**Name: Vidhi Gulhane**

**Roll No: 17 (D10B) IT**

Q1} To write a c program to implement Paging technique for memory management.

#include <stdio.h>

#include <stdlib.h>

#define PAGE\_SIZE 1024

#define NUM\_FRAMES 4

#define NUM\_PAGES 8

int page\_table[NUM\_PAGES];

void initialize\_page\_table() {

for (int i = 0; i < NUM\_PAGES; i++) {

page\_table[i] = -1; }}

void display\_page\_table() {

printf("\nPage Table:\n");

for (int i = 0; i < NUM\_PAGES; i++) {

printf("Page %d -> Frame %d\n", i, page\_table[i]);}

printf("\n");}

int main() {

initialize\_page\_table();

int logical\_memory[NUM\_PAGES][PAGE\_SIZE];

int physical\_memory[NUM\_FRAMES][PAGE\_SIZE];

for (int i = 0; i < NUM\_PAGES; i++) {

int page\_number = rand() % NUM\_PAGES;

if (page\_table[page\_number] == -1) {

int frame\_to\_replace = i % NUM\_FRAMES;

printf("Page %d is not in memory. Loading into Frame %d\n", page\_number, frame\_to\_replace);

for (int j = 0; j < PAGE\_SIZE; j++) {

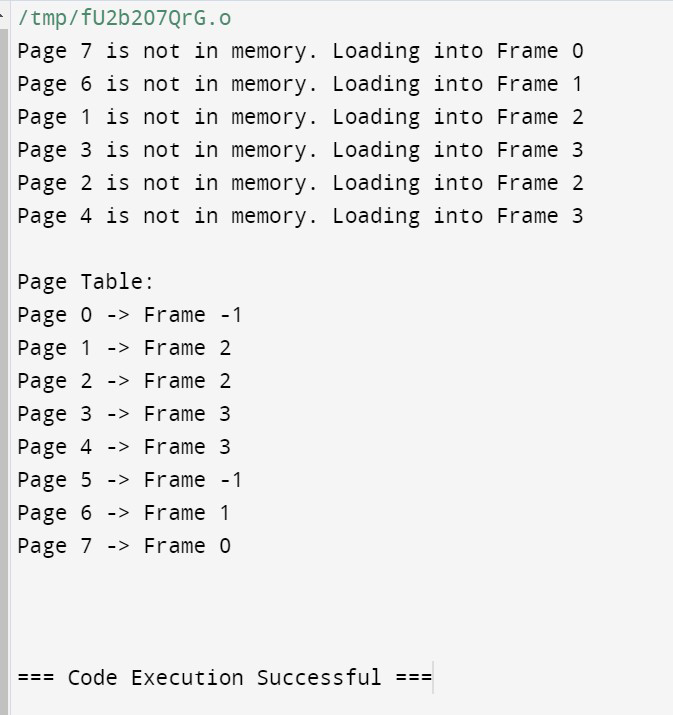
physical\_memory[frame\_to\_replace][j] = logical\_memory[page\_number][j];}

page\_table[page\_number] = frame\_to\_replace;}}

display\_page\_table();

return 0;}

**Output:**



**Q2} Implement various disk scheduling algorithms like SCAN, and C-SCAN in C/Python/Java.**

**SCAN**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define size 10

#define disk\_size 200

int comp(const void \* l, const void \* n) {

return (\*(int\*)l - \*(int\*)n);

}

void SCAN(int arr[], int head, char\* dn){

int seek\_num = 0;

int dt, cur\_track;

int leftside[size], rightside[size];

int seek\_seq[size + 3];

int m\_scan = 0, s\_scan = 0;

if (strcmp(dn, "leftside") == 0)

leftside[m\_scan++] = 0;

else if (strcmp(dn, "rightside") == 0)

rightside[s\_scan++] = disk\_size - 1;

for (int p\_s = 0; p\_s < size; p\_s++) {

if (arr[p\_s] < head)

leftside[m\_scan++] = arr[p\_s];

if (arr[p\_s] > head)

rightside[s\_scan++] = arr[p\_s];

}

qsort(leftside, m\_scan, sizeof(int), comp);

qsort(rightside, s\_scan, sizeof(int), comp);

int go = 2;

int ind = 0;

while (go--) {

if (strcmp(dn, "leftside") == 0) {

for (int p\_s = m\_scan - 1; p\_s >= 0; p\_s--) {

cur\_track = leftside[p\_s];

seek\_seq[ind++] = cur\_track;

dt = abs(cur\_track - head);

seek\_num += dt;

head = cur\_track;

}

dn = "rightside";

}

else if (strcmp(dn, "rightside") == 0) {

for (int p\_s = 0; p\_s < s\_scan; p\_s++) {

cur\_track = rightside[p\_s];

seek\_seq[ind++] = cur\_track;

dt = abs(cur\_track - head);

seek\_num += dt;

head = cur\_track;

}

dn = "leftside";

}

}

printf("Num of seek process = %d", seek\_num);

printf("Sequence is");

for (int p\_s = 0; p\_s < ind; p\_s++) {

printf("%d", seek\_seq[p\_s]);

}

}

int main(){

int arr[size] = { 126, 90, 14, 50, 25, 42, 51, 78, 102, 100 };

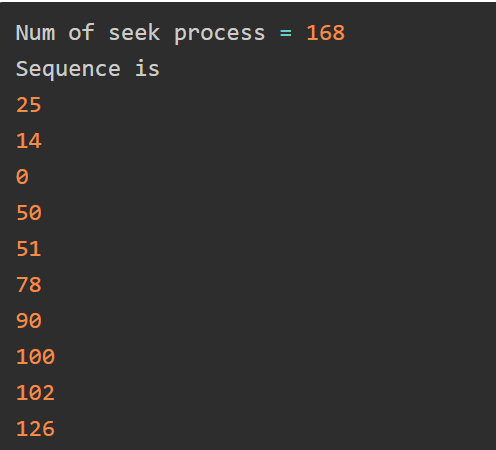
int head = 42;

char dn[] = "leftside";

SCAN(arr, head, dn);

return 0;}

Output:



**C-SCAN:**

size = 8

disk\_size = 200

def CSCAN(arr, head):

seek\_count = 0

distance = 0

cur\_track = 0

left = []

right = []

seek\_sequence = []

left.append(0)

right.append(disk\_size - 1)

for i in range(size):

if (arr[i] < head):

left.append(arr[i])

if (arr[i] > head):

right.append(arr[i])

left.sort()

right.sort()

for i in range(len(right)):

cur\_track = right[i]

seek\_sequence.append(cur\_track)

distance = abs(cur\_track - head)

seek\_count += distance

head = cur\_track

head = 0

seek\_count += (disk\_size - 1)

for i in range(len(left)):

cur\_track = left[i]

seek\_sequence.append(cur\_track)

distance = abs(cur\_track - head)

seek\_count += distance

head = cur\_track

print("Total number of seek operations =",

seek\_count)

print("Seek Sequence is")

print(\*seek\_sequence, sep="\n")

arr = [176, 79, 34, 60,

92, 11, 41, 114]

head = 50

print("Initial position of head:", head)

CSCAN(arr, head)

