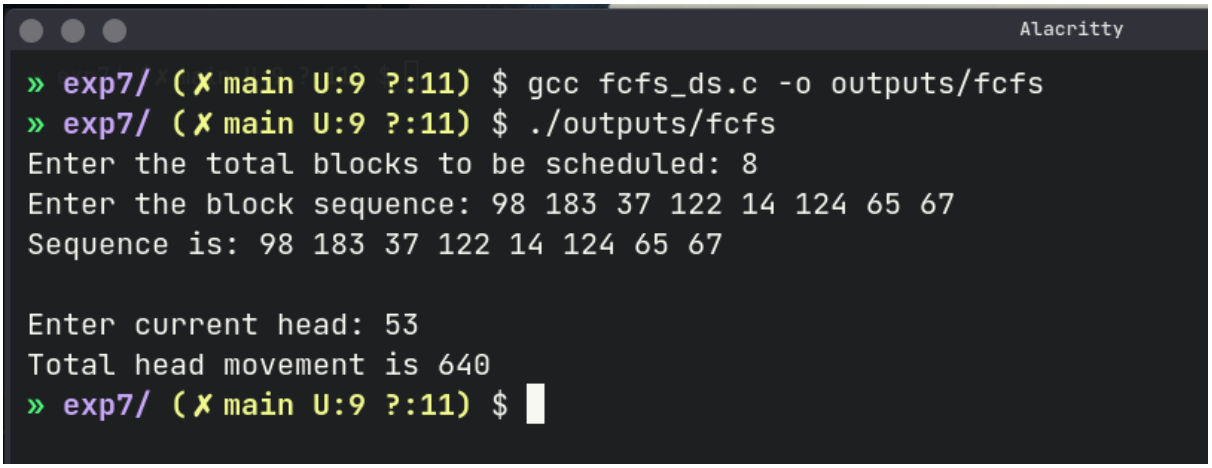
1] FCFS Disk Scheduling Algorithm

```
#include <stdio.h>

int min();
int max();
void fcfs();

#define MAX 100

int main()
{
    int sequence[MAX], total_blocks, i, j;

    printf("Enter the total blocks to be scheduled: ");
    scanf("%d", &total_blocks);

    printf("Enter the block sequence: ");
    for (i = 0; i < total_blocks; i++)
    {
        scanf("%d", &sequence[i]);
    }

    printf("Sequence is: ");
    for (i = 0; i < total_blocks; i++)
    {
        printf("%d ", sequence[i]);
    }

    fcfs(sequence, &total_blocks);

    return 0;
}

int min(int a, int b)
{
    return (a > b) ? b : a;
}

int max(int a, int b)
{
    return (a > b) ? a : b;
}

void fcfs(int *sequence, int *total_blocks)
{
    int previous = 0, current = 0, total_head_movement = 0, low, high, i;
```

```
    printf("\n\nEnter current head: ");
    scanf("%d", &previous);

    for (i = 0; i < *total_blocks; i++)
    {
        current = sequence[i];
        low = min(previous, current);
        high = max(previous, current);
        total_head_movement += (high - low);
        previous = sequence[i];
    }

    printf("Total head movement is %d\n", total_head_movement);
}
```

Output:



2] SSTF Disk Scheduling Algorithm

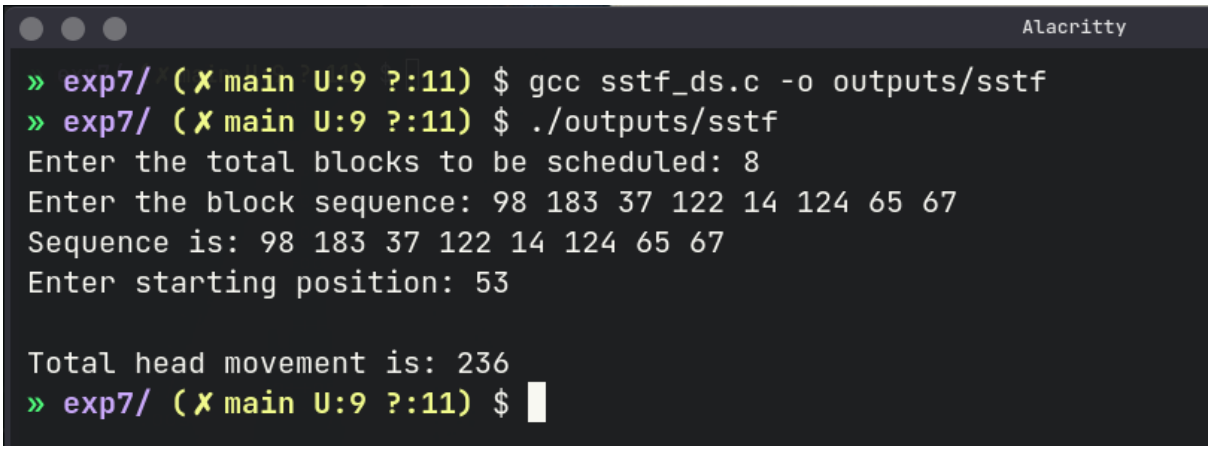
```
#include <stdio.h>
#include <stdlib.h>

void sstf(int *, int *);
int get_shortest_seek(int *, int *, int, int, int);

#define MAX 100

int main()
{
    int sequence[MAX], total_blocks, i, j;

    printf("Enter the total blocks to be scheduled: ");
    scanf("%d", &total_blocks);
```

```
    printf("Enter the block sequence: ");
    for (i = 0; i < total_blocks; i++)
    {
        scanf("%d", &sequence[i]);
    }

    printf("Sequence is: ");
    for (i = 0; i < total_blocks; i++)
    {
        printf("%d ", sequence[i]);
    }

    sstf(sequence, &total_blocks);

    return 0;
}

void sstf(int *sequence, int *total_blocks)
{
    int head_position, current, previous, total_head_movement = 0, temp=0,
index, len_g=0, len_s=0;
    int greater[*total_blocks], smaller[*total_blocks], blocks_covered=0;

    printf("\nEnter starting position: ");
    scanf("%d", &head_position);
    current = head_position;
    previous = current;

    while(blocks_covered =/= *total_blocks)
    {
        temp = 0;
        for (int i = 0; i < *total_blocks; i++)
        {
            if (sequence[i] > current)
            {
                greater[temp] = sequence[i];
                len_g++;
                temp++;
            }
        }
```

```
        temp = 0;
        for (int i = 0; i < *total_blocks; i++)
        {
            if (sequence[i] < current)
            {
                smaller[temp] = sequence[i];
                len_s++;
                temp++;
            }
        }

        current = get_shortest_seek(greater, smaller, current, len_s,
len_g);

        for (int i = 0; i < *total_blocks; i++)
        {
            if (sequence[i] == current)
            {
                sequence[i] = -1;
            }
        }

        total_head_movement += abs(current - previous);
        previous = current;

        blocks_covered++;
        len_s=0;
        len_g=0;
    }

    printf("\nTotal head movement is: %d\n", total_head_movement);
}

int get_shortest_seek(int *greater, int *smaller, int current, int len_s,
int len_g)
{
    int gmin = 1000, smax = -1;


    for (int i = 0; i < len_g; i++)
```

```
    {
        if (greater[i] < gmin && greater[i] =/= -1)
        {
            gmin = greater[i];
        }
    }

    for (int i = 0; i < len_s; i++)
    {
        if (smaller[i] > smax && smaller[i] =/= -1)
        {
            smax = smaller[i];
        }
    }

    if (abs(current - gmin) < abs(current - smax) && gmin =/= -1)
    {
        return gmin;
    }
    else if (abs(current - smax) < abs(current - gmin) && smax =/= -1)
    {
        return smax;
    }

    return (gmin == -1)? smax:gmin;
}
```

Output:



3] SCAN Disk Scheduling Algorithm

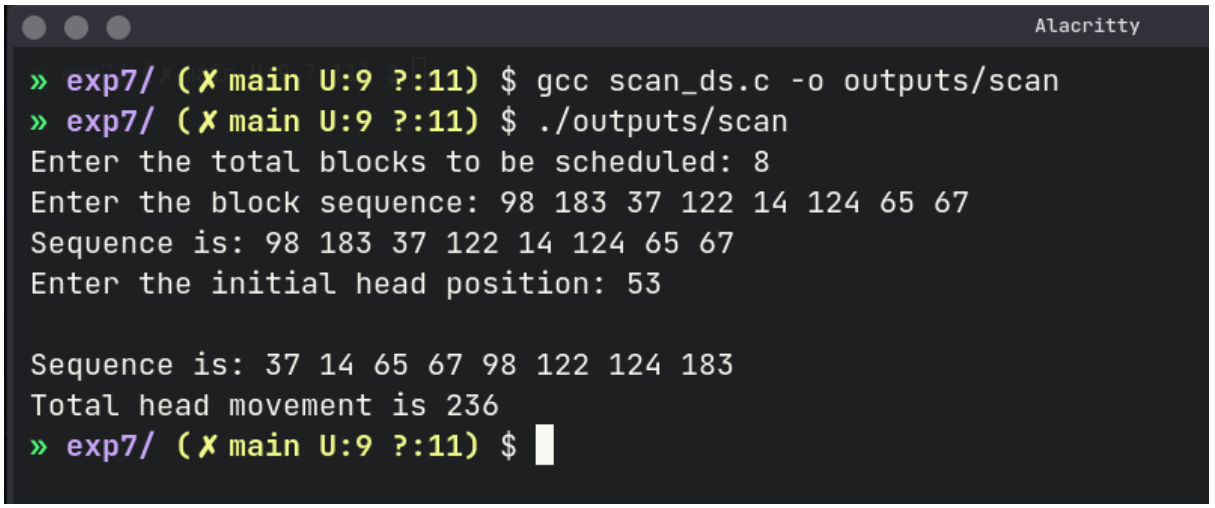
```
#include <stdio.h>
#include <stdlib.h>
```

```
#define MAX 100

int scan(int *, int);
void sort(int *, int);

int main()
{
      int sequence[MAX], total_blocks, i, j;

      printf("Enter the total blocks to be scheduled: ");
      scanf("%d", &total_blocks);

      printf("Enter the block sequence: ");
      for (i = 0; i < total_blocks; i++)
      {
      scanf("%d", &sequence[i]);
      }

      printf("Sequence is: "); // 98 183 37 122 14 124 65 67
      for (i = 0; i < total_blocks; i++)
      {
      printf("%d ", sequence[i]);
      }

      printf("\nTotal head movement is %d", scan(sequence, total_blocks));

      printf("\n");
      return 0;
}

void sort(int *arr, int arr_length)
{
      int i, j, temp;
      for (i = 0; i < arr_length - 1; i++)
      {
      for (j = 0; j < arr_length - i - 1; j++)
      {
            if (arr[j] > arr[j + 1])
            {
                  temp = arr[j];
```

```
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
      }
      }
}

int scan(int *sequence, int total_blocks)
{
      int initial_head, previous_head, current_head, i, j,
total_head_movement = 0;
      int below_head, above_head;
      printf("\nEnter the initial head position: ");
      scanf("%d", &initial_head);

      previous_head = initial_head;
      current_head = initial_head;

      sort(sequence, total_blocks);

      printf("\nSequence is: ");
      for (i = 0; i < total_blocks; i++)
      {
      if (sequence[i] > initial_head)
      {
            for (j = i - 1; j ⩾ 0; j--)
            {
                  printf("%d ", sequence[j]);
                  current_head = sequence[j];
                  total_head_movement += abs(current_head - previous_head);
                  previous_head = current_head;
            }

            // Move till zero
            current_head = 0;
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;

            for (j = i; j < total_blocks; j++)
            {
                  printf("%d ", sequence[j]);
```

```
                current_head = sequence[j];
                total_head_movement += abs(current_head - previous_head);
                previous_head = current_head;
            }
            break;
        }
        }

        return total_head_movement;
}
```

Output:



4] C-SCAN Disk Scheduling Algorithm

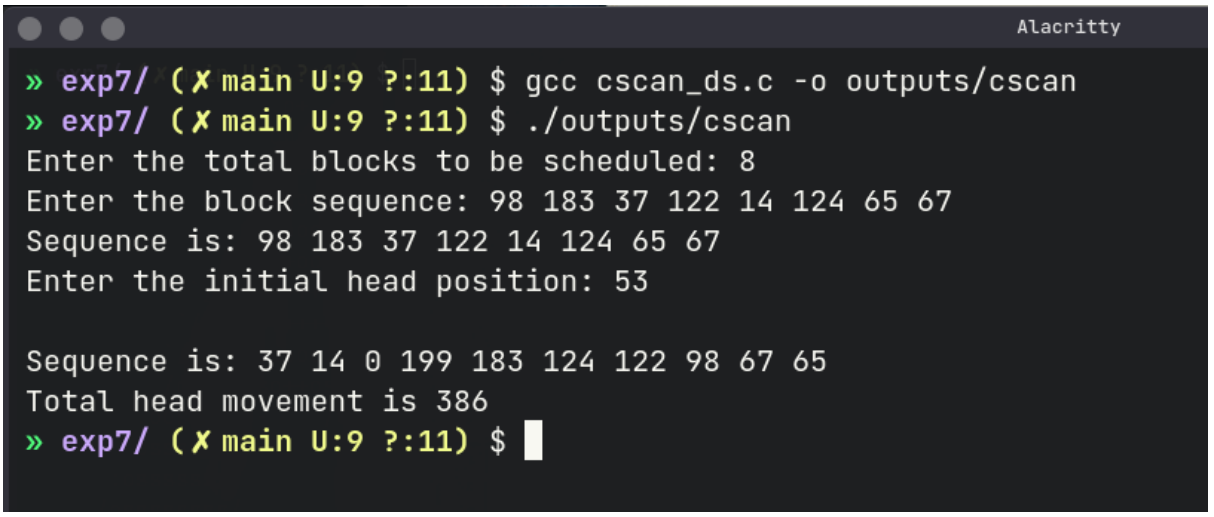
```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100
#define START 0
#define END 199

int cscan(int *, int);
void sort(int *, int);

int main()
{
    int sequence[MAX], total_blocks, i, j;
```

```
	printf("Enter the total blocks to be scheduled: ");
	scanf("%d", &total_blocks);

	printf("Enter the block sequence: ");
	for (i = 0; i < total_blocks; i++)
	{
	scanf("%d", &sequence[i]);
	}

	printf("Sequence is: "); // 98 183 37 122 14 124 65 67
	for (i = 0; i < total_blocks; i++)
	{
	printf("%d ", sequence[i]);
	}

	printf("\nTotal head movement is %d", cscan(sequence, total_blocks));

	printf("\n");
	return 0;
}

void sort(int *arr, int arr_length)
{
	int i, j, temp;
	for (i = 0; i < arr_length - 1; i++)
	{
	for (j = 0; j < arr_length - i - 1; j++)
	{
		if (arr[j] > arr[j + 1])
		{
			temp = arr[j];
			arr[j] = arr[j + 1];
			arr[j + 1] = temp;
		}
	}
	}
}

int cscan(int *sequence, int total_blocks)
{
	int initial_head, previous_head, current_head, i, j;
```

```
int below_head, above_head, total_head_movement = 0;
printf("\nEnter the initial head position: ");
scanf("%d", &initial_head);

previous_head = initial_head;
current_head = initial_head;

sort(sequence, total_blocks);

printf("\nSequence is: ");
for (i = 0; i < total_blocks; i++)
{
if (sequence[i] > initial_head)
{
      for (j = i - 1; j ⩾ 0; j--)
      {
            printf("%d ", sequence[j]);
            current_head = sequence[j];
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;
      }

      // Move till zero
      printf("%d ", START);
      current_head = START;
      total_head_movement += abs(current_head - previous_head);
      previous_head = current_head;

      printf("%d ", END);
      current_head = END;
      total_head_movement += abs(current_head - previous_head);
      previous_head = current_head;

      for (j = total_blocks - 1; j ⩾ i; j--)
      {
            printf("%d ", sequence[j]);
            current_head = sequence[j];
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;
      }
      break;
```

```
    }
    }

    return total_head_movement;
}
```

Output:



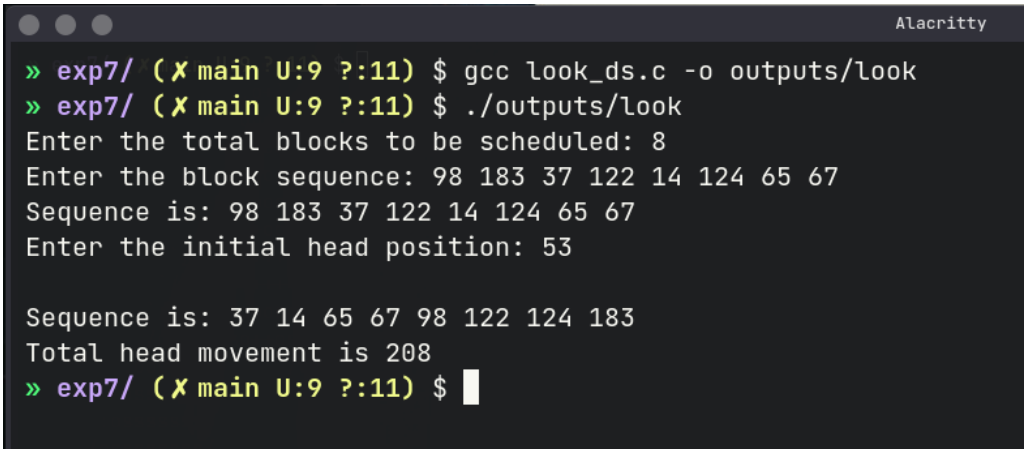
5] LOOK Disk Scheduling Algorithm

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

#define MAX 100

int look(int *, int);
void sort(int *, int);

int main()
{
    int sequence[MAX], total_blocks, i, j;

    printf("Enter the total blocks to be scheduled: ");
    scanf("%d", &total_blocks);

    printf("Enter the block sequence: ");
    for (i = 0; i < total_blocks; i++)
    {
    scanf("%d", &sequence[i]);
```

```
	}

	printf("Sequence is: "); // 98 183 37 122 14 124 65 67
	for (i = 0; i < total_blocks; i++)
	{
	printf("%d ", sequence[i]);
	}

	printf("\nTotal head movement is %d", look(sequence, total_blocks));

	printf("\n");
	return 0;
}

void sort(int *arr, int arr_length)
{
	int i, j, temp;
	for (i = 0; i < arr_length - 1; i++)
	{
	for (j = 0; j < arr_length - i - 1; j++)
	{
		if (arr[j] > arr[j + 1])
		{
			temp = arr[j];
			arr[j] = arr[j + 1];
			arr[j + 1] = temp;
		}
	}
	}
}

int look(int *sequence, int total_blocks)
{
	int initial_head, previous_head, current_head, i, j,
total_head_movement = 0;
	int below_head, above_head;
	printf("\nEnter the initial head position: ");
	scanf("%d", &initial_head);

	previous_head = initial_head;
	current_head = initial_head;
```

```
    sort(sequence, total_blocks);

    printf("\nSequence is: ");
    for (i = 0; i < total_blocks; i++)
    {
    if (sequence[i] > initial_head)
    {
        for (j = i - 1; j >= 0; j--)
        {
            printf("%d ", sequence[j]);
            current_head = sequence[j];
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;
        }

        for (j = i; j < total_blocks; j++)
        {
            printf("%d ", sequence[j]);
            current_head = sequence[j];
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;
        }
        break;
    }
    }

    return total_head_movement;
}
```

Output



6] C-LOOK Disk Scheduling Algorithm

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

#define MAX 100

int clook(int *, int);
void sort(int *, int);

int main()
{
      int sequence[MAX], total_blocks, i, j;

      printf("Enter the total blocks to be scheduled: ");
      scanf("%d", &total_blocks);

      printf("Enter the block sequence: ");
      for (i = 0; i < total_blocks; i++)
      {
      scanf("%d", &sequence[i]);
      }

      printf("Sequence is: "); // 98 183 37 122 14 124 65 67
      for (i = 0; i < total_blocks; i++)
      {
      printf("%d ", sequence[i]);
      }

      printf("\nTotal head movement is %d", clook(sequence, total_blocks));

      printf("\n");
      return 0;
}

void sort(int *arr, int arr_length)
{
      int i, j, temp;
      for (i = 0; i < arr_length - 1; i++)
      {
      for (j = 0; j < arr_length - i - 1; j++)
```

```
      {
            if (arr[j] > arr[j + 1])
            {
                  temp = arr[j];
                  arr[j] = arr[j + 1];
                  arr[j + 1] = temp;
            }
      }
      }
}

int clook(int *sequence, int total_blocks)
{
      int initial_head, previous_head, current_head, i, j,
total_head_movement = 0;
      int below_head, above_head;
      printf("\nEnter the initial head position: ");
      scanf("%d", &initial_head);

      previous_head = initial_head;
      current_head = initial_head;

      sort(sequence, total_blocks);

      printf("\nSequence is: ");
      for (i = 0; i < total_blocks; i++)
      {
      if (sequence[i] > initial_head)
      {
            for (j = i - 1; j ⩾ 0; j--)
            {
                  printf("%d ", sequence[j]);
                  current_head = sequence[j];
                  total_head_movement += abs(current_head - previous_head);
                  previous_head = current_head;
            }

            printf("%d ", sequence[total_blocks - 1]);
            current_head = sequence[total_blocks - 1];
            total_head_movement += abs(current_head - previous_head);
            previous_head = current_head;
```

```
            for (j = total_blocks - 1; j ⩾ i; j--)
            {
                  printf("%d ", sequence[j]);
                  current_head = sequence[j];
                  total_head_movement += abs(current_head - previous_head);
                  previous_head = current_head;
            }
            break;
      }
      }

      return total_head_movement;
}
```

Output: