Name : Manasi Rathod

Roll No : 23552

* **CODE**

/\* This program implements the following disck-scheduling algorithms:

FCFS, SSTF, SCAN, C-SCAN, LOOK, and C-LOOK

The program services a disk of 5,000 cylinders numbered 0 to 4999.

It generates a random series of 1000 cylinder requests and services

them according to each of he algorithms listed above. The program

will be passed the initial position of the disk head (as a parameter

on the command line) and report the total amount of head movement

required by each algorithm.

To run:

gcc -o diskAlgorithms diskAlgorithms.c

./file\_name followed by number from 0-4999 for index (see below for example)

./diskAlgorithms 423

\*/

#include <stdio.h>

#include <stdlib.h>

#define CYLINDERS 5000

#define REQUESTS 1000

int start = 0;

int ran\_array[REQUESTS];

int test\_array[REQUESTS];

int\* sort\_array() {

int temp = 0, i = 0, j = 0;

for (i = 0; i < REQUESTS; ++i) {

for (j = i + 1; j < REQUESTS; ++j) {

if (ran\_array[i] > ran\_array[j]) {

temp = ran\_array[i];

ran\_array[i] = ran\_array[j];

ran\_array[j] = temp;

}

}

}

return ran\_array;

}

/\* First-Come-First-Serve (fcfs) starts from the index after the starting

index and continually adds the headmovement from the starting index in

order recieved. If at end of array, start from index zero and continually

add until starting index \*/

int fcfs(int \*ran\_array) {

int i = 0, head\_movement = 0, this\_start = ran\_array[start];

for(i = start; i < REQUESTS; i++) {

head\_movement += abs(ran\_array[i] - this\_start);

}

for(i = 0; i < start; i++) {

head\_movement += abs(this\_start - ran\_array[i]);

}

return head\_movement;

}

/\* Shortest-Seek-Time-First (SSTF) takes the current head position, and

adds the position closest to the current head. This new position now becomes

the head, then this system repeats.

First we sort the array. Then We have counters for above and below start

index that we decrement if used. Once these equal to REQUEST-2 (excluding

start index) we exit. \*/

int sstf(int \* ran\_array) {

ran\_array = sort\_array();

int small\_i = start-1, large\_i = start+1;

int small\_diff = 0, large\_diff = 0;

int head\_movement = 0, total = REQUESTS-2, new\_head = start, head\_value = ran\_array[start];

while(total >= 0) {

small\_diff = abs(ran\_array[new\_head] - ran\_array[small\_i]);

large\_diff = abs(ran\_array[large\_i] - ran\_array[new\_head]);

if(small\_diff < large\_diff) {

head\_movement += small\_diff;

new\_head = small\_i;

small\_i--;

} else {

head\_movement += large\_diff;

new\_head = large\_i;

large\_i++;

}

total--;

}

return head\_movement;

}

/\* SCAN - array is already sorted from sstf. SCAN starts from one left of start,

and continually goes down to zero (if included in randome array or not). Then

starts at one higher than start and continually goes up to highest value (not 5000) \*/

int scan(int \* ranArray) {

int i = 0, curr\_val = 0, sav\_val = ran\_array[start], difference = 0;

int head\_movement = 0, curr\_i = 0;

for(i = start-1; i >= 0; --i) {

curr\_val = ran\_array[i];

difference = abs(sav\_val - curr\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

/\* used to subtract value from zero, or just add same value \*/

head\_movement += sav\_val;

sav\_val = 0;

for(i = start+1; i < REQUESTS; i++) {

curr\_val = ran\_array[i];

difference = abs(curr\_val - sav\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

return head\_movement;

}

/\* Circular Scan (C-SCAN) - start at start index, increase to upper boundary

(even if no value at boundary), save boundary value, go to start boundary

(zero value) increase till last value before start value \*/

int cscan(int \* ranArray) {

int i = 0, curr\_val = 0, sav\_val = ran\_array[start], difference = 0;

int head\_movement = 0, curr\_i = 0, upper\_bound = 4999;

for(i = start+1; i < REQUESTS; i++) {

curr\_val = ran\_array[i];

difference = abs(sav\_val - curr\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

/\* add last val - upper bound, go to and add zero bounday (4999)\*/

head\_movement += upper\_bound - sav\_val;

sav\_val = 0;

head\_movement += 4999;

for(i = 0; i < start; i++) {

curr\_val = ran\_array[i];

difference = abs(curr\_val - sav\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

return head\_movement;

}

/\* Look - start from value above start, increase to highest value.

Then goes to value below start value and decreases until smallest value \*/

int look(int\* ranArray) {

int i = 0, curr\_val = 0, sav\_val = ran\_array[start], difference = 0;

int head\_movement = 0, curr\_i = 0;

for(i = start+1; i < REQUESTS; i++) {

curr\_val = ran\_array[i];

difference = abs(sav\_val - curr\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

for(i = start-1; i >= 0; --i) {

curr\_val = ran\_array[i];

difference = abs(curr\_val - sav\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

return head\_movement;

}

/\* C-Look - Starts from value after start value, goes to highest value,

then goes to smallest value and increases until value before start value \*/

int clook(int\* ranArray) {

int i = 0, curr\_val = 0, sav\_val = ran\_array[start], difference = 0;

int head\_movement = 0, curr\_i = 0;

for(i = start+1; i < REQUESTS; i++) {

curr\_val = ran\_array[i];

difference = abs(sav\_val - curr\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

for(i = 0; i < start; i++) {

curr\_val = ran\_array[i];

difference = abs(curr\_val - sav\_val);

head\_movement += difference;

sav\_val = curr\_val;

}

return head\_movement;

}

int main (int argc, char \*argv[]) {

int i = 0;

start = atoi(argv[1]);

if(argc != 2) {

printf("Please compile program with starting index from 0-4999. Ex. ./diskAlgorithms 423\n");

exit(-1);

}

for(i = 0; i < REQUESTS; i++) {

ran\_array[i] = rand() % 5000;

}

printf("\nStart index: %d, start value: %d\n\n", start, ran\_array[start]);

printf("FCFS head movements: %d\n", fcfs(ran\_array));

printf("SSTF head movements: %d\n", sstf(ran\_array));

printf("SCAN head movements: %d\n", scan(ran\_array));

printf("CSCAN head movements: %d\n", cscan(ran\_array));

printf("LOOK head movements: %d\n", look(ran\_array));

printf("C-LOOK head movements: %d\n\n", clook(ran\_array));

return 0;

}

* **OUTPUT**

