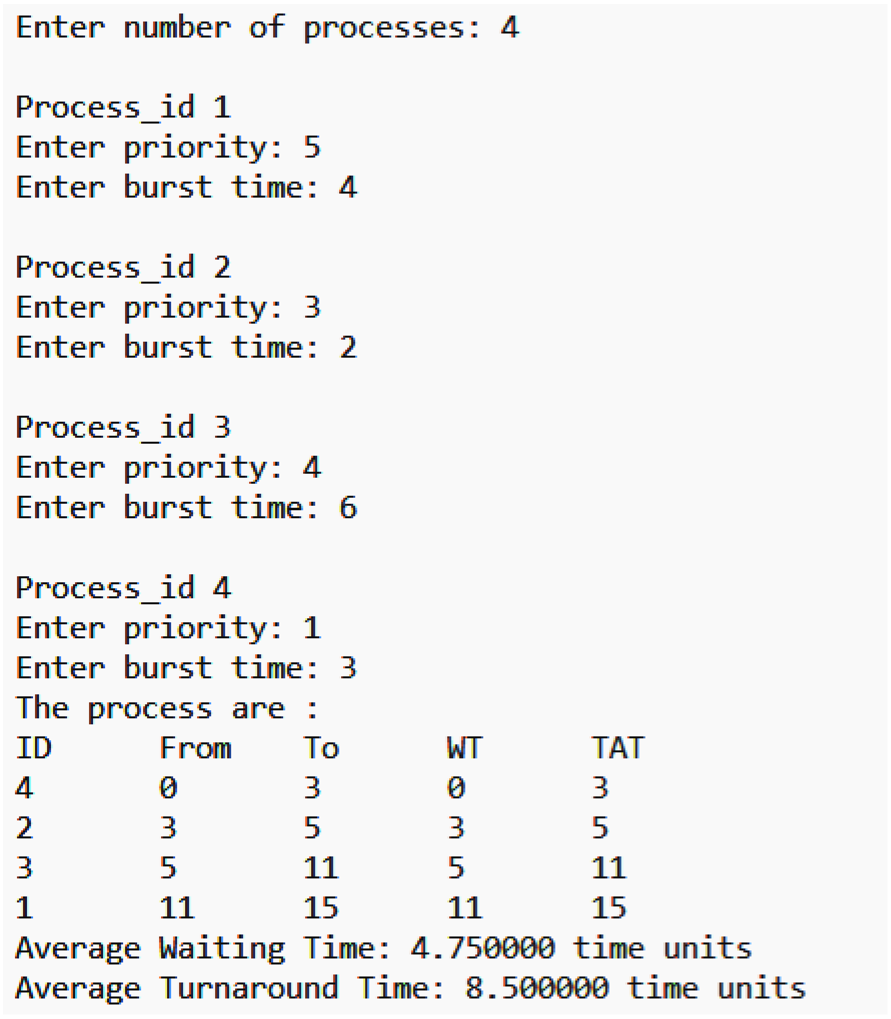
Program :

```
#include <stdio.h>

typedef struct process{
    int process_id;
    int priority;
    int burst_time;
} process;

void swap(process *a, process *b){
    process t = *a;
    *a = *b;
    *b = t;
}

void sort_by_priority(process *arr, int n){
    int swp = 0;
    do{
        for(int i = 1; i < n; i++){
            if(arr[i].priority < arr[i-1].priority){
                swap(&arr[i], &arr[i-1]);
                swp = 1;
            }
        }
        if(!swp) break;
        else swp = 0;
    } while(1);
}

int main(){
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    process arr[n];
    for(int i = 0; i < n; i++){
        arr[i].process_id = i+1;
        printf("\nProcess_id %d\n", i+1);
        printf("Enter priority: ");
        scanf("%d", &arr[i].priority);
        printf("Enter burst time: ");
        scanf("%d", &arr[i].burst_time);
    }
    sort_by_priority(arr, n);
    int wait = 0, turn = 0, tot_wait = 0;
    printf("The process are :\n");
    printf("ID\tFrom\tTo\tWT\tTAT\n");
    for(int i = 0; i < n; i++){
        printf("%d\t%d\t%d\t%d\t%d\n",arr[i].process_id,wait,wait+arr[i].burst_
time,wait, wait+arr[i].burst_time);
        if(i != n-1) {
            tot_wait += wait+arr[i].burst_time;
        }
        turn += wait+arr[i].burst_time;
        wait += arr[i].burst_time;
    }
    printf("Average Waiting Time: %f time units\n", (float)tot_wait/n);
    printf("Average Turnaround Time: %f time units\n", (float)turn/n);
    return 0;
}
```

Output :

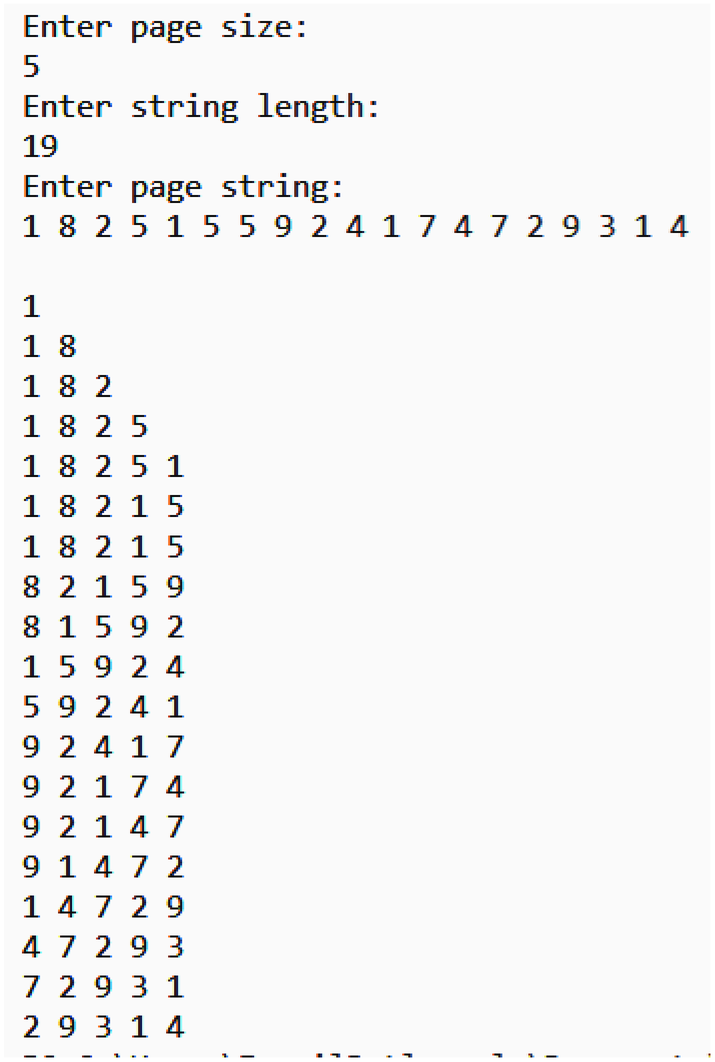


```
Enter number of processes: 4

Process_id 1
Enter priority: 5
Enter burst time: 4

Process_id 2
Enter priority: 3
Enter burst time: 2

Process_id 3
Enter priority: 4
Enter burst time: 6

Process_id 4
Enter priority: 1
Enter burst time: 3
The process are :
ID      From    To      WT      TAT
4       0       3       0       3
2       3       5       3       5
3       5       11      5       11
1       11      15      11      15
Average Waiting Time: 4.750000 time units
Average Turnaround Time: 8.500000 time units
```

Program :

```
#include<bits/stdc++.h>
using namespace std;

int main(void) {
    cout << "Enter page size:\n";
    int p;
    cin >> p;
    vector<int> pages;
    cout << "Enter string length:\n";
    int n;
    cin >> n;
    vector<int> a(n);
    cout << "Enter page string:\n";
    for(int i=0; i<n; i++) {
        cin >> a[i];
    }
    cout << endl;
    for(int i=0; i<n; i++) {
        if(pages.size() != p) {
            pages.push_back(a[i]);
        }
        else {
            int index = 0;
            for(int pos=0; pos<p; pos++) {
                if(pages[pos] == a[i]) {
                    index = pos;
                }
            }
            pages.erase(pages.begin()+index);
            pages.push_back(a[i]);
        }
        for(int itr=0; itr<pages.size(); itr++) {
            cout << pages[itr] << ' ';
        }
        cout << endl;
    }
}
```

Output :

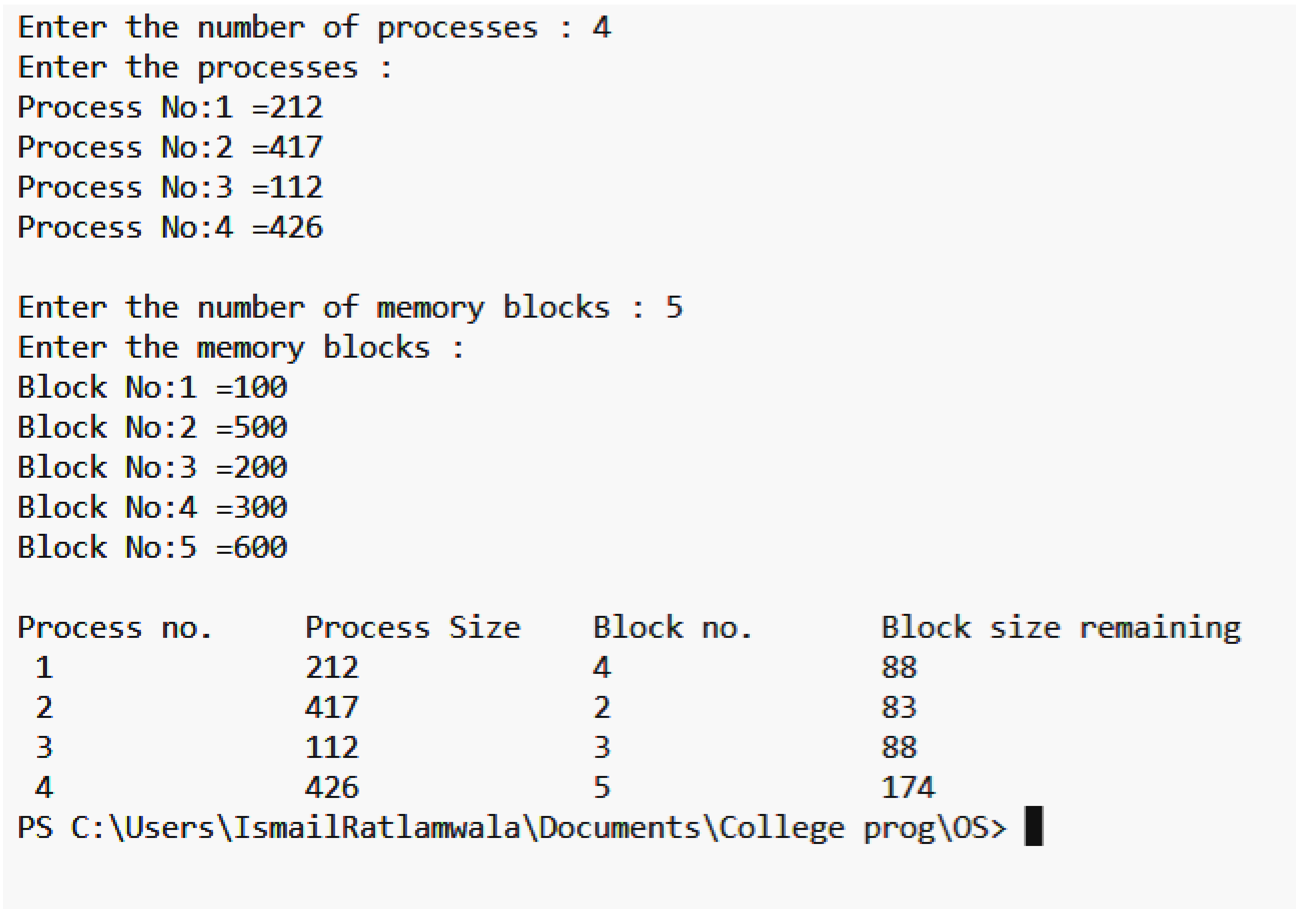


```
Enter page size:
5
Enter string length:
19
Enter page string:
1 8 2 5 1 5 5 9 2 4 1 7 4 7 2 9 3 1 4

1
1 8
1 8 2
1 8 2 5
1 8 2 5 1
1 8 2 1 5
1 8 2 1 5
8 2 1 5 9
8 1 5 9 2
1 5 9 2 4
5 9 2 4 1
9 2 4 1 7
9 2 1 7 4
9 2 1 4 7
9 1 4 7 2
1 4 7 2 9
4 7 2 9 3
7 2 9 3 1
2 9 3 1 4
```

Program :

```
import java.util.Scanner;

public class BestFit{
    static void bestFit(int blockSize[], int m, int processSize[], int n){
        int allocation[] = new int[n]; // Stores block id of the block
allocated to a process

        for (int i = 0; i < allocation.length; i++) // Initially no block is
assigned to any process
        allocation[i] = -1;

        System.out.println("\nProcess no.\tProcess Size\tBlock no.\tBlock size
remaining");
        for (int i=0; i<n; i++){
            int bestIdx = -1;  // Find the best fit block for current process
            boolean allocated = false;
            for (int j=0; j<m; j++){
                if (blockSize[j] >= processSize[i]){
                    if (allocated == false){ //to assign block first time
                        bestIdx = j;
                        allocated=true;
                    }
                    else if (blockSize[bestIdx] > blockSize[j])
                        bestIdx = j;
                }
            }
            if (allocated == true){      //if allocated update allocation list
and reduce corresp block size
                allocation[i] = bestIdx; // allocate block j to p[i] process
                blockSize[bestIdx] -= processSize[i]; // Reduce available
memory in this block.
            }

            System.out.print(" " + (i+1) + "\t\t" + processSize[i] + "\t\t"
);//printing output
            if (allocation[i] != -1)
                System.out.print(allocation[i] + 1 + "\t\t" +
blockSize[allocation[i]]);
            else
                System.out.print("Not Allocated");
            System.out.println();
        }
    }
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of processes : ");
        int n = sc.nextInt();
        int processSize[] = new int[n];
        System.out.println("Enter the processes : ");
        for(int i=0;i<n;i++){
            System.out.print("Process No:"+ (i+1) + " =");
            processSize[i]=sc.nextInt();
        }
        System.out.print("\nEnter the number of memory blocks : ");
        int m = sc.nextInt();
        int blockSize[] = new int[m];
        System.out.println("Enter the memory blocks : ");
        for(int i=0;i<m;i++){
            System.out.print("Block No:"+ (i+1) + " =");
            blockSize[i]=sc.nextInt();
        }
        bestFit(blockSize, m, processSize, n);
        sc.close();
    }
}
```

Output :



```
Enter the number of processes : 4
Enter the processes :
Process No:1 =212
Process No:2 =417
Process No:3 =112
Process No:4 =426

Enter the number of memory blocks : 5
Enter the memory blocks :
Block No:1 =100
Block No:2 =500
Block No:3 =200
Block No:4 =300
Block No:5 =600

Process no.     Process Size    Block no.       Block size remaining
 1              212             4               88
 2              417             2               83
 3              112             3               88
 4              426             5               174
PS C:\Users\IsmailRatlamwala\Documents\College prog\OS> █
```

Program :

```
#include <bits/stdc++.h>
using namespace std;

int bin2dec(char *s, int n){
   int res = 0;
   for(int i = n-1; i >= 0; i--)
       res += (s[i]-'0')*(1<<i);
   return res;
}

int checkTable(int page, int pt[][3]){
   return pt[page][2];
}

int main(){
   int processSize, pageSize, physicalMem;
   cout << "Enter process size in KB:\n";
   cin >> processSize;
   cout << "Enter page size in bytes:\n";
   cin >> pageSize;
   cout << "Enter size of physical memory in MB:\n";
   cin >> physicalMem;

   int frames = (physicalMem*(1<<20))/pageSize;
   printf("\nNo. of frames in memory: %d (i.e. 2^%.0f)\n", frames,
log2(frames));

   int n = (processSize*(1<<10))/pageSize;
   printf("No. of entries in page table: %d (i.e. 2^%.0f)\n", n, log2(n));

   float phy_add_bits = log2(physicalMem*(1<<20));
   printf("No. of bits in physical address: %0.f\n", phy_add_bits);

   float log_add_bits = log2(processSize*(1<<10));
   printf("No. of bits in logical address: %0.f\n", log_add_bits);
   printf("\nPage Segment: %0.f bits\tOffset: %0.f bits\n", log2(n),
log_add_bits-log2(n));

   int pt[n][3];
   cout << "\nInput page table: (Page No | Frame No | Valid Bit)\n";
   for(int i = 0; i < 4; i++) {
       scanf("%d %d %d", &pt[i][0], &pt[i][1], &pt[i][2]);
   }

   int repeat=1;
   while(repeat){
      char logAdd[(int)log_add_bits];
      printf("Input logical address: ");
      scanf("%s", logAdd);
```

```
        int page = bin2dec(logAdd, (int)log2(n));
        printf("%s", (checkTable(page, pt) ? "Page Hit\n":"Page Fault\n"));
        cout<<"\nTo continue enter 1 else 0 : ";
        cin>>repeat;
    }
    return 0;

}
```

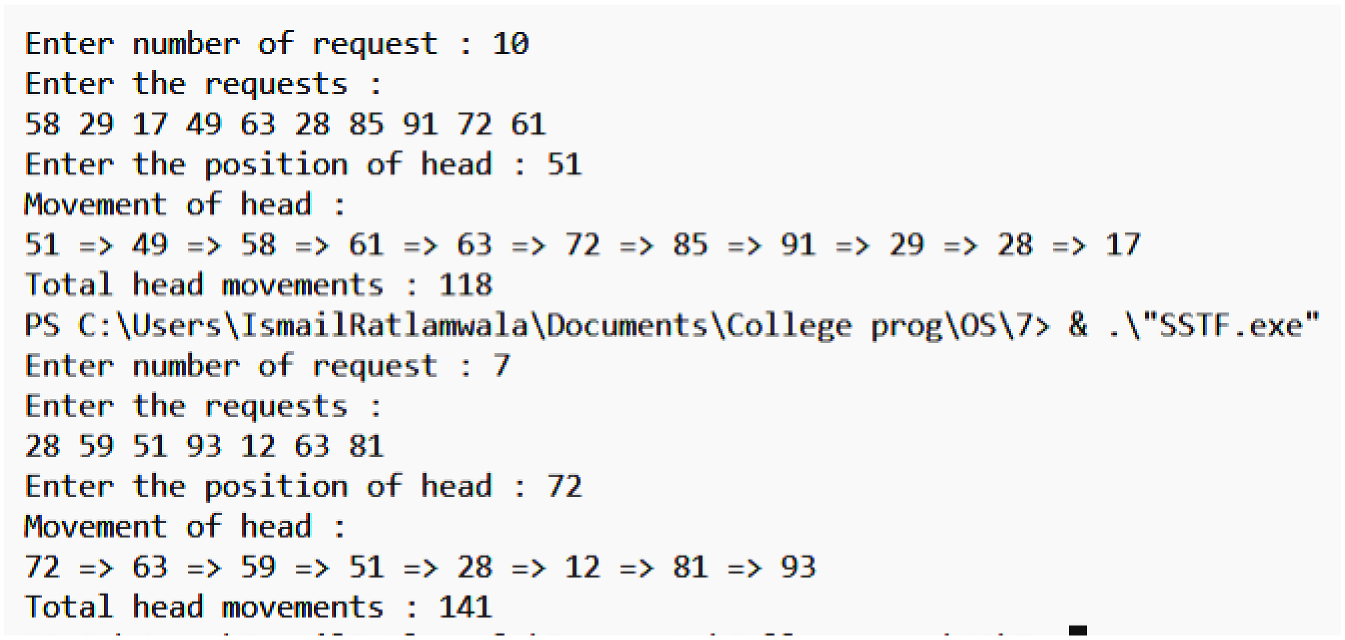
Output :

```
Enter process size in KB:
1
Enter page size in bytes:
256
Enter size of physical memory in MB:
1

No. of frames in memory: 4096 (i.e. 2^12)
No. of entries in page table: 4 (i.e. 2^2)
No. of bits in physical address: 20
No. of bits in logical address: 10

Page Segment: 2 bits    Offset: 8 bits

Input page table: (Page No | Frame No | Valid Bit)
0 4 1
1 2 1
2 3 0
3 1 1
Input logical address: 1110101011
Page Hit

To continue enter 1 else 0 : 1
Input logical address: 1110101010
Page Hit
```

Program :

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    vector<pair<int,bool>> request;
    cout<<"Enter number of request : ";
    int n,inp;
    cin>>n;
    cout<<"Enter the requests :"<<endl;

    for(int i=0;i<n;i++) {
        cin>>inp;
        request.push_back({inp,false});
    }

    cout<<"Enter the position of head : ";
    int head;
    cin>>head;
    int totalMov=0;
    cout<<"Movement of head : \n"<<head;

    for(int j=0;j<n;j++){
        int minDiff=INT_MAX, indx;

        for(int i=0;i<n;i++){
            if(!request[i].second && (abs(head-request[i].first)<minDiff)){
                minDiff=abs(head-request[i].first);
                indx=i;
            }
        }
        cout<<" => "<<request[indx].first;
        head=request[indx].first;
        request[indx].second=true;
        totalMov += minDiff ;
    }
    cout<<"\nTotal head movements : "<<totalMov<<endl;
    return 0;
}
```

Output :

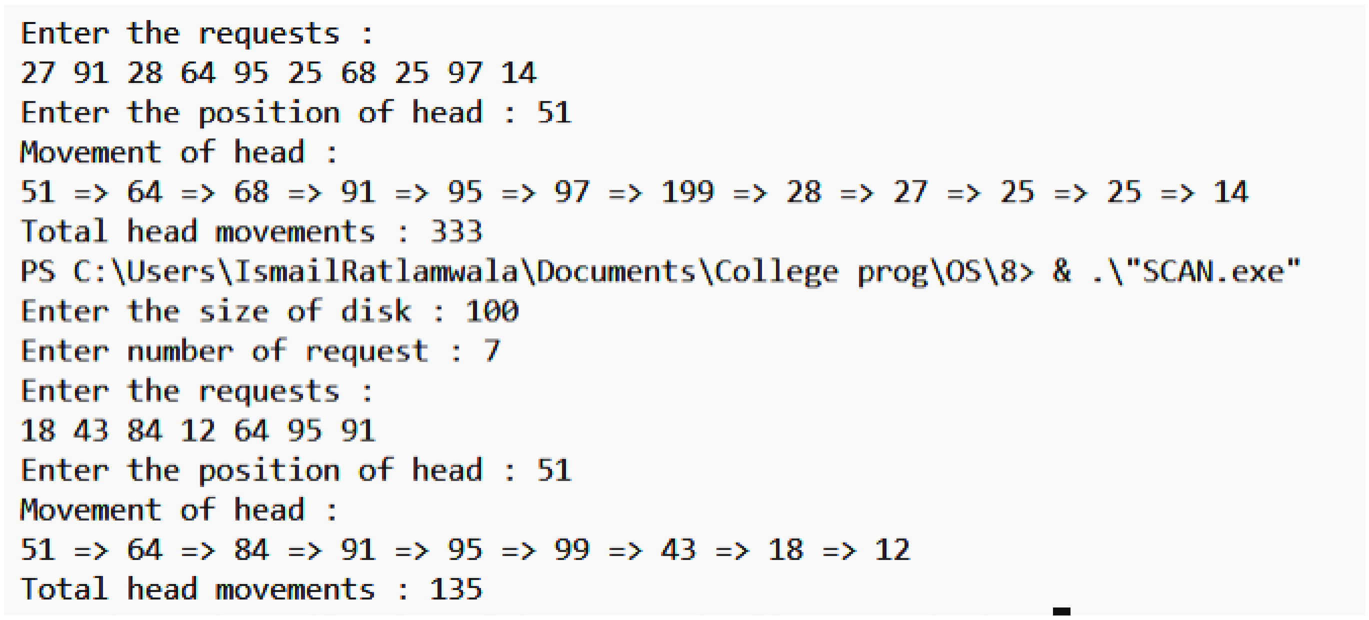


```
Enter number of request : 10
Enter the requests :
58 29 17 49 63 28 85 91 72 61
Enter the position of head : 51
Movement of head :
51 => 49 => 58 => 61 => 63 => 72 => 85 => 91 => 29 => 28 => 17
Total head movements : 118
PS C:\Users\IsmailRatlamwala\Documents\College prog\OS\7> & .\"SSTF.exe"
Enter number of request : 7
Enter the requests :
28 59 51 93 12 63 81
Enter the position of head : 72
Movement of head :
72 => 63 => 59 => 51 => 28 => 12 => 81 => 93
Total head movements : 141
```

Program :

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    vector<int> request;
    cout<<"Enter the size of disk : ";
    int size;
    cin>>size;
    size--;
    cout<<"Enter number of request : ";
    int n;
    cin>>n;
    cout<<"Enter the requests :"<<endl;

    for(int i=0;i<n;i++) {
        int inp;
        cin>>inp;
        request.push_back(inp);
    }
    cout<<"Enter the position of head : ";
    int head;
    cin>>head;
    bool fromLeft=false;
    sort(request.begin(),request.end());
    int cutoffInd;

    for(cutoffInd=0; cutoffInd<n; cutoffInd++)
        if(request[cutoffInd]>head) break;

    cout<<"Movement of head : \n"<<head;
    if(fromLeft){
        for(int i=cutoffInd-1;i>=0;i--) cout<<" => "<<request[i];
        for(int i=cutoffInd; i<n; i++) cout<<" => "<<request[i];

        cout<<"\nTotal head movements : "<<request[n-1]+head;
    }
    else{
        for(int i=cutoffInd; i<n; i++) cout<<" => "<<request[i];
        cout<<" => "<<size;
        for(int i=cutoffInd-1;i>=0;i--) cout<<" => "<<request[i];

        cout<<"\nTotal head movements : "<<2*size-head-request[0];
    }
    return 0;
}
```

Output :

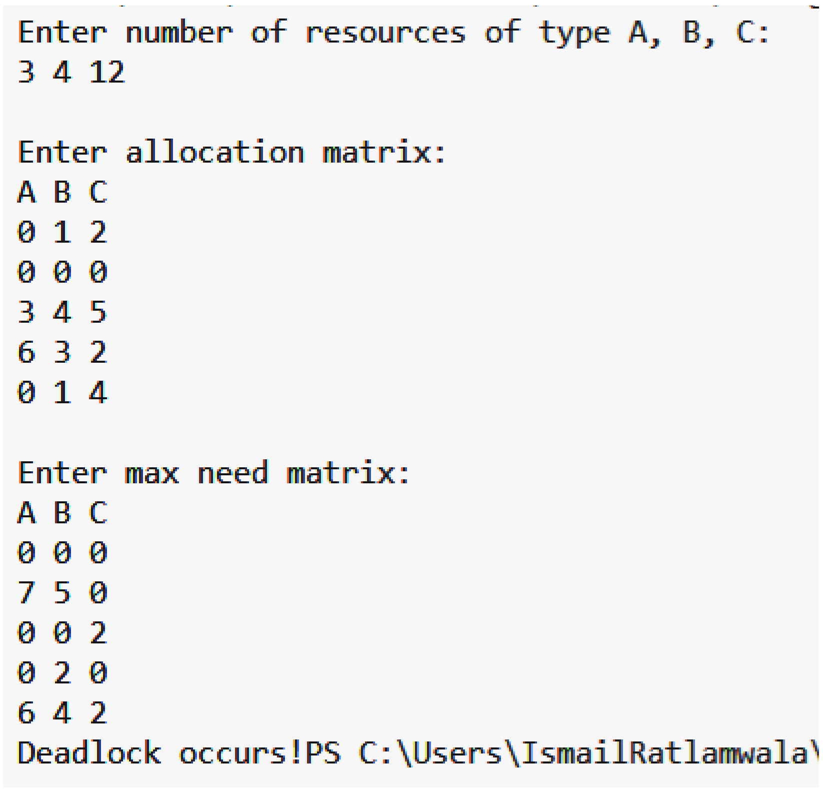


```
Enter the requests :
27 91 28 64 95 25 68 25 97 14
Enter the position of head : 51
Movement of head :
51 => 64 => 68 => 91 => 95 => 97 => 199 => 28 => 27 => 25 => 25 => 14
Total head movements : 333
PS C:\Users\IsmailRatlamwala\Documents\College prog\OS\8> & .\"SCAN.exe"
Enter the size of disk : 100
Enter number of request : 7
Enter the requests :
18 43 84 12 64 95 91
Enter the position of head : 51
Movement of head :
51 => 64 => 84 => 91 => 95 => 99 => 43 => 18 => 12
Total head movements : 135
```

Program :

```
#include <bits/stdc++.h>
using namespace std;

int main(void) {
    vector<vector<int>> allocation(5, vector<int>(3)),
                        maxNeed(5, vector<int>(3)),
                        remNeed(5, vector<int>(3));
    cout << "Enter number of resources of type A, B, C:\n";
    int a, b, c;
    cin >> a >> b >> c;
    cout << "\nEnter allocation matrix:\n";
    cout << "A B C\n";
    vector<int> totalAllocated(3);
    for(int i=0; i<5; i++) {
        for(int j=0; j<3; j++) {
            cin >> allocation[i][j];
            totalAllocated[j] += (allocation[i][j]);
        }
    }
    cout << "\nEnter max need matrix:\n";
    cout << "A B C\n";
    for(int i=0; i<5; i++) {
        for(int j=0; j<3; j++) {
            cin >> maxNeed[i][j];
        }
    }
    for(int i=0; i<5; i++) {
        for(int j=0; j<3; j++) {
            remNeed[i][j] = maxNeed[i][j] - allocation[i][j];
        }
    }
    vector<int> available(3);
    available[0] = a - totalAllocated[0];
    available[1] = b - totalAllocated[1];
    available[2] = c - totalAllocated[2];
    for(int it=0; it<5; it++) {
        bool deadlock = true;
        bool canExecute = true;
        for(int i=0; i<5; i++) {
            bool canExecute = true;
            for(int j=0; j<3; j++) {
                if(remNeed[i][j] > available[j]) {
                    canExecute = false;
                }
            }
            if(canExecute) {
                deadlock = false;
                for(int j=0; j<3; j++) {
                    available[j] += allocation[i][j];
                }
            }
        }
        if(deadlock) {
            cout << "Deadlock occurs!";
            return 0;
        }
    }
    cout << "\nRemaining need matrix:\n";
    for(int i=0; i<5; i++) {
        for(int j=0; j<3; j++) {
            cout << remNeed[i][j] << ' ';
        }
        cout << endl;
    }
}
```

Output :

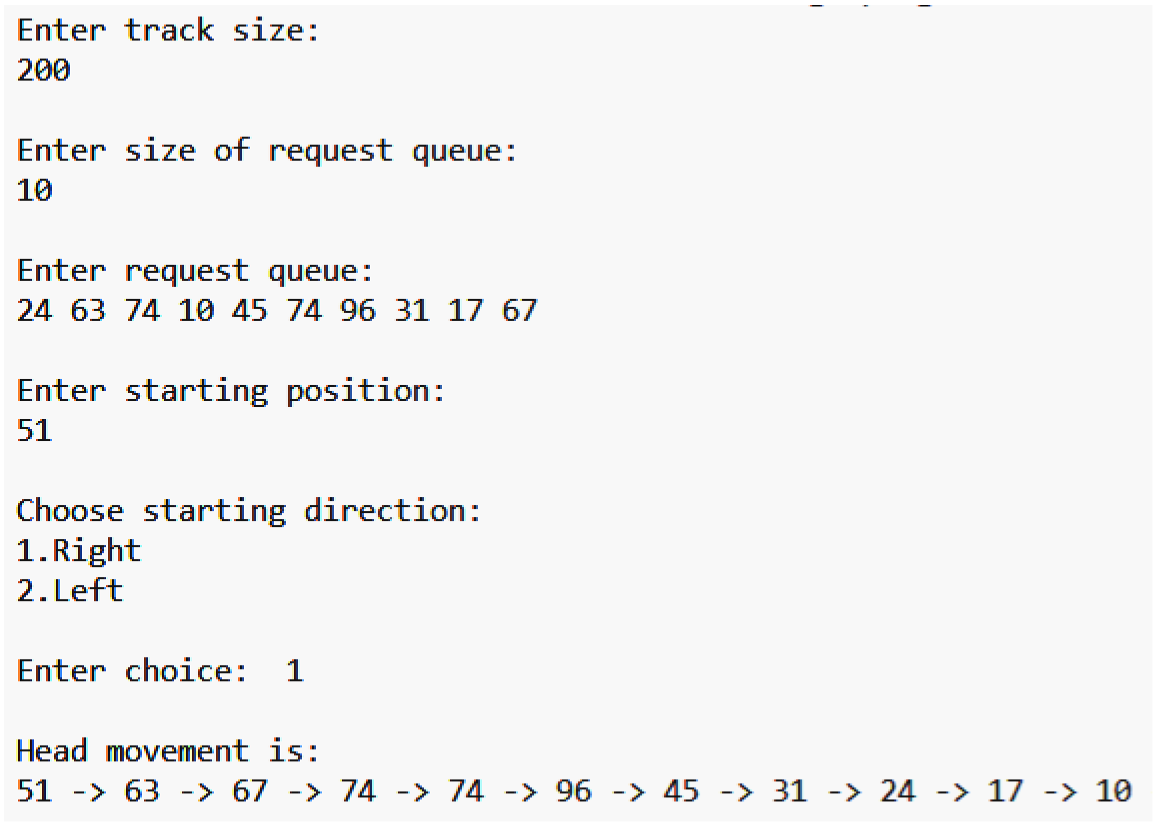


```
Enter number of resources of type A, B, C:
3 4 12

Enter allocation matrix:
A B C
0 1 2
0 0 0
3 4 5
6 3 2
0 1 4

Enter max need matrix:
A B C
0 0 0
7 5 0
0 0 2
0 2 0
6 4 2
Deadlock occurs!PS C:\Users\IsmailRatlamwala\
```

Program :

```
#include <bits/stdc++.h>
using namespace std;

int main(void) {
    cout << "Enter track size:\n";
    int t;
    cin >> t;
    cout << "\nEnter size of request queue:\n";
    int n;
    cin >> n;
    vector<int> a(n);
    cout << "\nEnter request queue:\n";
    for(int i=0; i<n; i++) {
        cin >> a[i];
    }
    sort(a.begin(), a.end());
    cout << "\nEnter starting position:\n";
    int starting;
    cin >> starting;
    cout << "\nChoose starting direction:\n1.Right\n2.Left\n";
    cout << "\nEnter choice:  ";
    int choice;
    cin >> choice;

    int total = 0;
    vector<int> ans;
    ans.push_back(starting);

    if(choice == 1) {
        for(int i=0; i<n; i++) {
            if(a[i] >= starting) {
                ans.push_back(a[i]);
            }
        }
        for(int i=n-1; i>=0; i--) {
            if(a[i] <= starting) {
                ans.push_back(a[i]);
            }
        }
        total += (a[n-1] - starting + a[n-1] - a[0]);
    }

    if(choice == 2) {
        for(int i=n-1; i>=0; i--) {
            if(a[i] <= starting) {
                ans.push_back(a[i]);
            }
        }
        for(int i=0; i<n; i++) {
            if(a[i] >= starting) {
                ans.push_back(a[i]);
            }
        }
        total += (starting - a[0] + a[n-1] - a[0]);
    }

    cout << "\nHead movement is:\n";
    for(int i=0; i<ans.size(); i++) {
        cout << ans[i] << " -> ";
    }
    cout << "Stop\n";
    cout << "\nTotal head movement = " << total;
}
```

Output :



```
Enter track size:
200

Enter size of request queue:
10

Enter request queue:
24 63 74 10 45 74 96 31 17 67

Enter starting position:
51

Choose starting direction:
1.Right
2.Left

Enter choice:  1

Head movement is:
51 -> 63 -> 67 -> 74 -> 74 -> 96 -> 45 -> 31 -> 24 -> 17 -> 10
```