PAGE REP ALGO:-

import java.io.\*;

import java.util.\*;

class OptimalPageReplacement{

// Function to check whether a page exists

// in a frame or not

static boolean search(int key, int[] fr)

{

for (int i = 0; i < fr.length; i++)

if (fr[i] == key)

return true;

return false;

}

// Function to find the frame that will not be used recently in future after given index in pg[0..pn-1]

static int predict(int pg[], int[] fr, int pn,int index){

// Store the index of pages which are going

// to be used recently in future

int res = -1, farthest = index;

for (int i = 0; i < fr.length; i++) {

int j;

for (j = index; j < pn; j++) {

if (fr[i] == pg[j]) {

if (j > farthest) {

farthest = j;

res = i;

}

break;

}

}

// If a page is never referenced in future,

// return it.

if (j == pn)

return i;

}

// If all of the frames were not in future,

// return any of them, we return 0. Otherwise

// we return res.

return (res == -1) ? 0 : res;

}

static void optimalPage(int pg[], int pn, int fn)

{

// Create an array for given number of

// frames and initialize it as empty.

int[] fr = new int[fn];

// Traverse through page reference array

// and check for miss and hit.

int hit = 0;

int index = 0;

for (int i = 0; i < pn; i++) {

// Page found in a frame : HIT

if (search(pg[i], fr)) {

hit++;

continue;

}

// Page not found in a frame : MISS

// If there is space available in frames.

if (index < fn)

fr[index++] = pg[i];

// Find the page to be replaced.

else {

int j = predict(pg, fr, pn, i + 1);

fr[j] = pg[i];

}

}

System.out.println("No. of hits = " + hit);

System.out.println("No. of misses = " + (pn - hit));

}

// driver function

public static void main(String[] args) { /\*Name :- Shivam Bendre Roll No:- COTA16 Division:- "A1" \*/

int pg[]

= { 8, 1, 9, 2, 0, 5, 0, 2, 2, 4, 0, 4, 4,6,0,2,7,6,0,3,9,6,2,9,8,6 };

int pn = pg.length;

int fn = 4;

optimalPage(pg, pn, fn); }

}