WEEK 9-10

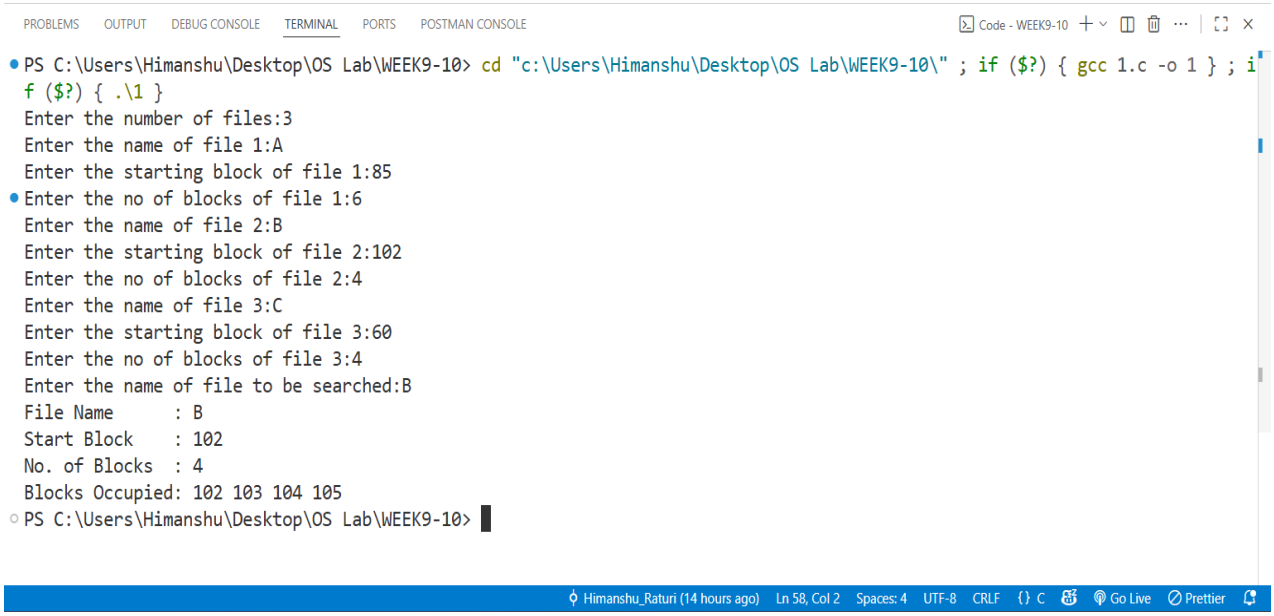
Program1) Write a program to implement Sequential file allocation strategies.

Source Code:

```
#include <stdio.h>
#include <stdbool.h>
typedef struct file
{
   char name;
   int start_block;
   int no_of_blocks;
} file;
void main()
{
   bool blocks[1000] = {true};
   int n;
   printf("Enter the number of files:");
   scanf("%d", &n);
   file files[n];
   for (int i = 0; i < n; i++)
   {
      getchar();
      printf("Enter the name of file %d:", i + 1);
      scanf("%c", &files[i].name);
      printf("Enter the starting block of file %d:", i + 1);
      scanf("%d", &files[i].start_block);
      printf("Enter the no of blocks of file %d:", i + 1);
      scanf("%d", &files[i].no_of_blocks);
      int st = files[i].start_block;
      for (int j = 0; j < files[i].no_of_blocks; j++)
```

```
        {
            blocks[st++] = false;
        }
    }
    char ch;
    getchar();
    printf("Enter the name of file to be searched:");
    scanf("%c", &ch);
    bool found = false;
    for (int i = 0; i < n; i++)
    {
        if (files[i].name == ch)
        {
            printf("File Name      : %c\n", files[i].name);
            printf("Start Block    : %d\n", files[i].start_block);
            printf("No. of Blocks  : %d\n", files[i].no_of_blocks);
            printf("Blocks Occupied: ");
            int st = files[i].start_block;
            for (int j = 0; j < files[i].no_of_blocks; j++)
            {
                printf("%d ", st++);
            }
            found = true;
            break;
        }
    }
    if (!found)
        printf("File not found");
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
Code - WEEK9-10
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK9-10> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK9-10\" ; if ($?) { gcc 1.c -o 1 } ; if ($?) { .\1 }
Enter the number of files:3
Enter the name of file 1:A
Enter the starting block of file 1:85
Enter the no of blocks of file 1:6
Enter the name of file 2:B
Enter the starting block of file 2:102
Enter the no of blocks of file 2:4
Enter the name of file 3:C
Enter the starting block of file 3:60
Enter the no of blocks of file 3:4
Enter the name of file to be searched:B
File Name : B
Start Block : 102
No. of Blocks : 4
Blocks Occupied: 102 103 104 105
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK9-10>
Himanshu_Raturi (14 hours ago)
Ln 58, Col 2
Spaces: 4
UTF-8
CRLF
{} C
Go Live
Prettier

Program2) Write a program to implement Linked file allocation strategies.

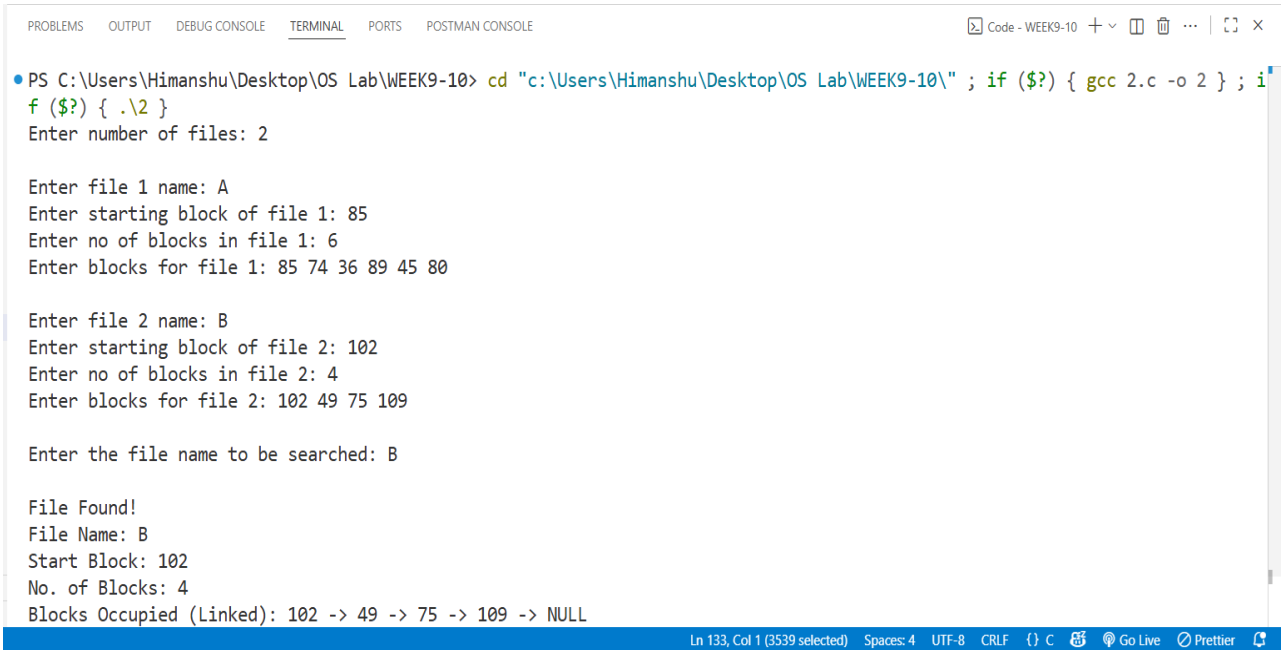
Source Code:

```
#include <stdio.h>
#include <stdbool.h>
typedef struct file{
   char name;
   int start_block;
   int blocks[100];
   int no_of_blocks;
} file;
void main(){
    bool blocks[1000];
   for (int i = 0; i < 1000; ++i)
      blocks[i] = true; // true means free
   int n;
   printf("Enter number of files: ");
   if (scanf("%d", &n) != 1 || n <= 0){
      printf("Invalid number of files.\n");
      return 1;
   }
   file files[n];
   for (int i = 0; i < n; i++)    {
      printf("\nEnter file %d name: ", i + 1);
      if (scanf(" %c", &files[i].name) != 1) {
         printf("Invalid file name input.\n");
         return 1;
      }
      printf("Enter starting block of file %d: ", i + 1);
      if (scanf("%d", &files[i].start_block) != 1)   {
```

```
        printf("Invalid start block input.\n");
        return 1;
    }
    printf("Enter no of blocks in file %d: ", i + 1);
    if (scanf("%d", &files[i].no_of_blocks) != 1 ||
        files[i].no_of_blocks < 0 ||
        files[i].no_of_blocks > 100)  {
        printf("Invalid number of blocks (0..%d).\n", 100);
        return 1;
    }
    if (files[i].no_of_blocks == 0)       {
        printf("No blocks to read for file %c.\n", files[i].name);
        continue;
    }
    printf("Enter blocks for file %d: ", i + 1);
    for (int j = 0; j < files[i].no_of_blocks; j++)       {
        int b;
        while (1)   {
            if (scanf("%d", &b) != 1)             {
                int c;
                while ((c = getchar()) != EOF && c != '\n')
                printf("Invalid input. Enter a valid block number: ");
                continue;
            }
            if (b < 0 || b >= 1000)   {
                printf("Block %d out of range (0..%d). Enter another block: ", b, 1000 - 1);
                continue;
            }
            if (blocks[b] == false)   {
                printf("Block %d already occupied, enter another block: ", b);
```

```
                continue;
            }
            files[i].blocks[j] = b;
            blocks[b] = false; // mark occupied
            break;
        }  } }
    char ch;
    printf("\nEnter the file name to be searched: ");
    if (scanf(" %c", &ch) != 1)    {
        printf("Invalid input.\n");
        return 1;
    }
    bool found = false;
    for (int i = 0; i < n; i++)    {
        if (files[i].name == ch)       {
            printf("\nFile Found!\n");
            printf("File Name: %c\n", files[i].name);
            printf("Start Block: %d\n", files[i].start_block);
            printf("No. of Blocks: %d\n", files[i].no_of_blocks);
            printf("Blocks Occupied (Linked): ");
            for (int j = 0; j < files[i].no_of_blocks; j++)  {
                printf("%d", files[i].blocks[j]);
          }
            printf(" -> NULL\n");
            found = true;
            break;
        }  }
    if (!found)
        printf("\nFile not found\n");
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
Code - WEEK9-10
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK9-10> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK9-10\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Enter number of files: 2
Enter file 1 name: A
Enter starting block of file 1: 85
Enter no of blocks in file 1: 6
Enter blocks for file 1: 85 74 36 89 45 80
Enter file 2 name: B
Enter starting block of file 2: 102
Enter no of blocks in file 2: 4
Enter blocks for file 2: 102 49 75 109
Enter the file name to be searched: B
File Found!
File Name: B
Start Block: 102
No. of Blocks: 4
Blocks Occupied (Linked): 102 -> 49 -> 75 -> 109 -> NULL
Ln 133, Col 1 (3539 selected)
Spaces: 4
UTF-8
CRLF
{} C
Go Live
Prettier

Program3) Write a program to implement Indexed file allocation strategies.

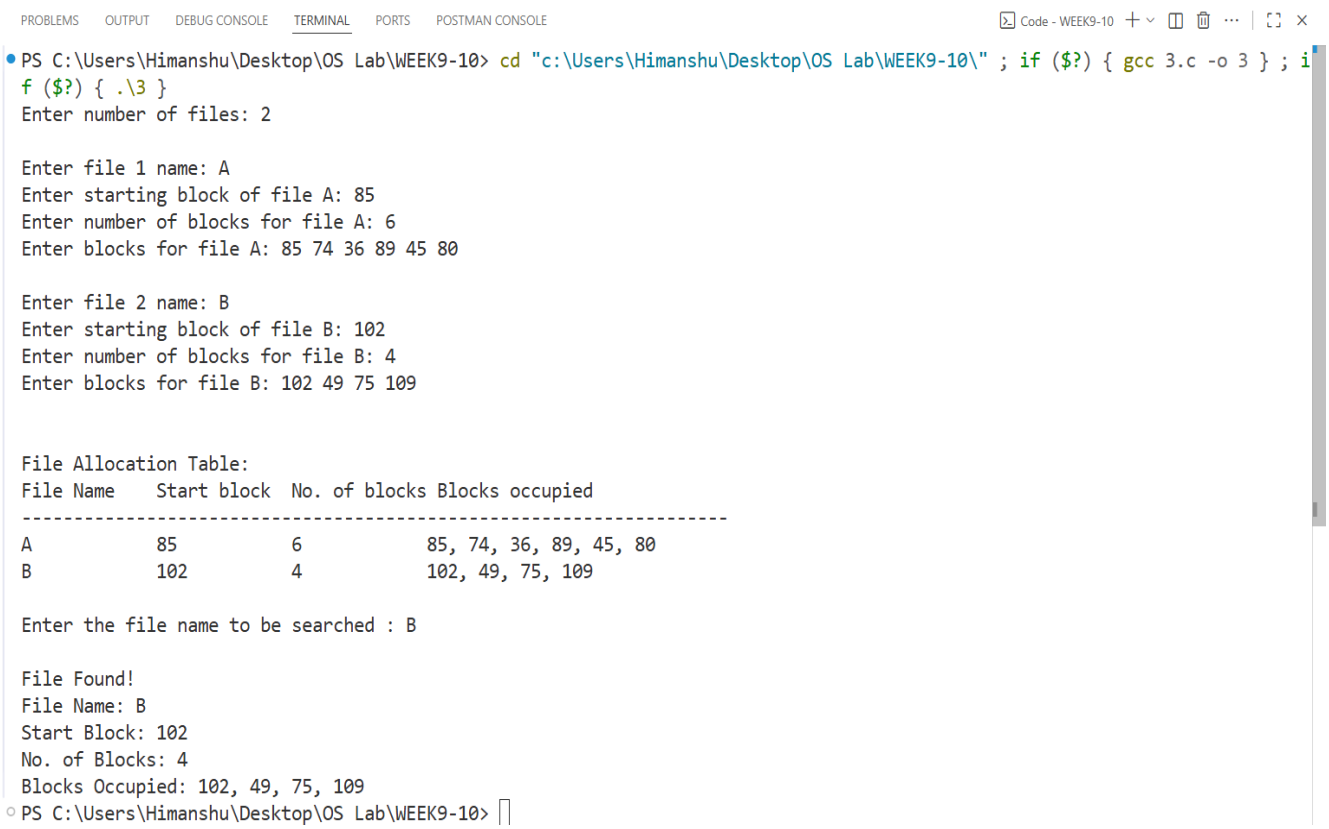
Source Code:

```
#include <stdio.h>
#include <stdbool.h>
struct file {
   char name;
   int start_block;
   int no_of_blocks;
   int flag;
};
struct block {
   int current;
   int next;
};
void main() {
   int n;
   printf("Enter number of files: ");
   scanf("%d", &n);
   struct file files[n];
   struct block sector[1000];
   for (int i = 0; i < 1000; i++)  {
      if (i % 2 == 0) {
         sector[i].current = 100;
         sector[i].next = 100;
      } else {
         sector[i].current = -1;
         sector[i].next = -1;
      } }
   for (int i = 0; i < n; i++)  {
```

```
    getchar();
    printf("\nEnter file %d name: ", i + 1);
    scanf("%c", &files[i].name);
    printf("Enter starting block of file %c: ", files[i].name);
    scanf("%d", &files[i].start_block);
    printf("Enter number of blocks for file %c: ", files[i].name);
    scanf("%d", &files[i].no_of_blocks);
  }
  int p = 0;
  for (int i = 0; i < n; i++)   {
    int req = files[i].no_of_blocks;
    int start_block = files[i].start_block;
    int index = -1;
    if (sector[start_block].current == -1)  {
      files[i].flag = 1;
      for (int j = start_block; j < 1000 && req > 0; j++) {
        if (sector[j].current == -1) {
          sector[j].current = p++; // assign a block number
          req--;
          if (index == -1)
            files[i].start_block = j;
          else
            sector[index].next = j;
          index = j;
        }
      }
    }
    else  {
      printf("enter correct starting block\n");
      files[i].flag = 0;
```

```
        }
    }
    printf("\n\nFile Allocation Table:\n");
    printf("File\tStart\tBlocks Linked\n");
    printf("---------------------------------\n");
    for (int i = 0; i < n; i++)    {
        if (!files[i].flag)
            continue;
        printf("%c\t%d\t", files[i].name, files[i].start_block);
        int j = files[i].start_block;
        while (j != -1)       {
            printf("%d ", j);
            j = sector[j].next;
        }
        printf("\n");
    }
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
Code - WEEK9-10
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK9-10> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK9-10\" ; if ($?) { gcc 3.c -o 3 } ; if ($?) { .\3 }
Enter number of files: 2
Enter file 1 name: A
Enter starting block of file A: 85
Enter number of blocks for file A: 6
Enter blocks for file A: 85 74 36 89 45 80
Enter file 2 name: B
Enter starting block of file B: 102
Enter number of blocks for file B: 4
Enter blocks for file B: 102 49 75 109
File Allocation Table:
File Name Start block No. of blocks Blocks occupied
A 85 6 85, 74, 36, 89, 45, 80
B 102 4 102, 49, 75, 109
Enter the file name to be searched : B
File Found!
File Name: B
Start Block: 102
No. of Blocks: 4
Blocks Occupied: 102, 49, 75, 109
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK9-10>

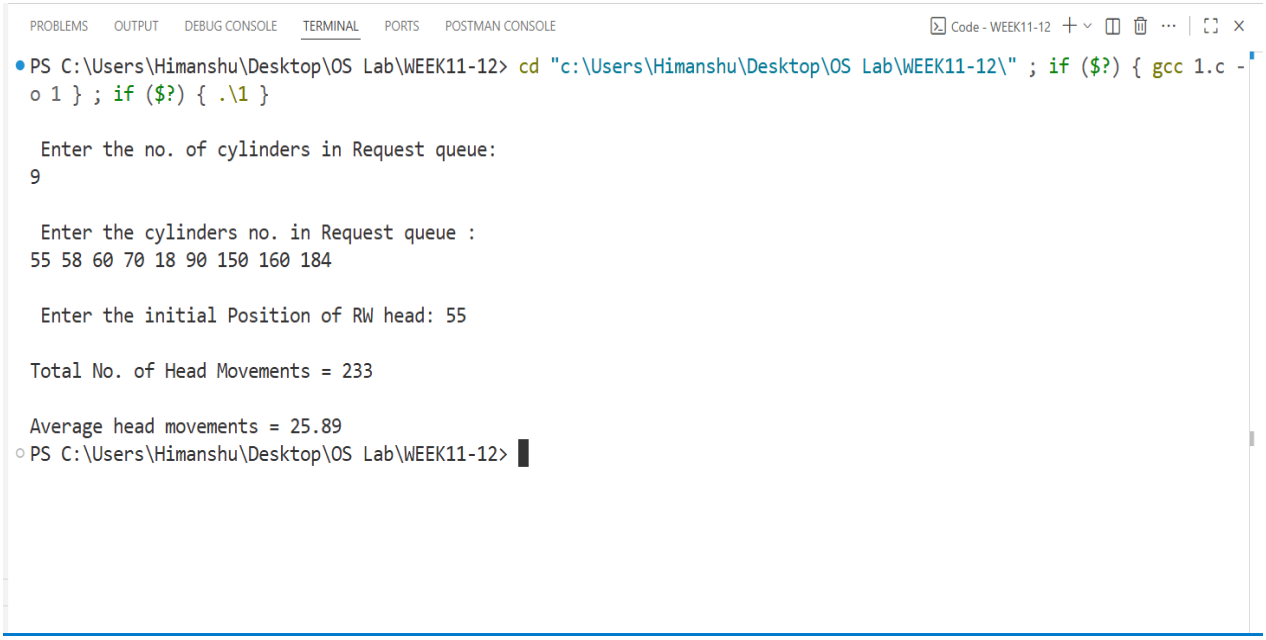
WEEK 11-12

Program1) Write a program to implement FCFS Disc Scheduling Algorithms.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
   int total_head_movement = 0, initial_pos, n;
   printf("\n Enter the no. of cylinders in Request queue:\n");
   scanf("%d", &n);
   int request_queue[n];
   printf("\n Enter the cylinders no. in Request queue :\n");
   for (int i = 0; i < n; i++)
      scanf("%d", &request_queue[i]);
   printf("\n Enter the initial Position of RW head: ");
   scanf("%d", &initial_pos);
   for (int i = 0; i < n; i++)
   {
      total_head_movement += abs(initial_pos - request_queue[i]);
      initial_pos = request_queue[i];
   }
   printf("\nTotal No. of Head Movements = %d\n", total_head_movement);
   printf("\nAverage head movements = %.2f\n", (float)total_head_movement / n);
   return 0;
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
Code - WEEK11-12
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK11-12\" ; if ($?) { gcc 1.c -o 1 } ; if ($?) { .\1 }
Enter the no. of cylinders in Request queue:
9
Enter the cylinders no. in Request queue :
55 58 60 70 18 90 150 160 184
Enter the initial Position of RW head: 55
Total No. of Head Movements = 233
Average head movements = 25.89
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12>

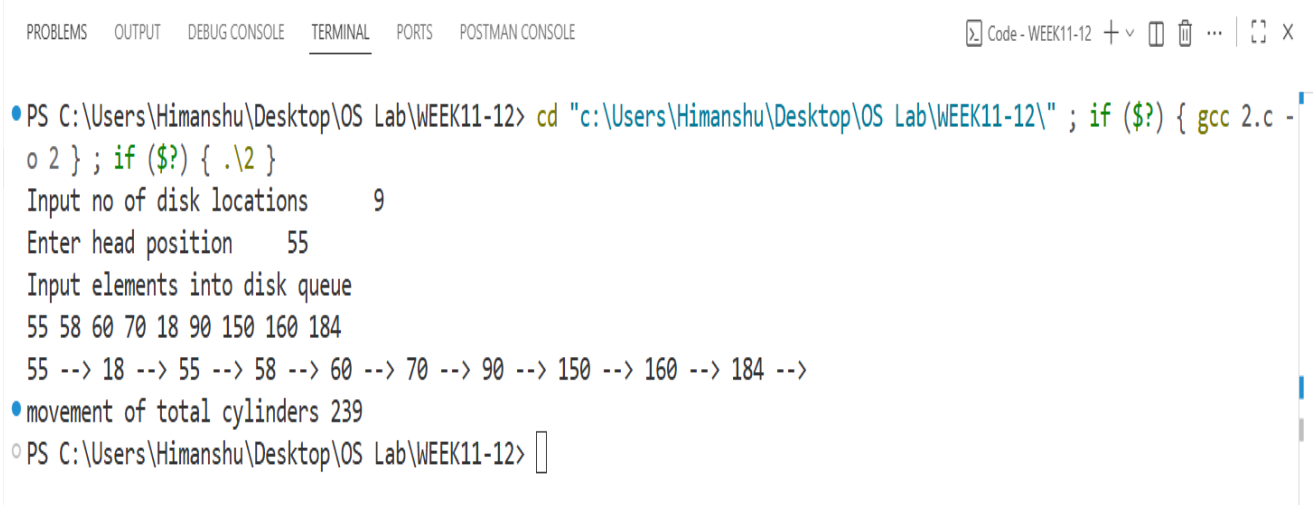
Program2) Write a program to implement SCAN Disc Scheduling Algorithms.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#define LOW 0
#define HIGH 199
Void main() {
   int queue[20];
   int head, max, q_size, temp, sum;
   int dloc; // location of disk (head) arr
   printf("%s\t", "Input no of disk locations");
   scanf("%d", &q_size);
   printf("%s\t", "Enter head position");
   scanf("%d", &head);
   printf("%s\n", "Input elements into disk queue");
   for (int i = 0; i < q_size; i++)
      scanf("%d", &queue[i]);
   queue[q_size] = head; // add RW head into queue
   q_size++;
   for (int i = 0; i < q_size; i++)    {
      for (int j = i; j < q_size; j++) {
         if (queue[i] > queue[j])  {
            temp = queue[i];
            queue[i] = queue[j];
            queue[j] = temp;
         }
      }
   }
   max = queue[q_size - 1];
```

```
    for (int i = 0; i < q_size; i++) {
        if (head == queue[i]) {
            dloc = i;
            break;
        }
    }
    if (abs(head - LOW) <= abs(head - HIGH))   {
        for (int j = dloc; j >= 0; j--)
            printf("%d --> ", queue[j]);
        for (int j = dloc + 1; j < q_size; j++)
            printf("%d --> ", queue[j]);
    } else
    {
        for (int j = dloc + 1; j < q_size; j++)
        {
            printf("%d --> ", queue[j]);
        }
        for (int j = dloc; j >= 0; j--)
        {
            printf("%d --> ", queue[j]);
        }
    }
    sum = head + max;
    printf("\nmovement of total cylinders %d", sum);
    return 0;
}
```

OUTPUT


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POSTMAN CONSOLE
Code - WEEK11-12
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK11-12\" ; if ($?) { gcc 2.c -o 2 } ; if ($?) { .\2 }
Input no of disk locations 9
Enter head position 55
Input elements into disk queue
55 58 60 70 18 90 150 160 184
55 --> 18 --> 55 --> 58 --> 60 --> 70 --> 90 --> 150 --> 160 --> 184 -->
movement of total cylinders 239
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12>

Program3) Write a program to implement C-SCAN Disc Scheduling Algorithms.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
int cmp(const void *a, const void *b) {
   int x = *(int *)a, y = *(int *)b;
   return (x > y) - (x < y);
}
void main() {
   const int START = 0, END = 199;
   int n;
   printf("Enter number of disk requests : ");
   if (scanf("%d", &n) != 1 || n < 0)
      return 0;
   int *req = (int *)malloc(sizeof(int) * n);
   for (int i = 0; i < n; ++i)
      scanf("%d", &req[i]);
   int head;
   printf("Enter initial head position : ");
   if (scanf("%d", &head) != 1)
      head = 0;
   char dir[16];
   printf("Enter initial direction (left/right) : ");
   scanf("%15s", dir);
   qsort(req, n, sizeof(int), cmp);
   int leftCount = 0, rightCount = 0;
   for (int i = 0; i < n; ++i)  {
      if (req[i] < head)
         leftCount++;
```

```
    else
        rightCount++;
} int *left = (int *)malloc(sizeof(int) * leftCount);
int *right = (int *)malloc(sizeof(int) * rightCount);
int li = 0, ri = 0;
for (int i = 0; i < n; ++i)  {
    if (req[i] < head)
        left[li++] = req[i];
    else
        right[ri++] = req[i];
}
long total_seek = 0;
int current = head;
if (dir[0] == 'l' || dir[0] == 'L')  {
    for (int i = leftCount - 1; i >= 0; --i)  {
        total_seek += abs(current - left[i]);
        current = left[i];
    }
    if (current != START) {
        total_seek += abs(current - START);
        current = START;
    }
    total_seek += abs(END - START);
    current = END;
    for (int i = rightCount - 1; i >= 0; --i)       {
        total_seek += abs(current - right[i]);
        current = right[i];
    }
}   else   {
    for (int i = 0; i < rightCount; ++i)  {
```

```
            total_seek += abs(current - right[i]);
            current = right[i];
        }
        if (current != END)
        {
            total_seek += abs(current - END);
            current = END;
        }
        // jump from end to start (counted)
        total_seek += abs(END - START);
        current = START;
        // service left in ascending order
        for (int i = 0; i < leftCount; ++i)
        {
            total_seek += abs(current - left[i]);
            current = left[i];
        }
    }
    printf("Total seek movement : %ld\n", total_seek);
    free(req);
    free(left);
    free(right);
    return 0;
}
```

OUTPUT

```
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12> cd "c:\Users\Himanshu\Desktop\OS Lab\WEEK11-12\" ; if ($?) { gcc 3.c -
o 3 } ; if ($?) { .\3 }
Enter number of disk requests : 9
55 58 60 70 18 90 150 160 184
Enter initial head position : 55
Enter initial direction (left/right) : right
Total seek movement : 361
PS C:\Users\Himanshu\Desktop\OS Lab\WEEK11-12>
```