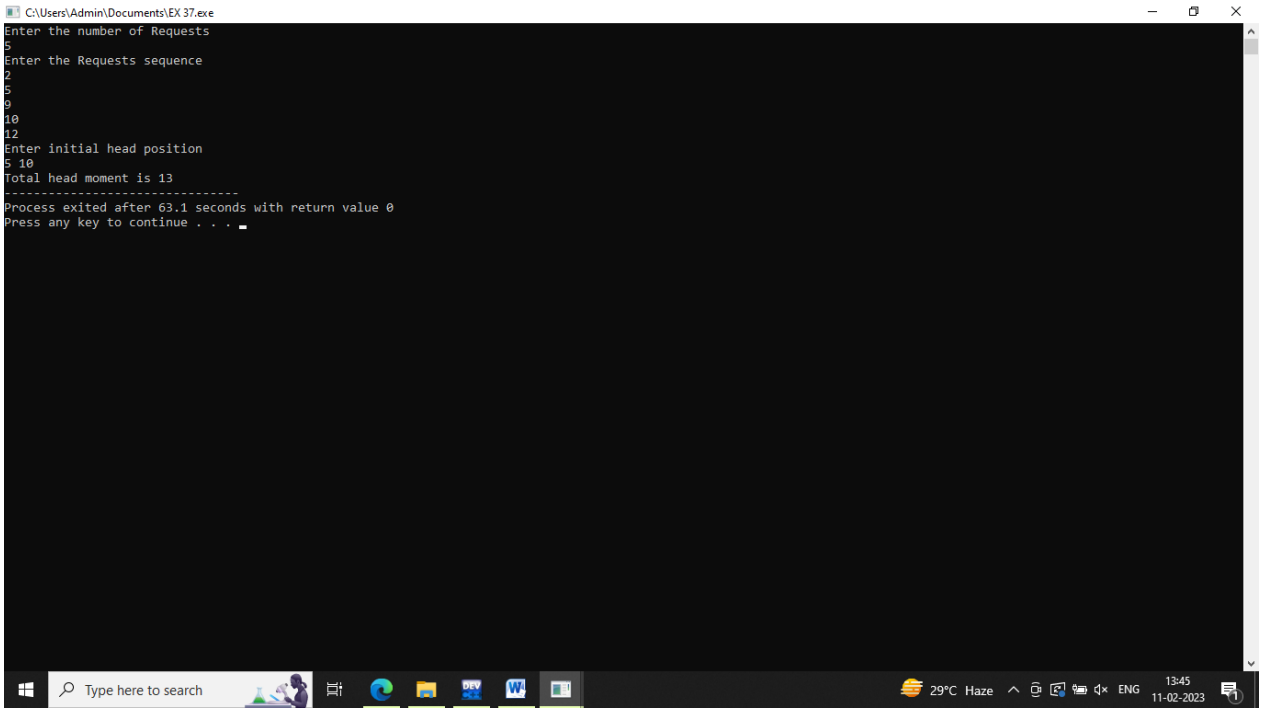
```
int main()
{
int f[50], p,i, st, len, j, c, k, a;
for(i=0;i<50;i++)
f[i]=0;
printf("Enter how many blocks already allocated: ");
scanf("%d",&p);
printf("Enter blocks already allocated: ");
for(i=0;i<p;i++)
{
scanf("%d",&a);
f[a]=1;
}
x: printf("Enter index starting block and length: ");
scanf("%d%d", &st,&len);
k=len;
if(f[st]==0)
{
for(j=st;j<(st+k);j++)
{
if(f[j]==0)
{
f[j]=1;
printf("%d-------->%d\n",j,f[j]);
}
else
```

```
{
printf("%d Block is already allocated \n",j);
k++;
}
}
}
else
printf("%d starting block is already allocated \n",st);
printf("Do you want to enter more file(Yes - 1/No - 0)");
scanf("%d", &c);
if(c==1)
goto x;
else
exit(0);
}
```



37.Construct a C program to simulate the First Come First Served disk scheduling algorithm.

```
#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;
}
```



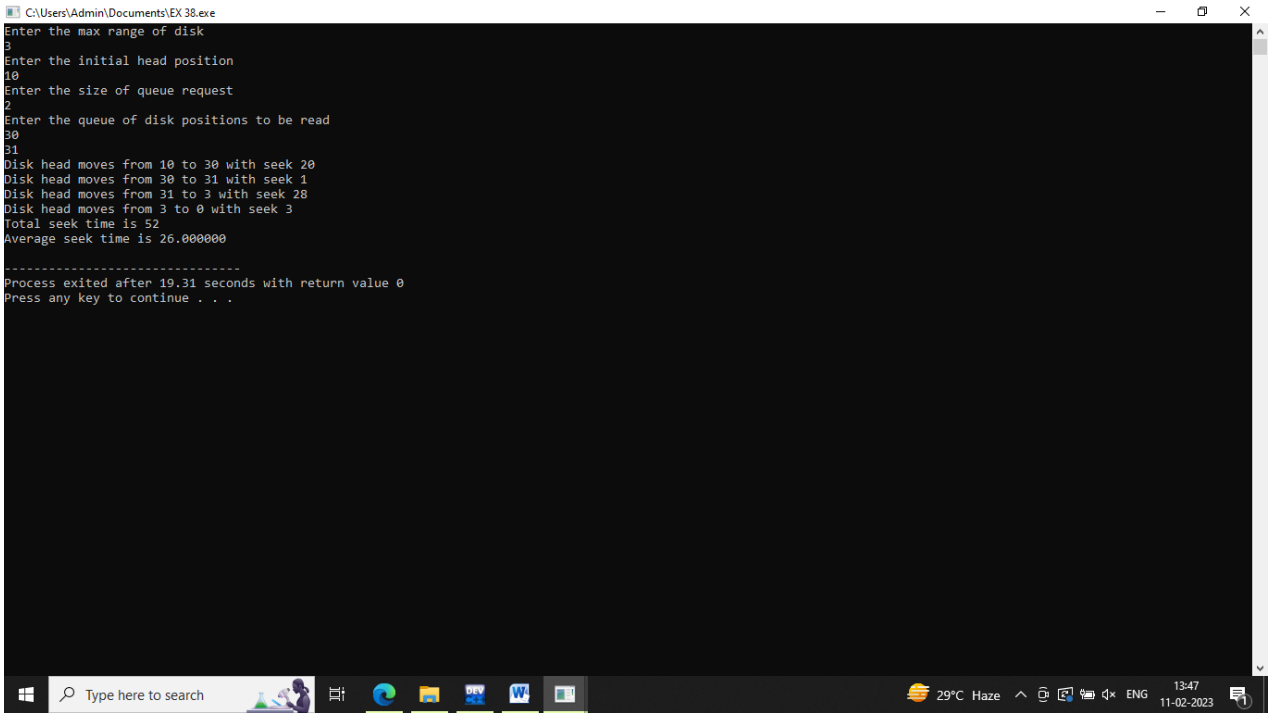
38. Design a C program to simulate SCAN disk scheduling algorithm.

```
#include <stdio.h>

#include <math.h>

int main()

{

   int queue[20], n, head, i, j, k, seek = 0, max, diff, temp, queue1[20],

   queue2[20], temp1 = 0, temp2 = 0;

   float avg;

   printf("Enter the max range of disk\n");

   scanf("%d", &max);

   printf("Enter the initial head position\n");

   scanf("%d", &head);

   printf("Enter the size of queue request\n");

   scanf("%d", &n);

   printf("Enter the queue of disk positions to be read\n");

   for (i = 1; i <= n; i++)
```

```
{
    scanf("%d", &temp);
    if (temp >= head)
    {
        queue1[temp1] = temp;
        temp1++;
    }
    else
    {
        queue2[temp2] = temp;
        temp2++;
    }
}
for (i = 0; i < temp1 - 1; i++)
{
    for (j = i + 1; j < temp1; j++)
    {
        if (queue1[i] > queue1[j])
        {
            temp = queue1[i];
            queue1[i] = queue1[j];
            queue1[j] = temp;
        }
    }
}
for (i = 0; i < temp2 - 1; i++)
```

```
{
   for (j = i + 1; j < temp2; j++)
   {
      if (queue2[i] < queue2[j])
      {
         temp = queue2[i];
         queue2[i] = queue2[j];
         queue2[j] = temp;
      }
   }
}
for (i = 1, j = 0; j < temp1; i++, j++)
   queue[i] = queue1[j];
queue[i] = max;
for (i = temp1 + 2, j = 0; j < temp2; i++, j++)
   queue[i] = queue2[j];
queue[i] = 0;
queue[0] = head;
for (j = 0; j <= n + 1; j++)
{
   diff = abs(queue[j + 1] - queue[j]);
   seek += diff;
   printf("Disk head moves from %d to %d with seek %d\n", queue[j],
   queue[j + 1], diff);
}
printf("Total seek time is %d\n", seek);
```

```
    avg = seek / (float)n;
    printf("Average seek time is %f\n", avg);
    return 0;
}
```



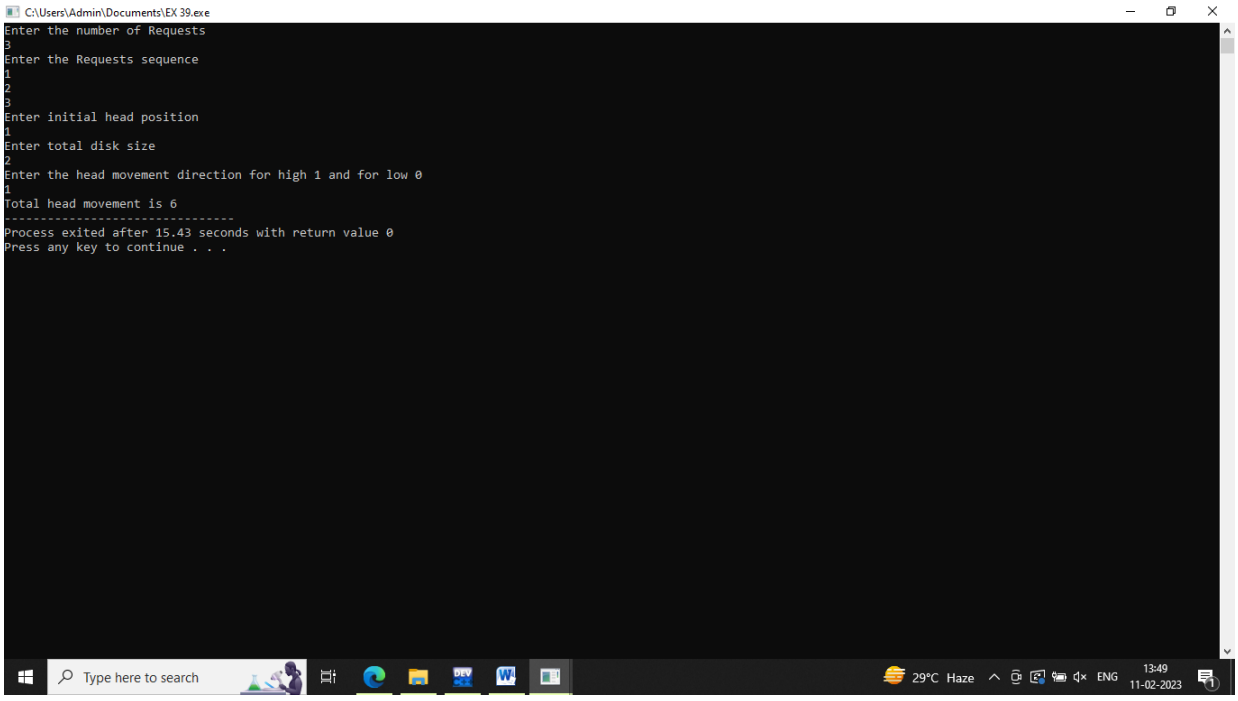
39. Develop a C program to simulate C-SCAN disk scheduling algorithm.

```
#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);
    TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
    initial=0;
    for( i=0;i<index;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);
```

```
    TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
    initial =size-1;
    for(i=n-1;i>=index;i--)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }
  }
  printf("Total head movement is %d",TotalHeadMoment);
  return 0;
}
```



40. Illustrate the various File Access Permission and different types users in Linux.

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

```
int main(int argc, char **argv) {
    int result;
    char *filename = (char *)malloc(512);
    if (argc < 2) {
        strcpy(filename, "/usr/bin/adb");
    } else {
        strcpy(filename, argv[1]);
    }
    result = access (filename, R_OK);
    if ( result == 0 ) {
        printf("%s is deva\n",filename);
    } else {
        printf("%s is deva\n",filename);
    }

    result = access (filename, W_OK);
    if ( result == 0 ) {
        printf("%s is siva\n",filename);
    } else {
        printf("%s is siva\n",filename);
    }

    result = access (filename, X_OK);
    if ( result == 0 ) {
        printf("%s is executable\n",filename);
```

```
    } else {
        printf("%s is executable\n",filename);
    }
    free(filename);
    return 0;
}
```