Week - 10
(24 August 2023)
Experiment - 9

Question:

(i) Write a C program to simulate the following file organization techniques:

(a) Single level directory

(b) Two level directory

(c) Hierarchical

(ii) Write a C program to simulate disk scheduling algorithms:

(a) FCFS

(b) SCAN

(c) c-SCAN

(iii) Write a C program to simulate disk scheduling algorithms:

(a) SSTF

(b) LOOK

(c) c-LOOK

Program:

(i) File Organization Techniques:

(a) Single Level Directory:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.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\n1. Create File\t2. Delete File\t3. Search File \n4. Display Files\t5.
Exit\nEnter your choice -- ");
```

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

switch(ch)
{
   case 1: printf("\nEnter the name of the file -- ");
           scanf("%s",dir.fname[dir.fcnt]);
           dir.fcnt++;
           break;

   case 2: printf("\nEnter 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("\nEnter 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("\nDirectory Empty");
           else
           {
              printf("\nThe Files are -- ");
              for(i=0;i<dir.fcnt;i++)
```

```
                printf("\t%s",dir.fname[i]);
            }
            break;

        default: exit(0);
    }
  }
}
```

(b) <u>Two Level Directory:</u>

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

struct
{
  char dname[10],fname[10][10];
  int fcnt;
}dir[10];

void main()
{
  int i,ch,dcnt,k;
  char f[30], d[30];
  dcnt=0;

  while(1)
  {
    printf("\n1. Create Directory\t2. Create File\t3. Delete File");
    printf("\n4. Search File\t\t5. Display\t6. Exit\nEnter your choice --");
    scanf("%d",&ch);

  switch(ch)
  {
    case 1: printf("\nEnter name of directory -- ");
        scanf("%s", dir[dcnt].dname);
        dir[dcnt].fcnt=0;
        dcnt++;
        printf("Directory created");
        break;

    case 2: printf("\nEnter name of the directory -- ");
        scanf("%s",d);
        for(i=0;i<dcnt;i++)
          if(strcmp(d,dir[i].dname)==0)
          {
            printf("Enter name of the file -- ");
```

```
            scanf("%s",dir[i].fname[dir[i].fcnt]);
            dir[i].fcnt++;
            printf("File created");
            break;
         }
      if(i==dcnt)
         printf("Directory %s not found",d);
      break;

case 3: printf("\nEnter name of the directory -- ");
      scanf("%s",d);
      for(i=0;i<dcnt;i++)
      {
         if(strcmp(d,dir[i].dname)==0)
         {
            printf("Enter name of the file -- ");
            scanf("%s",f);
            for(k=0;k<dir[i].fcnt;k++)
            {
               if(strcmp(f, dir[i].fname[k])==0)
               {
                  printf("File %s is deleted ",f);
                  dir[i].fcnt--;
                  strcpy(dir[i].fname[k],dir[i].fname[dir[i].fcnt]);
                  goto jmp;
               }
            }

         printf("File %s not found",f);
         goto jmp;
      }
      }

      printf("Directory %s not found",d);
      jmp : break;

case 4: printf("\nEnter name of the directory -- ");
      scanf("%s",d);
      for(i=0;i<dcnt;i++)
      {
         if(strcmp(d,dir[i].dname)==0)
         {
            printf("Enter the name of the file -- ");
            scanf("%s",f);
            for(k=0;k<dir[i].fcnt;k++)
            {
               if(strcmp(f, dir[i].fname[k])==0)
```

```
                    {
                        printf("File %s is found ",f);
                        goto jmp1;
                    }
                }
                printf("File %s not found",f);
                goto jmp1;
            }
        }
        printf("Directory %s not found",d);
        jmp1: break;

    case 5: if(dcnt==0)
            printf("\nNo Directory's ");
        else
        {
            printf("\nDirectory\tFiles");
            for(i=0;i<dcnt;i++)
            {
                printf("\n%s\t\t",dir[i].dname);
                for(k=0;k<dir[i].fcnt;k++)
                printf("\t%s",dir[i].fname[k]);
            }
        }
        break;
    default:exit(0);
  }
}
}
```

(c) <u>Hierarchical</u>:

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
//#include<graphics.h>

struct tree_element
{
  char name[20];
  int x,y,ftype,lx,rx,nc,level;
  struct tree_element *link[5];
};

typedef struct tree_element node;

void main()
{
```

```
   int gm;
   node *root;
   root=NULL;

   create(&root,0,"root",0,639,320);

   //initgraph(&gd,&gm,"c:\\tc\\BGI");
   display(root);

   //closegraph();
}

create(node **root,int lev,char *dname,int lx,int rx,int x)
{
   int i,gap;
   if(*root==NULL)
   {
      (*root)=(node *)malloc(sizeof(node));

      printf("Enter name of dir/file(under %s) :",dname);
      fflush(stdin);
      gets((*root)->name);

      printf("enter 1 for Dir/2 forfile :");
      scanf("%d",&(*root)->ftype);

      (*root)->level=lev;
      (*root)->y=50+lev*50;
      (*root)->x=x;
      (*root)->lx=lx;
      (*root)->rx=rx;

      for(i=0;i<5;i++)
         (*root)->link[i]=NULL;

      if((*root)->ftype==1)
      {
         printf("No of sub directories/files(for %s):",(*root)->name); scanf("%d",&(*root)-
>nc);
         if((*root)->nc==0)
            gap=rx-lx;
         else
            gap=(rx-lx)/(*root)->nc;

         for(i=0;i<(*root)->nc;i++)
            create(&((*root)->link[i]),lev+1,(*root)-
>name,lx+gap*i,lx+gap*i+gap,lx+gap*i+gap/2);
```

```
        }

        else (*root)->nc=0;
    }
}

/*display(node *root)
{
    int i;
    settextstyle(2,0,4);
    settextjustify(1,1);
    setfillstyle(1,BLUE);
    setcolor(14);

    if(root!=NULL)
    {
        for(i=0;i<root->nc;i++)
        {
            line(root->x,root->y,root->link[i]->x,root->link[i]->y);
        }

        if(root->ftype==1)
            bar3d(root->x-20,root->y-10,root->x+20,root->y+10,0,0);

        else
            fillellipse(root->x,root->y,20,20);

        outtextxy(root->x,root->y,root->name);

        for(i=0;i<root->nc;i++)
        {
            display(root->link[i]);
        }
    }
}*/
```

Output (File Organization Techniques):
(a) Single Level Directory:

```
Enter name of directory -- BMSCE


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

Enter the name of the file -- 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 -- ISE


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

The Files are --        CSE     ISE

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

Enter the name of the file -- CSE
File CSE is deleted
```

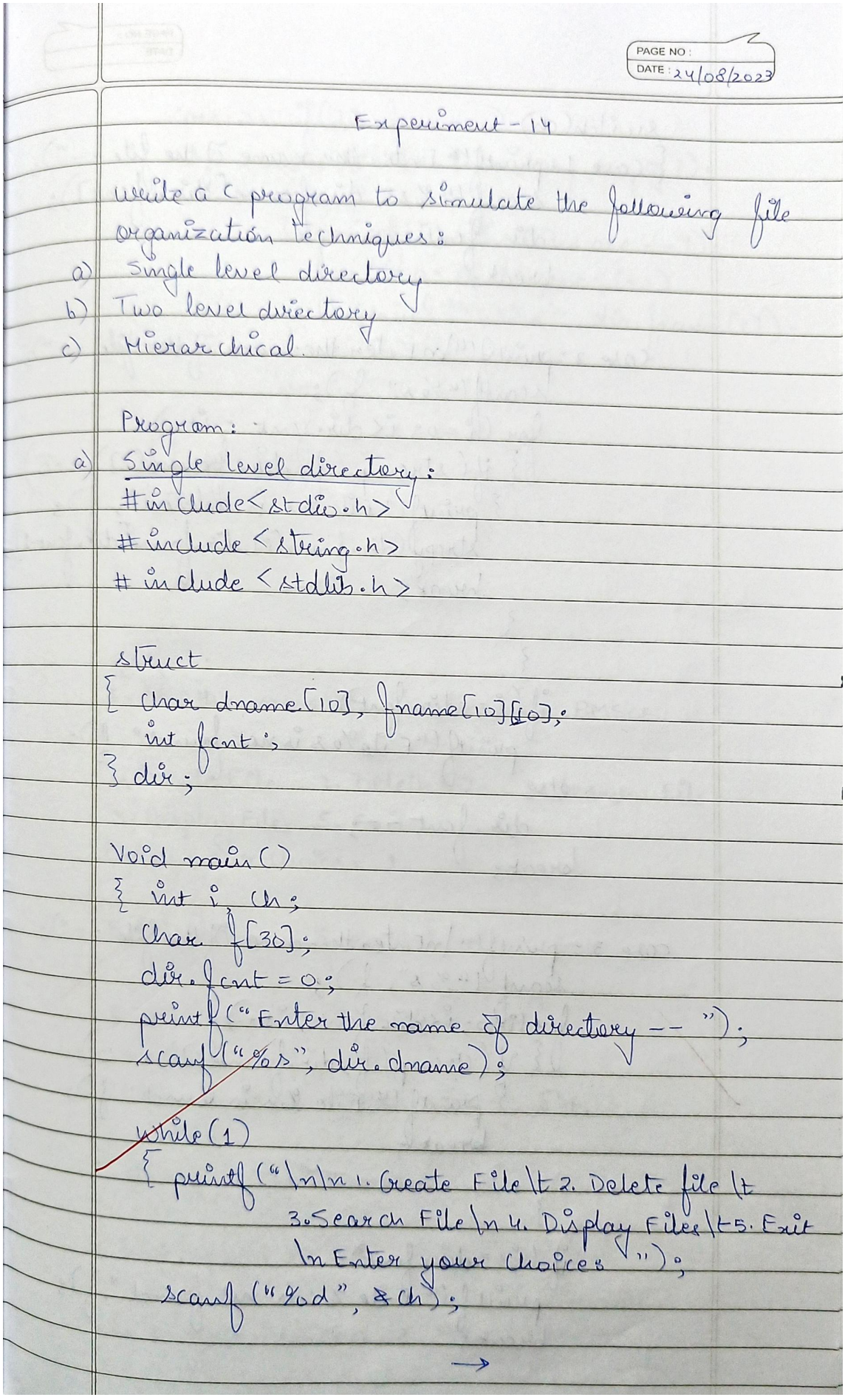
```
1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice -- 3

Enter the name of the file -- CSE
File CSE not found

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

The Files are --        ISE

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

(b) Two Level Directory:

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

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

Directory       Files
BMSCE                   CSE     ISE
```

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

Enter name of the directory -- BMSCE
Enter name of the file -- CSE
File CSE is deleted
1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice --4

Enter name of the directory -- BMSCE
Enter the name of the file -- CSE
File CSE not found
1. Create Directory     2. Create File  3. Delete File
4. Search File          5. Display      6. Exit
Enter your choice --6
```

Observation Book Pictures (File Organization Techniques):



PAGE NO :
DATE : 24/08/2023

Experiment - 14

Write a C program to simulate the following file organization techniques:

a) Single level directory
b) Two level directory
c) Hierarchical

Program:

a) Single level directory:

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

struct
{ char dname[10], fname[10][10];
  int fcnt;
} dir;

Void main()
{ int i, ch;
  char f[30];
  dir.fcnt = 0;
  printf("Enter the name of directory -- ");
  scanf("%s", dir.dname);

  while(1)
  { printf("\n\n1. Create File\t2. Delete file\t
           3. Search File\n4. Display Files\t5. Exit
           \nEnter your choice: ");
    scanf("%d", &ch);
```

→

```
switch(ch)
{ case 1: printf("Enter the name of the file --");
          scanf("%s", dir.fname[dir.fcnt]);
          dir.fcnt++;
          break;

case 2: printf("\nEnter 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 is not found", f);
      else
        dir.fcnt--;
      break;

case 3: printf("\nEnter 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 is 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);
}
}
}
```

Output:

Enter the name of directory -- BMSCE

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

Enter the name of the File -- CSE

Enter your choice: 1

Enter the name of the file -- ISE

Enter your choice: 4

The Files are -- CSE ISE

Enter your choice: 2

Enter the name of the File -- CSE
File CSE is deleted.

Enter your choice -- 3

Enter the name of file -- CSE
File CSE not found

Enter your choice -- 5
Exit.

b) Two level directory:

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

struct
{ char dname[10], fname[10][10];
  int fcnt;
} dir;

void main()
{ int i, ch;
  char f[30];
  dir.fcnt = 0;

  printf("\n Enter the name of the directory: ");
  scanf("%s", dir.dname);

  while(1)
  { printf("\n\n 1. Create File \t 2. Delete File \t
           3. Search File \t 4. Display Files \t 5. Exit \n
           Enter your choice: ");
```

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

switch(ch)
{ case 1: printf("\n Enter the name of file: ");
          scanf("%s", dir.fname[dir.fcnt]);
          dir.fcnt++;
          break;

case 2: printf("\n Enter the name of 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("Enter the name of 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);
  }
}
```

Output 1:

Enter the name of directory : BMSCE

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

Enter the name of File -- ISE

Enter your choice : 1

Enter the name of File -- CSE

Enter your choice : 4

The Files are -- ISE CSE

Enter your choice : 2.

Enter the name of the file -- CSE
File CSE has been deleted.

Enter your choice :- 3

Enter the name of the file -- CSE
File CSE not found.

Enter your choice : 5
Exit.

c) Hierarchical :

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

struct tree_element
{ char name[20];
  int x, y, ftype, lx, rx, nc, level;
  struct tree_element *link[5];
};
typedef struct tree_element
node;

void main()
{ int gd = DETECT, gm;
  node *root;
  root = NULL;
  create(&root, 0, "root", 0, 639, 320);
  initgraph(&gd, &gm, "C:\\tc\\BGI");
  display(root);
  getch();
  closegraph();
}
```

→

```
create(node **root, int lev, char *dname, int lx,
        int rx, int x)
{ int i,gap ;
  if (*root == NULL)
  { (*root) = (node *) malloc( sizeof(node));
    printf("Enter the name of dir/file(under %s)",
            dname);
    fflush (stdin);
    gets((*root) -> name);
    printf(" enter 1 for Dir/2 for file : ");
    scanf("%d", &(*root) -> ftype );
    (*root) -> level = lev;
    (*root) -> y = 50+ lev *50;
    (*root) -> x = x;
    (*root) -> lx = lx ;
    (*root) -> rx = rx;
    for (i=0; i<5; i++)
    (*root) -> link[i] = NULL;
    if ((*root) -> ftype == 1 )
    { printf(" No. of sub directories/files (for %s):',
      (*root) -> name);
      scanf ("%d", &(*root) -> nc);
      if ((*root) -> nc == 0)
        gap = rx - lx
      else
        gap = (rx - lx)/(*root) -> nc ;
      for (i=0; i<(*root) -> nc ; i++)
      create (&((*root) -> link[i]), lev+1,
              (*root) -> name, lx + gap * i,
              lx+gap*i + gap, lx + gap * i +
              gap/2);
    }
```

```
create(node **root, int lev, char *dname, int lx,
       int rx, int x)
{ int i,gap;
  if (*root == NULL)
  { (*root) = (node *) malloc(sizeof(node));
    printf("Enter the name of dir/file(under %s):",
           dname);
    fflush(stdin);
    gets((*root)->name);
    printf("enter 1 for Dir/2 for file : ");
    scanf("%d",&(*root)->ftype);
    (*root)->level = lev;
    (*root)->y = 50+lev*50;
    (*root)->x = x;
    (*root)->lx = lx;
    (*root)->rx = rx;
    for (i=0; i<5; i++)
    (*root)->link[i] = NULL;
    if((*root)->ftype == 1)
    { printf("No. of sub directories/files(for %s):",
      (*root)->name);
      scanf("%d",&(*root)->nc);
      if((*root)->nc == 0)
        gap = rx-lx
      else
        gap = (rx-lx)/(*root)->nc;
      for(i=0; i<(*root)->nc; i++)
      create(&((*root)->link[i]), lev+1,
             (*root)->name, lx+gap*i,
             lx+gap*i+gap, lx+gap*i+
             gap/2);
    }
```

```
        else
        (*root) ->nc = 0;
      }
    }
/*  display(node *root)
{   int i;
    settextstyle(2,0,4);
    settextjustify(1,1);
    setfillstyle(1,BLUE);
    setcolor(14);
    if (root != NULL)
    {   for(i=0; i<(root)->nc; i++)
        {   line(root->x, root->y, root->link[i]->x,
                 root->link[i]->y);
        }
        if(root->ftype==1) bar3d(root->x-20,
                                 root->y-10,
                                 root->x+20,
                                 root->y+10, 0, 0);
        else
        fillellipse(root->x, root->y, 20, 20);

        outtextxy(root->x, root->y, root->name);

        for(i=0; i<root->nc; i++)
        {   display(root->link[i]);
        }
    }
}
*/
```

→

PAGE NO :
DATE :

Output :

Enter the Name of dir/file(under root) : ROOT
Enter 1 for Dir/ 2 for file : 1
No. of subdirectories/files (for ROOT) : 2
Enter the Name of dir/file(under ROOT) : USER 1
Enter 1 for Dir/2 for file : 1
No. of subdirectories/files (for User 1) : 1
Enter the name of dir/file(under USER 1) : SUBDIR
Enter 1 for Dir/2 for file : 1
No. of subdirectories/files (for SUBDIR) : 2
Enter the name of dir/file(under SUBDIR) : JAVA
Enter 1 for Dir/2 for file : 1
NO. of subdirectories/files for JAVA : 0
Enter the Name of dir/file (under SUBDIR) : VB
Enter 1 for Dir/2 for file : 1
NO. of subdirectories/files (for VB) = 0

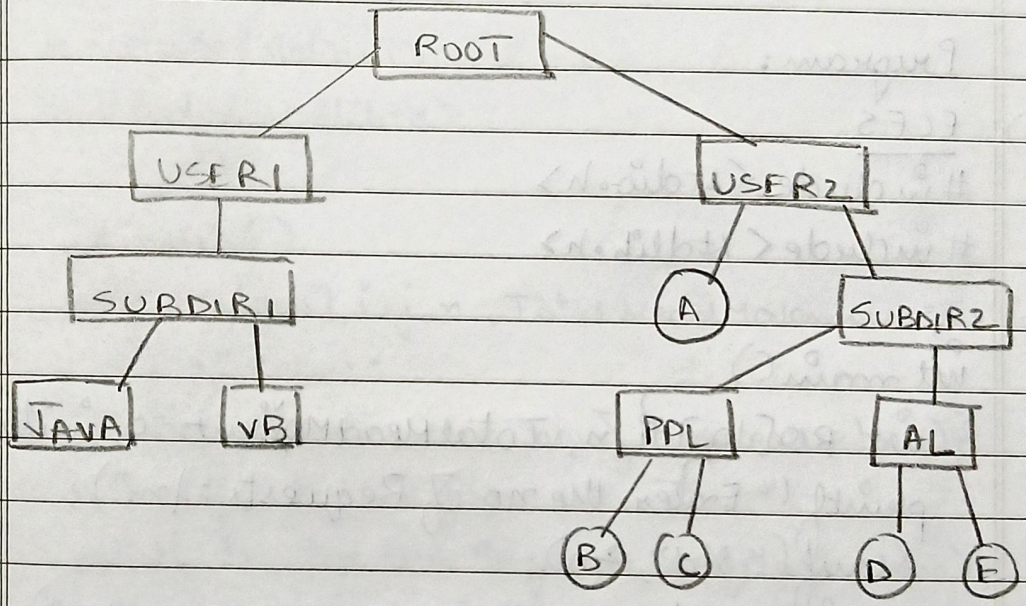
Enter the Name of dir/file(under ROOT) : USER2
Enter 1 for Dir/2 for file : 1
No. of subdirectories/files (for USER2) : 2
Enter the Name of dir/file (under ROOT) : A
Enter 1 for Dir/2 for file : 2
Enter Name of dir/file (under USER2) : SUBDIR2
Enter 1 for Dir/2 for file : 1
No. of subdirectories/files (for SUBDIR2) : 2
Enter the Name of dir/file (under SUBDIR2) : PPL
Enter 1 for Dir/2 for file : 1
No. of subdirectories/files (for PPL) : 2
Enter Name of dir/file (under PPL) : B
Enter 1 for Dir/2 for file : 2
Enter Name of Dir/file (under PPL) : C
Enter 1 for Dir/2 for file : 2
Enter the Name of dir/file(under SUBDIR) : AI

PAGE NO :
DATE :

Enter 1 for Dir/2 for file : 1
No. of subdirectories / files (for AI) : 2
Enter name of dir/file (under AI) : D
Enter 1 for Dir/2 for file : 2
Enter Name of dir/file (under AI) : E
Enter 1 for Dir / 2 for File : 2

10/10

24/8/23



(ii) Disk Scheduling Algorithms:

(a) FCFS:

```
#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);

   // logic for FCFS disk scheduling

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

   printf("Total head moment is %d",TotalHeadMoment);
   return 0;

}
```

(b) SCAN:

```
#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);

   // logic for Scan disk scheduling
```

```
   /*logic for sort the request array */
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 movement is towards high value
if(move==1)
{
   for(i=index;i<n;i++)
   {
      TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
      initial=RQ[i];
   }
   //  last movement for max size
   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];

   }
}
// if movement is towards low value
else
```

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

(c) <u>c-SCAN:</u>

```
#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);

    // logic for C-Scan disk scheduling

       /*logic for sort the request array */
    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 movement is towards high value
if(move==1)
{
    for(i=index;i<n;i++)
    {
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
        initial=RQ[i];
    }
    //  last movement for max size
    TotalHeadMoment=TotalHeadMoment+abs(size-RQ[i-1]-1);
    /*movement max to min disk */
    TotalHeadMoment=TotalHeadMoment+abs(size-1-0);
    initial=0;
    for( i=0;i<index;i++)
    {
         TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
         initial=RQ[i];

    }
}
// if movement is towards low value
else
{
    for(i=index-1;i>=0;i--)
    {
```

```
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }
        //  last movement for min size
        TotalHeadMoment=TotalHeadMoment+abs(RQ[i+1]-0);
        /*movement min to max disk */
        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;
}
```

Output (Disk Scheduling Algorithms):

(a) FCFS:

```
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
```

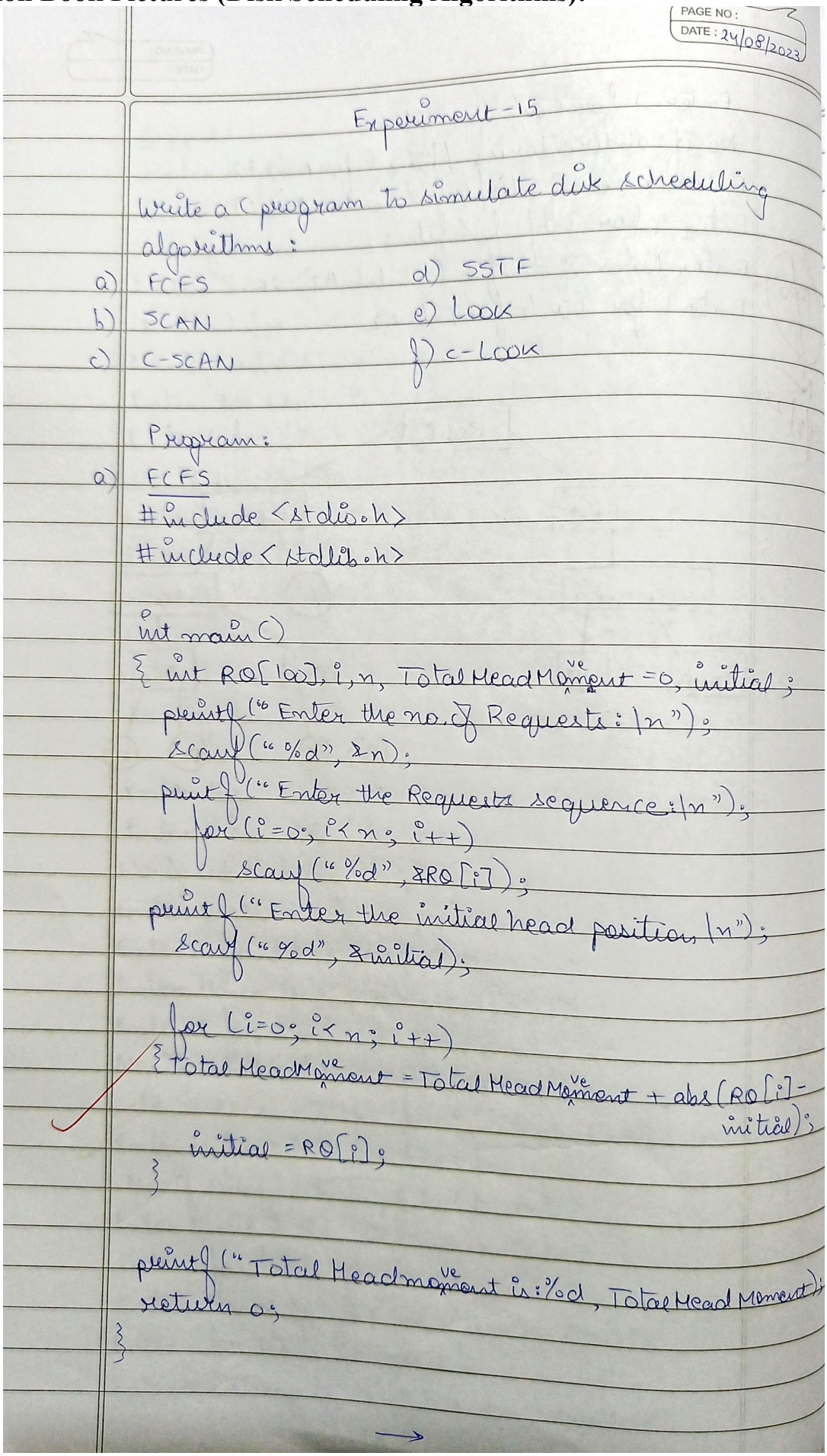
(b) SCAN:

```
Enter the number of Requests
6
Enter the Requests sequence
90 120 30 60 50 80
Enter initial head position
70
Enter total disk size
200
Enter the head movement direction for high 1 and for low 0
0
Total head movement is 190
```

(c) C-SCAN:

```
Enter the number of Requests
3
Enter the Requests sequence
2 1 0
Enter initial head position
1
Enter total disk size
3
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 4
```

Observation Book Pictures (Disk Scheduling Algorithms):



PAGE NO :
DATE : 24/08/2023

Experiment - 15

Write a C program to simulate disk scheduling algorithms :

a) FCFS
b) SCAN
c) C-SCAN
d) SSTF
e) Look
f) c-Look

Program:

a) FCFS

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

int main()
{ int RQ[100], i, n, TotalHeadMovement = 0, initial;
  printf("Enter the no. of Requests : \n");
  scanf("%d", &n);
  printf("Enter the Request sequence : \n");
  for (i=0; i<n; i++)
      scanf("%d", &RQ[i]);
  printf("Enter the initial head position, \n");
  scanf("%d", &initial);

  for (i=0; i<n; i++)
  { TotalHeadMovement = TotalHeadMovement + abs(RQ[i] -
                                                initial);
    initial = RQ[i];
  }

  printf("Total HeadMovement is : %d", TotalHeadMovement);
  return 0;
}
```

PAGE NO :
DATE :

Output:

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 Movement is: 644

b) SCAN:

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

int main()
{ int RQ[100],i,j,n,TotalHeadMovement=0,initial,
  size,move;
  printf("Enter the no. of Requests:\n");
   scanf("%d",&n);
  printf("Enter the Requests sequence:\n");
   for(i=0;i<n;i++)
    scanf("%d",&RQ[i]);
  printf("Enter the 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);
   scanf("%d",&move);

 // logic for SCAN Disk Scheduling
 //logic for sort the request array.
  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 movement is towards high value
    if(move==1)
    { for(i=index; i<n; i++)
        { TotalHeadMovement = TotalHeadMovement +
                              abs(RQ[i]-initial);
          initial = RQ[i];
        }
    // last movement for max size
    TotalHeadMovement = TotalHeadMovement +
                        abs(size-RQ[i-1]-1);
    initial = size-1;
    for(i=index-1; i>=0; i--)
    { TotalHeadMovement = TotalHeadMovement +
                          abs(RQ[i]-initial);
      initial = RQ[i];
    }
}
```

```
// if movement is towards low value
else
{
    for (i = index - 1; i >= 0; i--)
    { TotalHeadMovement = TotalHeadMovement +
                          abs(RQ[i] - initial);
      initial = RQ[i];
    }

// last movement for min size
TotalHeadMovement = TotalHeadMovement +
                    abs(RQ[i+1] - 0);
initial = 0;
for (index; i < n; i++)
{ TotalHeadMovement = TotalHeadMoment +
                      abs(RQ[i] - initial);
  initial = RQ[i];
}
}

printf("Total Head Movement is : %d",
       TotalHeadMovement);
return 0;
}
```

Output :

```
Enter the no. of Requests : 6
Enter the Requests Sequence :
90 120 30 60 50 80
Enter initial head position : 70
Enter total disk size : 200
Enter the head movement direction : for high-1 / for low-0 :
0
Total head movement is 190.
```

PAGE NO :
DATE :

c) C-SCAN

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

int main()
{ int RQ[100], i, j, n, initial, size, move;
  int TotalHeadMovement = 0;
  printf("Enter the no. of Requests: ");
   scanf("%d", &n);
  printf("Enter the Request sequence: \n");
   for(i=0; i<n; i++)
     scanf("%d", &RQ[i]);
  printf("Enter the initial head position: \n");
   scanf("%d", &initial);
  printf("Enter total disk size: \n");
   scanf("%d", &size);
  printf("Enter the head movement direction for hig
          for low - 0: \n");
   scanf("%d", &move);

  // logic for sort the request array
  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 movement is towards high value
if(move == 1)
{ for(i=index; i<n; i++)
  { TotalHeadMovement = TotalHeadMovement +
                        abs(RQ[i] - initial);
    initial = RQ[i];
  }
  // last movement for max ~~value~~ size
  TotalHeadMovement = TotalHeadMovement +
                      abs(size - RQ[i-1] - 1);
  // movement max to min disk.
  TotalHeadMovement = TotalHeadMovement +
                      abs(size - 1 - 0);
  initial = 0;
  for(i=0; i<index; i++)
  { TotalHeadMovement = TotalHeadMovement +
                        abs(RQ[i] - initial);
    initial = RQ[i];
  }
}
else // if movement is towards low value
{ for(i=index-1; i>=0; i--)
  { TotalHeadMovement = TotalHeadMovement +
                        abs(RQ[i] - initial);
    initial = RQ[i];
  }
```

→

```
//movement min to max disk
TotalHeadMovement = TotalHeadMovement +
                    abs(size-1-0);
initial = size-1;
for (i=n-1; i>=index; i--)
{ TotalHeadMovement = TotalHeadMovement +
                      abs(RQ[i] - initial);
  initial = RQ[i]
}

// last movement for min size
TotalHeadMovement = TotalHeadMovement +
                    abs(RQ[i+1] - 0);
}
printf("Total Head Movement is: %d",
       TotalHeadMovement);
return 0;
}
```

Output:

```
Enter the no. of Requests: 3
Enter the Requests sequence:
2 1 0
Enter initial head position: 1
Enter Total disk size: 3
Enter the head movement direction for high-1 for low-0:
1
Total Head Movement: 4
```

10/10

24/8/23

(iii) Disk Scheduling Algorithms:

(a) SSTF:

```
#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;
        }

      }
      TotalHeadMoment=TotalHeadMoment+min;
      initial=RQ[index];
      // 1000 is for max
      // you can use any number
      RQ[index]=1000;
      count++;
   }

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

(b) LOOK:

```
#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);

   // logic for look disk scheduling

      /*logic for sort the request array */
   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 movement is towards high value
   if(move==1)
   {
      for(i=index;i<n;i++)
      {
```

```
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

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

        }
    }
    // if movement is towards low value
    else
    {
        for(i=index-1;i>=0;i--)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];
        }

        for(i=index;i<n;i++)
        {
            TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
            initial=RQ[i];

        }
    }

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

(c) <u>c-LOOK</u>:

```
#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);

   // logic for C-look disk scheduling

      /*logic for sort the request array */
   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 movement is towards high value
```

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

       for( i=0;i<index;i++)
       {
           TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
           initial=RQ[i];

       }
    }
    // if movement is towards low value
    else
    {
       for(i=index-1;i>=0;i--)
       {
          TotalHeadMoment=TotalHeadMoment+abs(RQ[i]-initial);
          initial=RQ[i];
       }

       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;
}
```

Output (Disk Scheduling Algorithms):

(a) SSTF:

```
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 movement is 236
```

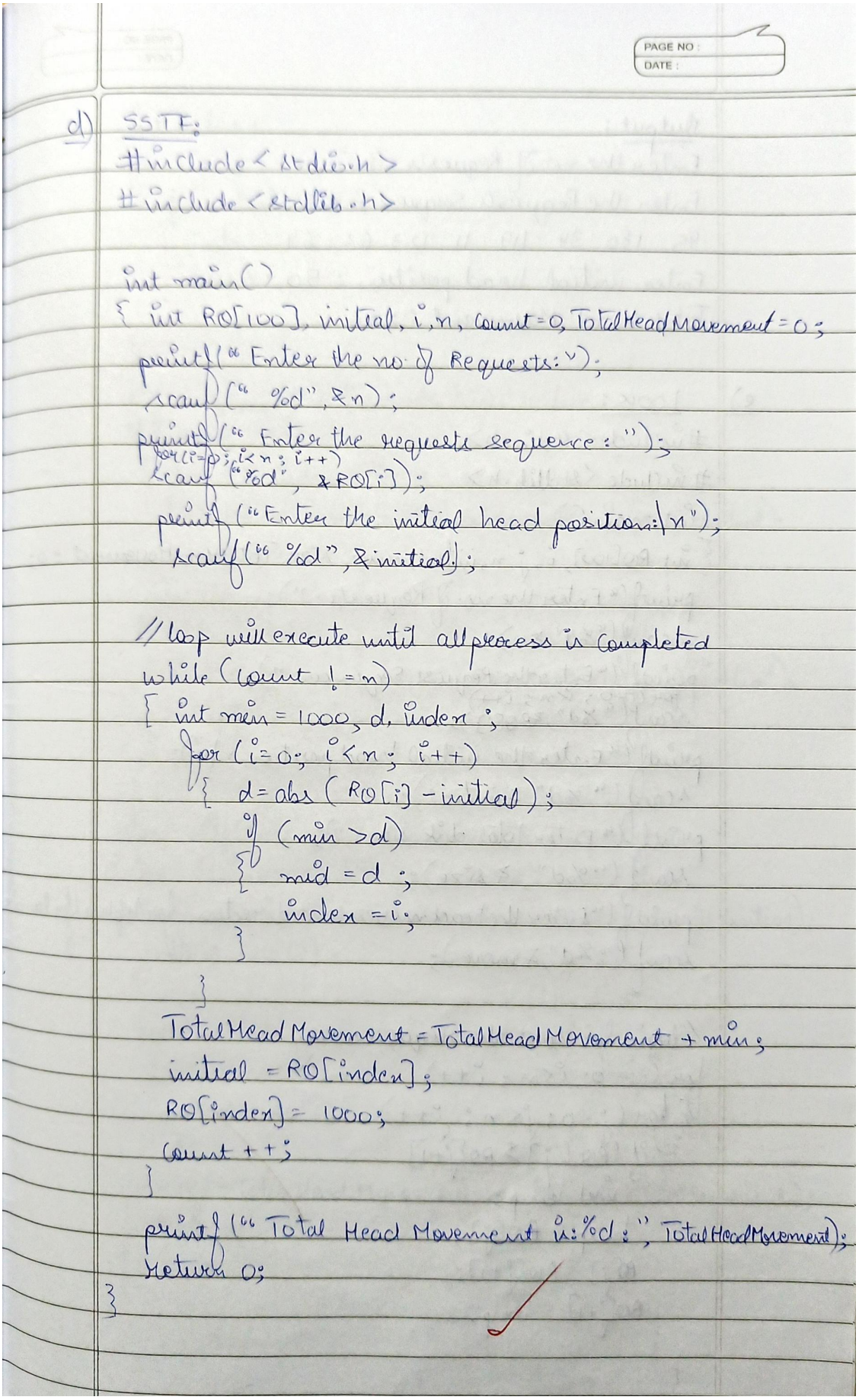
(b) LOOK:

```
Enter the number of Requests
3
Enter the Requests sequence
2 1 0
Enter initial head position
1
Enter total disk size
3
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 3
```

(c) c-LOOK:

```
Enter the number of Requests
3
Enter the Requests sequence
2 1 0
Enter initial head position
1
Enter total disk size
3
Enter the head movement direction for high 1 and for low 0
1
Total head movement is 4
```

Observation Book Pictures (Disk Scheduling Algorithms):



d) SSTF:

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

int main()
{ int RQ[100], initial, i, n, count = 0, TotalHeadMovement = 0;
  printf("Enter the no of Requests:");
  scanf("%d", &n);
  printf("Enter the requests sequence:");
  for(i=0; i<n; i++)
  scanf("%d", &RQ[i]);
  printf("Enter the initial head position:\n");
  scanf("%d", &initial);

// 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;
    }
  }
  TotalHeadMovement = TotalHeadMovement + min;
  initial = RQ[index];
  RQ[index] = 1000;
  count++;
}
printf("Total Head Movement is:%d:", TotalHeadMovement);
return 0;
}
```

Output:

```
Enter the no of Requests : 8
Enter the Requests Sequence :
95 180 34 119 11 123 62 64
Enter initial head position : 50
Total Head Movement : 236
```

e) Look:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{ int RQ[100], i, j, n, initial, move, size, TotalHeadMovement = 0;
  printf("Enter the no. of Requests : ");
  scanf("%d", &n);
  printf("Enter the Request Sequence : ");
  for(i=0; i<n; i++)
  scanf("%d", &RQ[i]);
printf("Enter the initial head position : \n");
  scanf("%d", &initial);
printf("Enter total disk size : ");
  scanf("%d", &size);
printf("Enter the head movement direction for high-1 for low");
  scanf("%d", &move);

// logic for sort the request array
for (i = 0; i<n; i++)
{ for (j = 0; j<n; 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 movement is towards high value
if (move == 1)
{ for (i= index; i< n; i++)
  { TotalHeadMovement = TotalHeadMovement + abs(RQ[i] - initial);
    initial = RQ[i];
  }
 for (i= index-1; i>=0; i--)
 { TotalHeadMovement = TotalHeadMovement + abs(RQ[i] - initial);
   initial = RQ[i];
 }

else // if movement is towards low value
{ for (i= index-1; i>=0; i--)
{ TotalHeadMovement = TotalHeadMovement + abs(RQ[i] - initial);
  initial = RQ[i];
}

 for (i= index; i<n; i++)
 { TotalHeadMovement += abs(RQ[i] - initial);
   initial = RQ[i];
 } }
printf("Total Head Movement is: %d", TotalHeadMovement);
return 0;
}
```

→

Output:

```
Enter the no. of Requests : 3
Enter Request Sequence : 2 1 0
Enter initial head position : 1
Enter total disk size : 3
Enter head movement direction for high-1 for low-0 : 1
Total Head Movement : 3
```

g) c-look:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{ int RQ[100], i, j, n, initial, size, move, TotalHeadMovement=0;
  printf("Enter the no. of requests : ");
    scanf("%d", &n);
  printf("Enter the Request Sequance : ");
    for(i=0; i<n; i++)
      scanf("%d", &RQ[i]);
  printf("Enter initial head position : ");
    scanf("%d", &initial);
  printf("Enter total disk size : ");
    scanf("%d", &size);
  printf("Enter head movement direction for high-1 for low-0 : ");
    scanf("%d", &move);

  for(i=0; i<n; i++) // logic for sort the request array
  { for(j=0; j<n; 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) // if movement is towards high value
{ for (i = index; i< n; i++)
  { TotalHeadMovement += abs(RQ[i] - initial);
    initial = RQ[i];
  }
  for (i= 0; i<index; i++)
  { TotalHeadMovement += abs(RQ[i] - initial);
    initial = RQ[i];
  }
}
else // if movement is towards low value
{ for (i = index-1; i >= 0; i--)
{ TotalHeadMovement += abs(RQ[i] - initial);
  initial = RQ[i];
}
for (i = n-1; i >= index; i--)
{ TotalHeadMovement += abs(RQ[i] - initial);
  initial = RQ[i];
  }
}
printf(" Total Head Movement : %d", TotalHeadMovement);
return 0;
}
```

→

PAGE NO :
DATE :

Output :

Enter the no of Requests : 3

Enter the Requests Sequence : 2 1 0

Enter initial head position : 1

Enter total disk size : 3

Enter head movement direction for high-1 / for low-0 : 1

Total Head Movement : 4