#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

#include <time.h>

void \*fifo(int \*arr, int numFrames);

void \*lru(int \*arr, int numFrames);

void \*optimal(int \*arr, int numFrames);

void \*printOutput(int numFramesAllocated, double freq, int frames, int numMjrFlts);

int count = 0;

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

{

int numFrames = atoi(argv[3]);

int arr[10000];

int i = 0;

int x;

char command;

FILE \*fptr;

fptr = fopen(argv[7], "r");

if (fptr == NULL)

printf("File Cannot Be Read");

if (strcmp(argv[1], "-f") == 0)

{

//printf("\nwe are calling fifo\n");

for (x = 0; x < 10000; x++)

{

arr[x] = -1;

}

while (fscanf(fptr, "%c %d", &command, &i) == 2)

{

arr[count] = i;

count++;

}

fifo(arr, numFrames);

}

else if (strcmp(argv[1], "-l") == 0)

{

//printf("\nwe are in the algorithim low: %lf high: %lf numFrames: %d\n", low, high, numFrames);

for (x = 0; x < 10000; x++)

{

arr[x] = -1;

}

while (fscanf(fptr, "%c %d", &command, &i) == 2)

{

arr[count] = i;

count++;

}

lru(arr, numFrames);

}

else if (strcmp(argv[1], "-o") == 0)

{

for (x = 0; x < 10000; x++)

{

arr[x] = -1;

}

while (fscanf(fptr, "%c %d", &command, &i) == 2)

{

arr[count] = i;

count++;

}

optimal(arr, numFrames);

}

}

void \*fifo(int \*arr, int numFrames)

{

int low = -1;

int high = 1;

//printf("\nwe are in the algorithim low: %lf high: %lf numFrames: %d\n", low, high, numFrames);

int i = 0;

int point = 0;

int \*mem;

int curmem[numFrames];

mem = curmem;

int numFlts = 0;

int numMjrFlts = 0;

double ratio = 1.00;

double sc = 0.00;

while (arr[i] != -1 && i < 10000)

{

if (point < numFrames)

{

mem[point] = arr[i];

point++;

numFlts++;

}

else

{

int status = 0;

int j;

for (j = 0; j < numFrames; j++)

{

if (mem[j] == arr[i])

{

status = 1;

}

}

sc++;

if (status == 0)

{

int k;

ratio = (numMjrFlts / sc);

point++;

if (ratio > high)

{

//adding new frame

numFrames += 1;

int newArr[numFrames];

for (k = 0; k < numFrames - 1; k++)

{

newArr[k] = mem[k];

}

newArr[numFrames - 1] = arr[i];

mem = newArr;

printf("\nInput Line %d: n = %d", i, numFrames);

}

else if (ratio < low && numFrames > 1)

{

numFrames -= 1;

int newArr[numFrames];

for (k = 0; k < numFrames; k++)

{

newArr[k] = mem[k + 1];

}

newArr[0] = arr[k];

mem = newArr;

numMjrFlts++;

printf("\nInput Line %d: n = %d", i, numFrames);

}

else

{

//Not adding new frame

for (k = 0; k < numFrames - 1; k++)

{

mem[k] = mem[k + 1];

}

mem[numFrames - 1] = arr[i];

numMjrFlts++;

}

numFlts++;

}

}

i++;

}

printOutput(numFlts, ratio, numFrames, numMjrFlts);

}

void \*lru(int \*arr, int numFrames)

{

int low = -1;

int high = 1;

//printf("\nwe are in the algorithim low: %lf high: %lf numFrames: %d\n", low, high, numFrames);

int i = 0;

int length = count;

int point = 0;

int \*mem;

int curmem[numFrames];

mem = curmem;

int numFlts = 0;

int numMjrFlts = 0;

double ratio = 1.00;

double sc = 0.00;

while (arr[i] != -1 && i < 10000)

{

if (point < numFrames)

{

mem[point] = arr[i];

point++;

numFlts++;

}

else

{

int status = 0;

int j;

for (j = 0; j < numFrames; j++)

{

if (mem[j] == arr[i])

{

status = 1;

}

}

sc++;

if (status == 0)

{

int k;

ratio = (numMjrFlts / sc);

point++;

if (ratio > high)

{

//adding new frame

numFrames += 1;

int newArr[numFrames];

for (k = 0; k < numFrames - 1; k++)

{

newArr[k] = mem[k];

}

newArr[numFrames - 1] = arr[i];

mem = newArr;

printf("\nInput Line %d: n = %d", i, numFrames);

}

else if (ratio < low && numFrames > 1)

{

//taking away a frame

//printf("\nWere taking away a frame");

int u = i + 1;

int h = 0;

int stat = 0;

int fake = 0;

int saveA = 0;

int saveB = 0;

int saveCheck = 0;

int compare[numFrames];

for (k = 0; k < numFrames; k++)

{

compare[k] = 0;

}

while (stat != 1)

{

if (fake == numFrames - 2 || u < 0)

{

int o;

for (o = 0; o < numFrames; o++)

{

if (compare[o] == 0 && saveCheck == 0)

{

saveA = o;

saveCheck = 1;

}

else if (compare[o] == 0 && saveCheck == 1)

{

saveB = o;

}

}

//Found both now time to do a pg fault.

int newArr[numFrames - 1];

int f;

int ef = 0;

int nA = 0;

for (f = 0; f < numFrames; f++)

{

if (saveA == f)

{

nA = ef;

newArr[ef] = mem[f];

ef++;

}

else if (saveB == f)

{

//get outta here b

}

else

{

newArr[ef] = mem[f];

ef++;

}

}

newArr[nA] = arr[i];

mem = newArr;

numFrames -= 1;

numMjrFlts++;

stat = 1;

break;

}

else

{

for (h = 0; h < numFrames; h++)

{

if (arr[u] == mem[h] && compare[h] != 1)

{

compare[h] = 1;

fake++;

//break;

}

}

}

u--;

}

printf("\nInput Line %d: n = %d", i, numFrames);

}

else

{

//Not adding new frame

//First we need to find the lru

int u = i + 1;

int h = 0;

int stat = 0;

int fake = 0;

int save = 0;

int compare[numFrames];

for (k = 0; k < numFrames; k++)

{

compare[k] = 0;

}

while (stat != 1)

{

if (fake == numFrames - 1)

{

int o;

for (o = 0; o < numFrames; o++)

{

if (compare[o] == 0)

{

save = o;

}

}

//Found it now time to do a pg fault.

mem[save] = arr[i];

numMjrFlts++;

stat = 1;

break;

}

else

{

for (h = 0; h < numFrames; h++)

{

if (arr[u] == mem[h] && compare[h] != 1)

{

compare[h] = 1;

fake++;

break;

}

}

}

u--;

}

}

numFlts++;

}

}

i++;

}

printOutput(numFlts, ratio, numFrames, numMjrFlts);

}

void \*optimal(int \*arr, int numFrames)

{

int low = -1;

int high = 1;

//printf("\nwe are in the algorithim low: %lf high: %lf numFrames: %d\n", low, high, numFrames);

int i = 0;

int length = count;

int point = 0;

int \*mem;

int curmem[numFrames];

mem = curmem;

int numFlts = 0;

int numMjrFlts = 0;

double ratio = 0.00;

double sc = 0.00;

while (arr[i] != -1 && i < 10000)

{

if (point < numFrames)

{

mem[point] = arr[i];

point++;

numFlts++;

}

else

{

int status = 0;

int j;

for (j = 0; j < numFrames; j++)

{

if (mem[j] == arr[i])

{

status = 1;

}

}

sc++;

if (status == 0)

{

int k;

ratio = (numMjrFlts / sc);

point++;

if (ratio > high)

{

//adding new frame

numFrames += 1;

int newArr[numFrames];

for (k = 0; k < numFrames - 1; k++)

{

newArr[k] = mem[k];

}

newArr[numFrames - 1] = arr[i];

mem = newArr;

printf("\nInput Line %d: n = %d", i, numFrames);

}

else if (ratio < low && numFrames > 1)

{

//taking away a frame

//printf("\nWere taking away a frame");

int u = i + 1;

int h = 0;

int stat = 0;

int fake = 0;

int saveA = 0;

int saveB = 0;

int saveCheck = 0;

int compare[numFrames];

for (k = 0; k < numFrames; k++)

{

compare[k] = 0;

}

while (stat != 1)

{

if (fake == numFrames - 2 || u >= length)

{

int o;

for (o = 0; o < numFrames; o++)

{

if (compare[o] == 0 && saveCheck == 0)

{

saveA = o;

saveCheck = 1;

}

else if (compare[o] == 0 && saveCheck == 1)

{

saveB = o;

}

}

//Found both now time to do a pg fault.

int newArr[numFrames - 1];

int f;

int ef = 0;

int nA = 0;

for (f = 0; f < numFrames; f++)

{

if (saveA == f)

{

nA = ef;

newArr[ef] = mem[f];

ef++;

}

else if (saveB == f)

{

//get outta here b

}

else

{

newArr[ef] = mem[f];

ef++;

}

}

newArr[nA] = arr[i];

mem = newArr;

numFrames -= 1;

numMjrFlts++;

stat = 1;

break;

}

else

{

for (h = 0; h < numFrames; h++)

{

if (arr[u] == mem[h])

{

compare[h] = 1;

fake++;

break;

}

}

u++;

}

}

printf("\nInput Line %d: n = %d", i, numFrames);

}

else

{

//Not adding new frame

//First we need to find the lru

int u = i + 1;

int h = 0;

int stat = 0;

int fake = 0;

int save = 0;

int compare[numFrames];

for (k = 0; k < numFrames; k++)

{

compare[k] = 0;

}

while (stat != 1)

{

if (fake == numFrames - 1 || u >= length)

{

int o;

for (o = 0; o < numFrames; o++)

{

if (compare[o] == 0)

{

save = o;

}

}

//Found it now time to do a pg fault.

mem[save] = arr[i];

numMjrFlts++;

stat = 1;

break;

}

else

{

for (h = 0; h < numFrames; h++)

{

if (arr[u] == mem[h])

{

compare[h] = 1;

fake++;

break;

}

}

}

u++;

}

}

numFlts++;

}

}

i++;

}

printOutput(numFlts, ratio, numFrames, numMjrFlts);

}

void \*printOutput(int numFlts, double freq, int frames, int numMjrFlts)

{

printf("\nPage hits: %d\nMinor faults: %lf\nFinal number of frames swapped: %d\n Major page faults: %d\n", numFlts, freq, frames, numMjrFlts);

}