**Registration Number: 19BCE2119**

**Name: Gaurav Kumar Singh**

**Course: Operating System**

**Digital Assignment-4**

**1. Memory Management**

**(a)**

#include <iostream>

using namespace std;

int main() {

int c, i, j, k, n, l, m[10], p[10], po[20], flag, z, y, temp, temp1;

cout << "Enter memory total partitions:\t";

cin >> n;

cout << "\nEnter memory size for\n";

for (i = 1; i <= n; i++) {

cout << "\npartition " << i << " :\t";

cin >> m[i];

po[i] = i;

}

cout << "\nEnter total number of process:\t";

cin >> j;

cout << "\nEnter memory size for\n";

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

cout << "\nprocess " << i << " :\t";

cin >> p[i];

}

c = 1;

while (c > 0 && c < 4) {

cout << "1.First fit\n2.Best fit\n3.Worst fit\nEnter your choice:\t";

cin >> c;

switch (c) {

case 1:

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

flag = 1;

for (k = 1; k <= n; k++) {

if (p[i] <= m[k]) {

cout << "\nProcess " << i << " of size " << p[i] << "KB allocated at memory partition:\t" << po[k];

m[k] = m[k] - p[i];

break;

} else {

flag++;

}

}

if (flag > n) {

cout << "\nProcess " << i << " of size " << p[i] << "KB can't be allocated";

}

cout << "\n";

}

break;

case 2:

for (y = 1; y <= n; y++) {

for (z = y; z <= n; z++) {

if (m[y] > m[z]) {

temp = m[y];

m[y] = m[z];

m[z] = temp;

temp1 = po[y];

po[y] = po[z];

po[z] = temp1;

}

}

}

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

flag = 1;

for (k = 1; k <= n; k++) {

if (p[i] <= m[k]) {

cout << "\nProcess " << i << " of size " << p[i] << "KB allocated at memory partition:\t" << po[k];

m[k] = m[k] - p[i];

break;

} else {

flag++;

}

}

if (flag > n) {

cout << "\nProcess " << i << " of size " << p[i] << "KB can't be allocated";

}

cout << "\n";

}

break;

case 3:

for (y = 1; y <= n; y++) {

for (z = y; z <= n; z++) {

if (m[y] < m[z]) {

temp = m[y];

m[y] = m[z];

m[z] = temp;

temp1 = po[y];

po[y] = po[z];

po[z] = temp1;

}

}

}

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

flag = 1;

for (k = 1; k <= n; k++) {

if (p[i] <= m[k]) {

cout << "\nProcess " << i << " of size " << p[i] << "KB allocated at memory partition:\t" << po[k];

m[k] = m[k] - p[i];

break;

} else {

flag++;

}

}

if (flag > n) {

cout << "\nProcess " << i << " of size " << p[i] << "KB can't be allocated";

}

cout << "\n";

}

break;

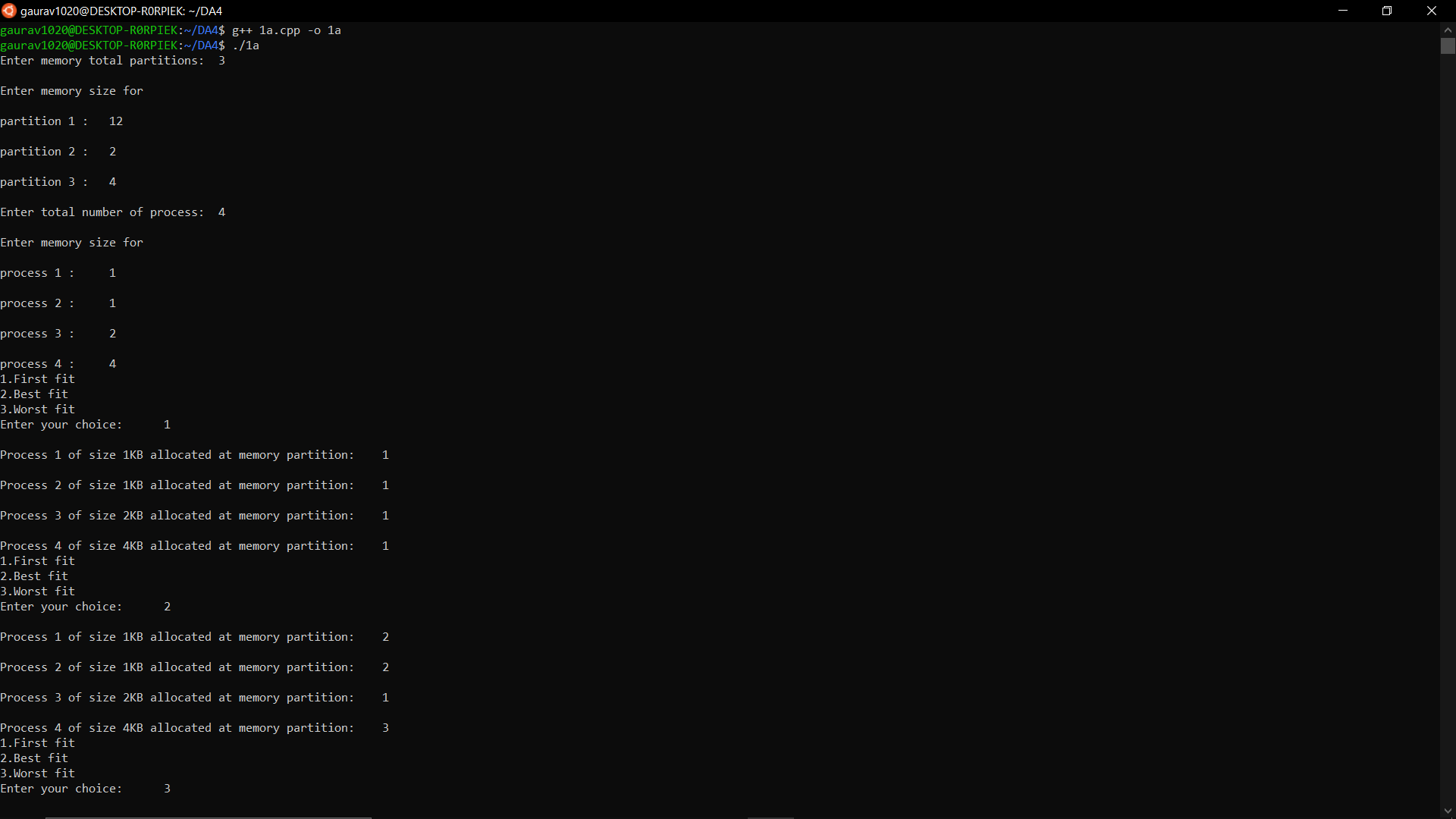
}

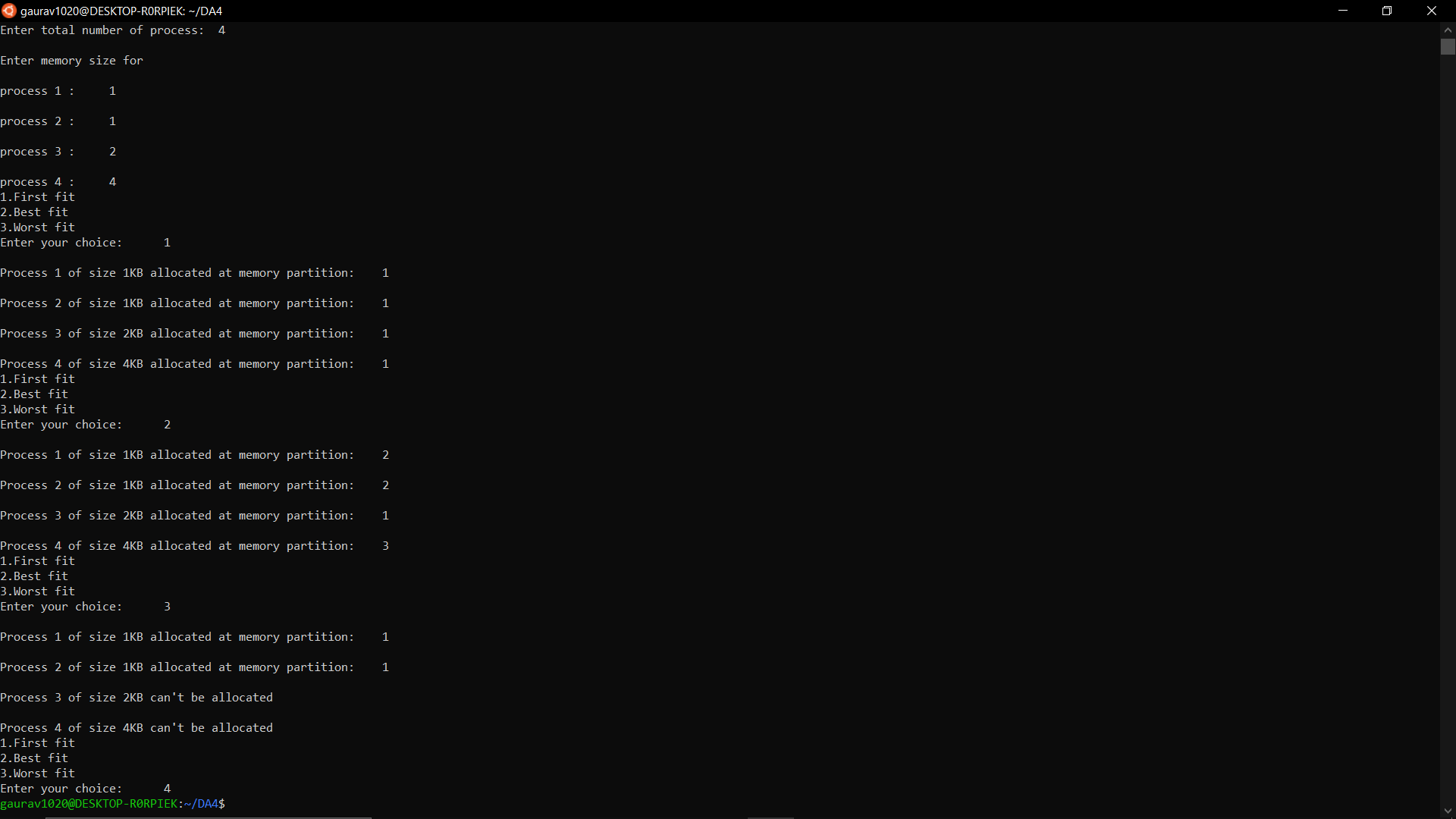
}

return 0;

}

**OUTPUT**





**(b)**

1. **FIFO**

#include<stdio.h>

int main() {

int reference\_string[10], page\_faults = 0, m, n, s, pages, frames;

printf("\nEnter Total Number of Pages:\t");

scanf("%d", & pages);

printf("\nEnter values of Reference String:\n");

for (m = 0; m < pages; m++) {

printf("Value No. [%d]:\t", m + 1);

scanf("%d", & reference\_string[m]);

}

printf("\nEnter Total Number of Frames:\t");

scanf("%d", & frames);

int temp[frames];

for (m = 0; m < frames; m++){

temp[m] = -1;

}

for (m = 0; m < pages; m++){

s = 0;

for (n = 0; n < frames; n++){

if (reference\_string[m] == temp[n]){

s++;

page\_faults--;

}

}

page\_faults++;

if ((page\_faults <= frames) && (s == 0)){

temp[m] = reference\_string[m];

} else if (s == 0){

temp[(page\_faults - 1) % frames] = reference\_string[m];

}

printf("\n");

for (n = 0; n < frames; n++){

printf("%d\t", temp[n]);

}

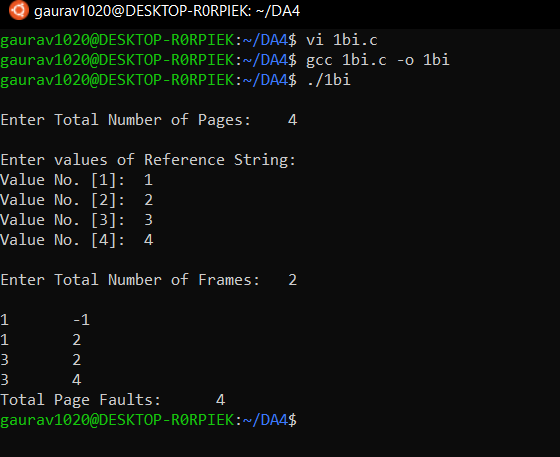
}

printf("\nTotal Page Faults:\t%d\n", page\_faults);

return 0;

}

**OUTPUT**



1. **LRU**

#include<stdio.h>

int main(){

int frames[10], temp[10], pages[10];

int total\_pages, m, n, position, k, l, total\_frames;

int a = 0, b = 0, page\_fault = 0;

printf("\nEnter Total Number of Frames:\t");

scanf("%d", & total\_frames);

for (m = 0; m < total\_frames; m++){

frames[m] = -1;

}

printf("Enter Total Number of Pages:\t");

scanf("%d", & total\_pages);

printf("Enter Values for Reference String:\n");

for (m = 0; m < total\_pages; m++){

printf("Value No.[%d]:\t", m + 1);

scanf("%d", & pages[m]);

}

for (n = 0; n < total\_pages; n++){

a = 0, b = 0;

for (m = 0; m < total\_frames; m++){

if (frames[m] == pages[n]){

a = 1;

b = 1;

break;

}

}

if (a == 0){

for (m = 0; m < total\_frames; m++){

if (frames[m] == -1){

frames[m] = pages[n];

b = 1;

break;

}

}

}

if (b == 0){

for (m = 0; m < total\_frames; m++){

temp[m] = 0;

}

for (k = n - 1, l = 1; l <= total\_frames - 1; l++, k--){

for (m = 0; m < total\_frames; m++){

if (frames[m] == pages[k]){

temp[m] = 1;

}

}

}

for (m = 0; m < total\_frames; m++){

if (temp[m] == 0){

position = m;

}

}

frames[position] = pages[n];

page\_fault++;

}

printf("\n");

for (m = 0; m < total\_frames; m++){

printf("%d\t", frames[m]);

}

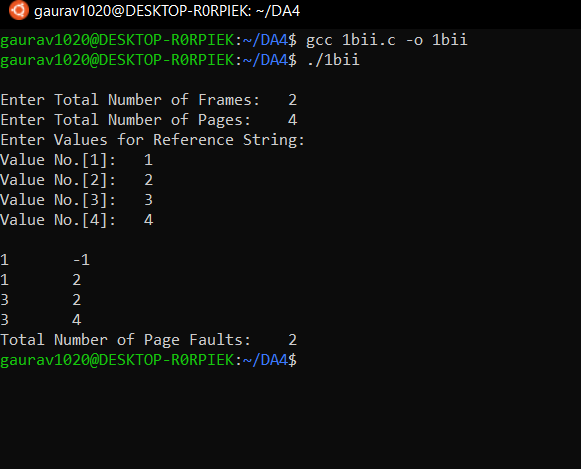
}

printf("\nTotal Number of Page Faults:\t%d\n", page\_fault);

return 0;

}

**OUTPUT**



iii) **Optimal Page Replacement**

#include<stdio.h>

int main()

{

int reference\_string[25], frames[25], interval[25];

int pages, total\_frames,m, n, temp, flag, found, position, maximum\_interval, page\_faults = 0, previous\_frame = -1;

printf("\nEnter Total Number of Pages:\t");

scanf("%d", & pages);

printf("\nEnter Values of Reference String\n");

for (m = 0; m < pages; m++){

printf("Value No.[%d]:\t", m + 1);

scanf("%d", & reference\_string[m]);

}

printf("\nEnter Total Number of Frames:\t");

scanf("%d", & total\_frames);

for (m = 0; m < total\_frames; m++){

frames[m] = -1;

}

for (m = 0; m < pages; m++){

flag = 0;

for (n = 0; n < total\_frames; n++){

if (frames[n] == reference\_string[m]){

flag = 1;

printf("\t");

break;

}

}

if (flag == 0){

if (previous\_frame == total\_frames - 1){

for (n = 0; n < total\_frames; n++){

for (temp = m + 1; temp < pages; temp++){

interval[n] = 0;

if (frames[n] == reference\_string[temp]){

interval[n] = temp - m;

break;

}

}

}

found = 0;

for (n = 0; n < total\_frames; n++){

if (interval[n] == 0){

position = n;

found = 1;

break;

}

}

} else{

position = ++previous\_frame;

found = 1;

}

if (found == 0){

maximum\_interval = interval[0];

position = 0;

for (n = 1; n < total\_frames; n++){

if (maximum\_interval < interval[n]){

maximum\_interval = interval[n];

position = n;

}

}

}

frames[position] = reference\_string[m];

printf("FAULT\t");

page\_faults++;

}

for (n = 0; n < total\_frames; n++){

if (frames[n] != -1){

printf("%d\t", frames[n]);

}

}

printf("\n");

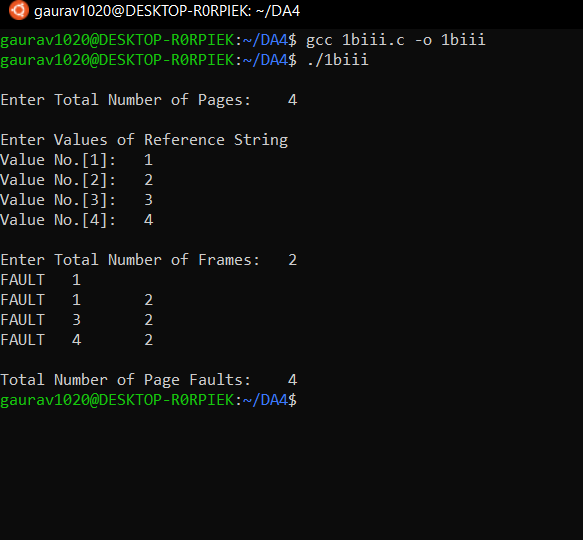
}

printf("\nTotal Number of Page Faults:\t%d\n", page\_faults);

return 0;

}

**OUTPUT**



**(c)**

#include <stdio.h>

int n, pg[30], fr[10];

void fifo();

void optimal();

void lru();

void main() {

int i, ch;

printf("\nEnter total number of pages:");

scanf("%d", & n);

printf("\nEnter page references:");

for (i = 0; i < n; i++)

scanf("%d", & pg[i]);

do {

printf("\n\tMENU\n");

printf("\n1)FIFO");

printf("\n2)OPTIMAL");

printf("\n3)LRU");

printf("\n4)Exit");

printf("\nEnter your choice:");

scanf("%d", & ch);

switch (ch) {

case 1:

fifo();

break;

case 2:

optimal();

break;

case 3:

lru();

break;

}

} while (ch != 4);

getchar();

}

void fifo() {

int i, f, r, s, count, flag, num, psize;

f = 0;

r = 0;

s = 0;

flag = 0;

count = 0;

printf("\nEnter size of page frame:");

scanf("%d", & psize);

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

fr[i] = -1;

}

while (s < n) {

flag = 0;

num = pg[s];

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

if (num == fr[i]) {

s++;

flag = 1;

break;

}

}

if (flag == 0) {

if (r < psize) {

fr[r] = pg[s];

r++;

s++;

count++;

} else {

if (f < psize) {

fr[f] = pg[s];

s++;

f++;

count++;

} else

f = 0;

}

}

printf("\n");

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

printf("%d\t", fr[i]);

}

}

printf("\nPage Faults=%d", count);

getchar();

}

void optimal() {

int count[10], i, j, k, l, m, p, r, fault, fSize, flag, temp, max, tempflag = 0;

fault = 0;

k = 0;

printf("\nEnter frame size:");

scanf("%d", & fSize);

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

count[i] = 0;

fr[i] = -1;

}

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

flag = 0;

temp = pg[i];

for (j = 0; j < fSize; j++) {

if (temp == fr[j]) {

flag = 1;

break;

}

}

if ((flag == 0) && (k < fSize)) {

fault++;

fr[k] = temp;

k++;

}

else if ((flag == 0) && (k == fSize)) {

fault++;

for (l = 0; l < fSize; l++) {

count[l] = 0;

}

for (m = 0; m < fSize; m++)

{

tempflag = 0;

for (r = i + 1; r < n; r++) {

if (fr[m] == pg[r]) {

if (count[m] == 0)

count[m] = r;

tempflag = 1;

}

}

if (tempflag != 1) {

count[m] = n + 1;

}

}

p = 0;

max = count[0];

for (l = 0; l < fSize; l++) {

if (count[l] > max) {

max = count[l];

p = l;

}

}

fr[p] = temp;

}

printf("\n");

for (l = 0; l < fSize; l++) {

printf("%d\t", fr[l]);

}

}

printf("\nTotal number of faults=%d", fault);

getchar();

}

void lru() {

int count[10], i, j, k, fault, f, flag, temp, current, c, dist, least, m, cnt, p, x;

fault = 0;

dist = 0;

k = 0;

printf("\nEnter frame size:");

scanf("%d", & f);

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

count[i] = 0;

fr[i] = -1;

}

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

flag = 0;

temp = pg[i];

for (j = 0; j < f; j++) {

if (temp == fr[j]) {

flag = 1;

count[j] = i;

break;

}

}

if ((flag == 0) && (k < f)) {

fault++;

fr[k] = temp;

count[k] = i;

k++;

}

else if ((flag == 0) && (k == f)) {

fault++;

least = count[0];

for (m = 0; m < f; m++) {

if (count[m] < least) {

least = count[m];

p = m;

}

}

fr[p] = temp;

count[p] = i;

p = 0;

}

printf("\n");

for (x = 0; x < f; x++) {

printf("%d\t", fr[x]);

}

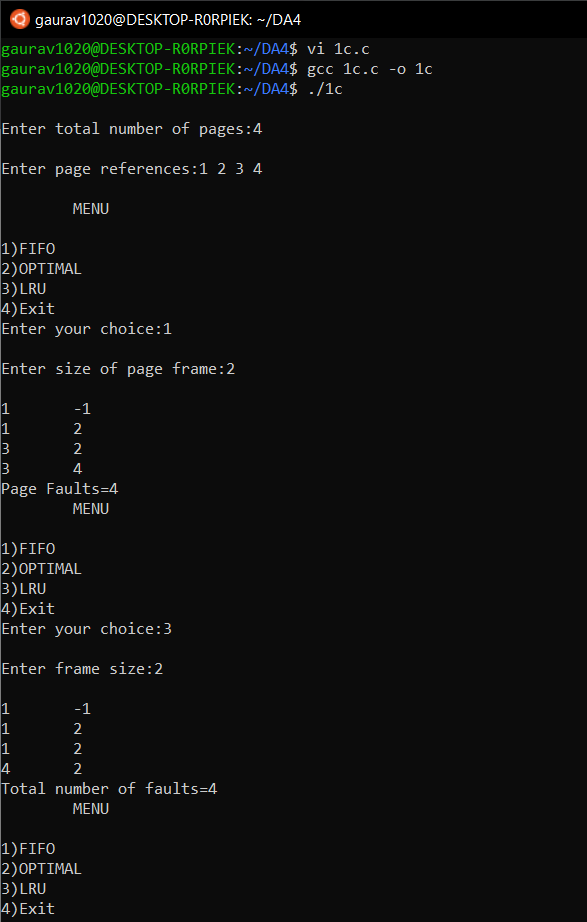
}

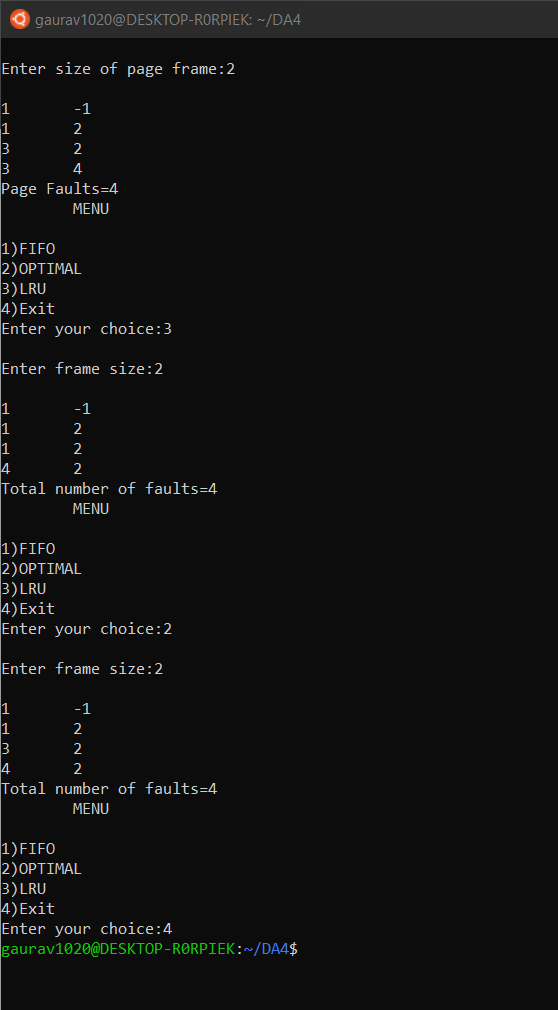
printf("\nTotal number of faults=%d", fault);

getchar();

}

**OUTPUT**





**2. File System and Disk Management**

**(a)**

1. **SSTF**

#include<bits/stdc++.h>

using namespace std;

int main() {

int i, j, k, n, m, sum = 0, x, y, h;

cout << "Enter the size of disk\n";

cin >> m;

cout << "Enter number of requests\n";

cin >> n;

cout << "Enter the requests\n";

vector < int > a(n), b;

map < int, int > mp;

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

cin >> a[i];

mp[a[i]]++;

}

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

if (a[i] > m) {

cout << "Error, Unknown position " << a[i] << "\n";

return 0;

}

}

cout << "Enter the head position\n";

cin >> h;

int temp = h;

int ele;

b.push\_back(h);

int count = 0;

while (count < n) {

int diff = 999999;

for (auto q: mp) {

if (abs(q.first - temp) < diff) {

ele = q.first;

diff = abs(q.first - temp);

}

}

mp[ele]--;

if (mp[ele] == 0) {

mp.erase(ele);

}

b.push\_back(ele);

temp = ele;

count++;

}

cout << b[0];

temp = b[0];

for (i = 1; i < b.size(); i++) {

cout << " -> " << b[i];

sum += abs(b[i] - temp);

temp = b[i];

}

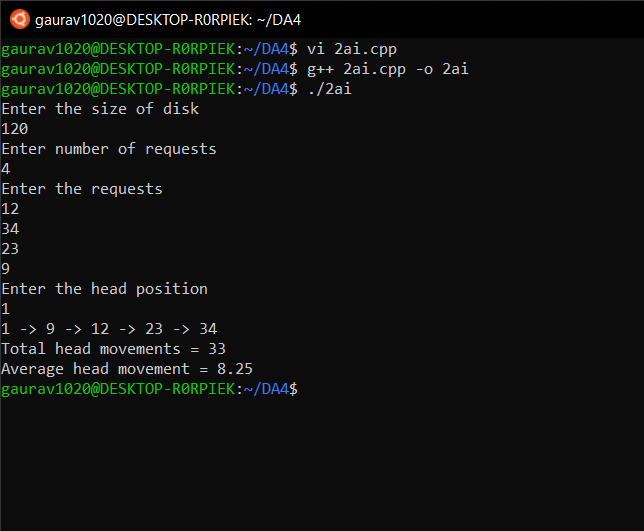
cout << '\n';

cout << "Total head movements = " << sum << '\n';

cout << "Average head movement = " << (float) sum / n << '\n';

return 0;

}



1. **SCAN**

#include <stdio.h>

void main(){

int i, j, n, h, temp = 0, dEnd = 199, hPos, sum = 0, count = 1;

int rq[100], sq[100];

printf("\nEnter No. of Processes: ");

scanf("%d", & n);

printf("\nEnter Head value: ");

scanf("%d", & h);

printf("\nEnter elements into Request Queue");

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

scanf(" %d", & rq[i]);

}

rq[i] = h;

rq[i + 1] = 0;

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

for (j = 0; j < n - 1; j++){

if (rq[j] > rq[j + 1]){

temp = rq[j];

rq[j] = rq[j + 1];

rq[j + 1] = temp;

}

}

}

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

if (rq[i] > h){

hPos = i - 1;

break;

}

}

sq[0] = h;

printf("\nScheduling\n");

if (h < (dEnd - h)){

for (i = hPos; i >= 0; i--){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

for (i = hPos + 1; i < n; i++){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

} else{

for (i = hPos + 1; i < n; i++){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

for (i = hPos; i >= 0; i--){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

}

printf("\n Head Movements: ");

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

if (sq[i] > sq[i + 1]){

sum += (sq[i] - sq[i + 1]);

} else{

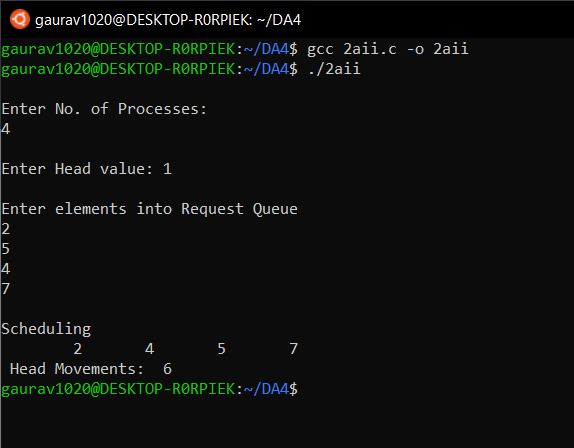
sum += (sq[i + 1] - sq[i]);

}

}

printf(" %d \n", sum);

}



1. **C-SCAN**

#include <stdio.h>

void main(){

int i, j, n, h, temp = 0, dEnd = 199, hPos, sum = 0, count = 1;

int rq[100], sq[100];

printf("\nEnter No. of Processes: ");

scanf("%d", & n);

printf("\nEnter Head value: ");

scanf("%d", & h);

printf("\nEnter elements into Request Queue");

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

scanf(" %d", & rq[i]);

}

rq[i] = h;

rq[i + 1] = 0;

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

for (j = 0; j < n - 1; j++){

if (rq[j] > rq[j + 1]){

temp = rq[j];

rq[j] = rq[j + 1];

rq[j + 1] = temp;

}

}

}

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

if (rq[i] > h){

hPos = i - 1;

break;

}

}

sq[0] = h;

printf("\nScheduling\n");

if (h < (dEnd - h)){

for (i = hPos; i >= 0; i--){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

for (i = n - 1; i > hPos; i--){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

} else{

for (i = hPos + 1; i < n; i--){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

for (i = 0; i >= hPos; i++){

sq[count] = rq[i];

count++;

printf("\t%d ", rq[i]);

}

}

printf("\n Head Movements: ");

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

if (sq[i] > sq[i + 1]){

sum += (sq[i] - sq[i + 1]);

} else{

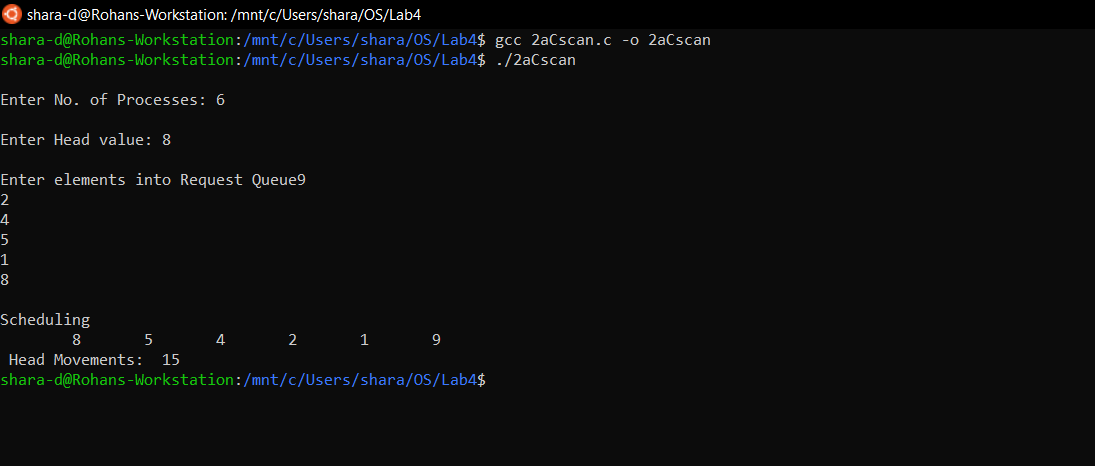
sum += (sq[i + 1] - sq[i]);

}

}

printf(" %d \n", sum);

}



1. **FCFS**

#include<stdio.h>

void main(){

int h, i, rq[100], sum = 0, n, j;

printf("\n Enter the length: ");

scanf("%d", & n);

printf("\n Enter the Head Value: ");

scanf("%d", & h);

printf("\n Enter the Request Queue ");

for (i = 1; i < n + 1; i++){

scanf("%d", & rq[i]);

}

rq[0] = h;

for (j = 0; j < n; j++){

if (rq[j] > rq[j + 1]){

sum = (sum + (rq[j] - rq[j + 1]));

} else{

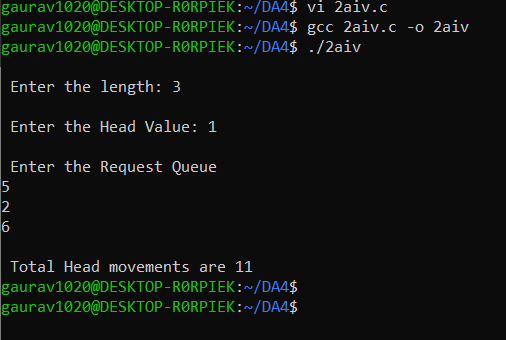
sum = (sum + (rq[j + 1] - rq[j]));

}

}

printf("\n Total Head movements are %d \n", sum);

}



**(b)**

1. **Sequential**

#include<stdio.h>

int main()

{

int n, i, j, b[20], sb[20], t[20], x, c[20][20];

printf("Enter no.of files:");

scanf("%d", & n);

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

printf("Enter no. of blocks occupied by file%d", i + 1);

scanf("%d", & b[i]);

printf("Enter the starting block of file%d", i + 1);

scanf("%d", & sb[i]);

t[i] = sb[i];

for (j = 0; j < b[i]; j++){

c[i][j] = sb[i]++;

}

}

printf("Filename\tStart block\tlength\n");

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

printf("%d\t %d \t%d\n", i + 1, t[i], b[i]);

}

printf("blocks occupiedare:");

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

printf("fileno%d", i + 1);

for (j = 0; j < b[i]; j++){

printf("\t%d", c[i][j]);

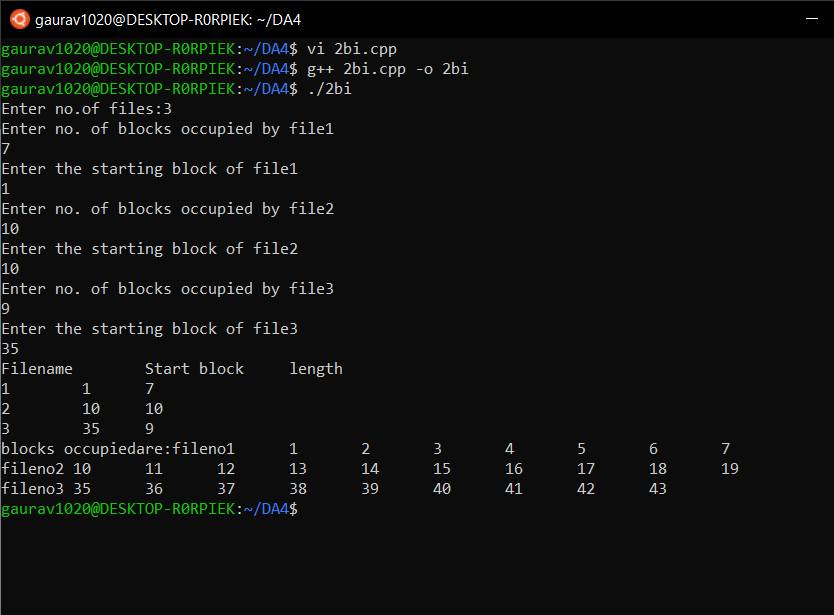
}

printf("\n");

}

return 0;

}



1. **Indexed**

#include<stdio.h>

int main(){

int n, m[20], i, j, ib[20], b[20][20];

printf("Enter no. of files:");

scanf("%d", & n);

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

printf("Enter index block :", i + 1);

scanf("%d", & ib[i]);

printf("Enter blocks occupied by file%d:", i + 1);

scanf("%d", & m[i]);

printf("enter blocks of file%d:", i + 1);

for (j = 0; j < m[i]; j++){

scanf("%d", & b[i][j]);

}

}

printf("\nFile\t index\tlength\n");

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

printf("%d\t%d\t%d\n", i + 1, ib[i], m[i]);

}

printf("blocks occupiedare:");

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

printf("fileno%d", i + 1);

for (j = 0; j < m[i]; j++){

printf("\t%d--->%d\n", ib[i], b[i][j]);

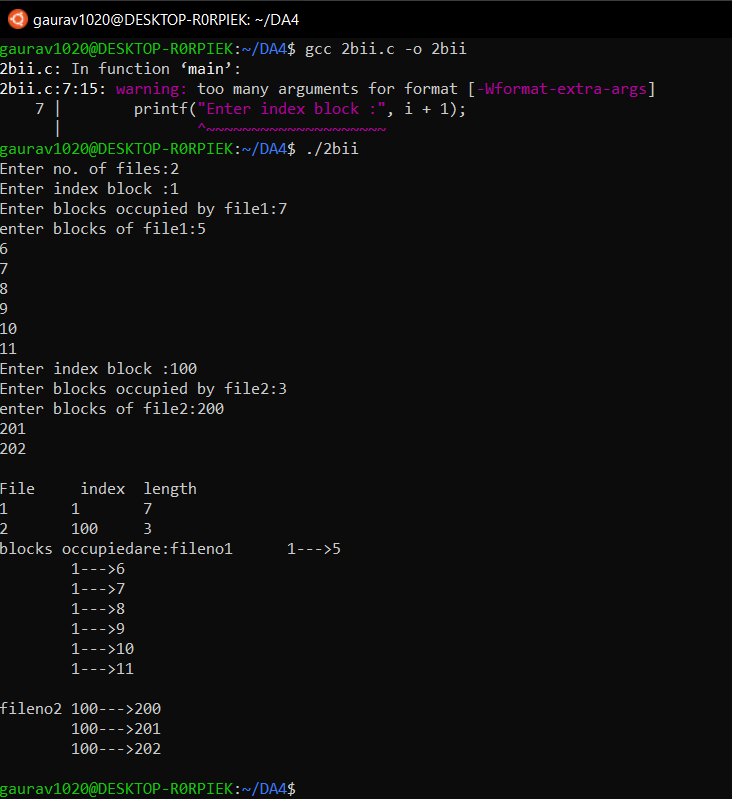
}

printf("\n");

}

return 0;

}



1. **Linked**

#include<stdio.h>

struct file{

char fname[10];

int start, size, block[10];

}

f[10];

int main(){

int i, j, n;

printf("Enter no. of files:");

scanf("%d", & n);

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

printf("Enter file name:");

scanf("%s", & f[i].fname);

printf("Enter starting block:");

scanf("%d", & f[i].start);

f[i].block[0] = f[i].start;

printf("Enter no.of blocks:");

scanf("%d", & f[i].size);

printf("Enter block numbers:");

for (j = 1; j <= f[i].size; j++){

scanf("%d", & f[i].block[j]);

}

}

printf("File\tstart\tsize\tblock\n");

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

printf("%s\t%d\t%d\t", f[i].fname, f[i].start, f[i].size);

for (j = 0; j < f[i].size; j++){

printf("%d--->", f[i].block[j]);

}

printf("%d", f[i].block[j]);

printf("\n");

}

return 0;

}

