Name: Pariksha Negi University Roll no. 2023075
~~Roll no ——— to~~
Subject: Operating System

Pnegi

Ans 23 #include <stdio.h>
int absolute Value(int); // Declaring function absolute Value

void main()
{
int queue[25], n, headposition, i, j, k, seek = 0, max range,
difference, temp, queue1[20], queue[20], temp1 = 0, temp2 = 0;
float average Seek Time;

// Reading the maximum Range of the Disk.
Print f("Enter the maximum Range of disk :");
Scan f ("%d", & max range);

// Reading the number of Queue Request (Disk access requests)

print f("Enter the number of queue requests:");
scan f ("%d", &n);

// Reading the initial head position (ie. the starting point of execution)

Print f ("Enter the initial head position:");
Scan f ("%d", & headposition);

```
// Reading disk positions to be read in the order of arrival
Printf ("Enter the disk positions to be read (queue):");
for(i=1;i<=n;i++) // Note that i varies from 1 to n
   instead of 0 to n-1
{
Scanf ("%d", &temp); // Reading position value to a
      temporary variable.
// Now if the requested position is greater than
   current headposition,
// then pushing that to array queue1
if (temp > headposition)
{
   queue1 [temp1] = temp; // temp1 is the index
      variable of queue1 []
   temp1++; // incrementing temp 1
}
else // else if temp < current headposition, then push
   to array queue 2[]
{
   queue 2 [temp2] = Temp; // temp2 is the index
   variable of queue 2[]
   temp2++;
}
```

```
}

// Now we have to sort the two arrays
// Starting array queue1[] is in ascending order.

for(i=0;i<temp1-1;i++)
{
  for(j=i+1; j<temp1; j++)
  {
    if (queue1[i] > queue1[j])
    {
      temp = queue[i];
      queue1[i] = queue[j];
      queue1[j] = temp;
    }
  }
}

// SORTING array queue2[] in descending order
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;
      }
    }
}

// Copying first array queue1[] into queue[]
  for(i=1, j=0; j<temp1; i++, j++)
   {
      queue[i] = queue1[j];
   }

// Setting queue[i] to max range because the head goes to
// end of disk and comes back in Scan algorithm
    queue[i] = maxrange;

//Copying second array queue2[] after the first one
is copied into queue[]
for(i=temp1+2, j=0; j<temp2; i++, j++)
{
    queue[i] = queue2[j];
}
```

```
// Setting queue[i] to 0. Because that is the innermost
      cylinder.
   queue[i] = 0.

// At this point, we have the queue [] with the requests in
// Correct order of execution as per scan algo algorithm
// Now we have a to set 0th index of queue [] to be
   the initial headposition.
   queue[0] = headposition;

// Calculating - SEEK TIME. Seek is initially set to 0 in
   the declaration part.
for (j = 0; j < n; j++) // Loop starts from headposition
   (i.e 0th index of queue)
{
   // Find the diff b/w next position and current
      position.
   difference = absolute value (Queue[j+1] - queue[j]);

// Adding difference to the current Seek time value
   Seek = Seek + difference;
```

```
// Display a message to show the movement of disk head.

printf ("Disk head moves from position %d to %d with
        Seek %d \n",
    queue[j], queue[j+1], difference);

}

// Calculating Average Seek time
  averageSeekTime = Seek / (float) n;

// Display Total and Average Seek time(s)
  printf ("Total Seek Time = %d \n", seek);
  printf ("Average Seek Time = %f \n", averageSeek
                                       Time);
}

// Defining function absolute Value
  int absolute Value (int x)
{
  if (x > 0)
  { return x;
  }
```

```
else
{
    return x*-1;
}
}
```

Pnegi