/\* Author: Samuel Casto

\* PantherID: 6330314

\* Description: This program accepts two inputs, a file containing a list of page references and the number of frames. The

\* program will read this list of page references and accounting for the number of frames, output the number of page faults

\* using a FIFO replacement policy. It will also output the final state of memory.

\*/

#include <stdlib.h>

#include <stdio.h>

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

//Verifying we have the right number of arguments

if(argc != 3){

fprintf(stderr, "Argument error, usage: file numOfFrames\n");

return 1;

}

//we have the right number of arguments and need to declare a file and int

int \*frames;

frames = (int\*)malloc(sizeof(int));

if(atoi(argv[2]))

\*frames = atoi(argv[2]);

else {

fprintf(stderr, "Argument order error, usage: file numOfFrames\n");

return 1;

}

//input validation testing

//printf("This is our number of frames: %i\n",\*frames);

//verifying the number of frames is between 1 and 10

if(\*frames < 1 || \*frames > 10){

fprintf(stderr, "Number of frames needs to be between 1 and 10\n");

return 1;

}

//declaring and opening our file

FILE\* file;

//verifying we opened the file

if(!(file = fopen(argv[1], "r"))){

fprintf(stderr, "File did not open\n");

return 1;

}

//keeping track of what is in our page table

int table[\*frames];

//populating it with -1 to represent no pages being in it yet

for(int i = 0; i < \*frames; i++)

table[i] = -1;

//variables for keeping track of our page faults and the current page in the file

int pageFaults = 0;

int current;

int temp = 0;

int frameCount = 0;//used for keeping track of which table value to update

//looping through our file input and checking if there is a page fault or not

while(fscanf(file,"%d",&current) == 1){

//verifying this works as anticipated

//printf("%d is temp\n",current);

//current holds our current value that we need to check if it is in the table yet

for(int i = 0; i < \*frames; i++){

//printf("current value: %d | table[i] value: %d | i value: %d\n",current, table[i], i);

if(current == table[i]){

//we have a hit

temp = 1;

}

}

if(temp == 1)

temp = 0;

else {

//we had a page fault and need to increment pageFaults and update table

pageFaults++;

//after looping through the table we need to update it where the value at 0,..,\*frames

//needs to be updated

if(frameCount == \*frames){

//if OOB then we need to reset frameCount before updating

frameCount = 0;

}

//updating based off frameCount

table[frameCount++] = current;

}

}

//outputting result

printf("FIFO: %d page faults\n",pageFaults);

//fun little switch statement to output based on number of frames

switch(\*frames) {

case 1:

printf("Final state of memory: %d\n",table[0]);

break;

case 2:

printf("Final state of memory: %d %d\n",table[0],table[1]);

break;

case 3:

printf("Final state of memory: %d %d %d\n",table[0],table[1],table[2]);

break;

case 4:

printf("Final state of memory: %d %d %d %d\n",table[0],table[1],table[2],table[3]);

break;

case 5:

printf("Final state of memory: %d %d %d %d %d\n",table[0],table[1],table[2],table[3],table[4]);

break;

case 6:

printf("Final state of memory: %d %d %d %d %d %d\n",table[0],table[1],table[2],table[3],

table[4],table[5]);

break;

case 7:

printf("Final state of memory: %d %d %d %d %d %d %d\n",table[0],table[1],table[2],table[3],

table[4],table[5],table[6]);

break;

case 8:

printf("Final state of memory: %d %d %d %d %d %d %d %d\n",table[0],table[1],table[2],

table[3],table[4],table[5],table[6],table[7]);

break;

case 9:

printf("Final state of memory: %d %d %d %d %d %d %d %d %d\n",table[0],table[1],table[2],

table[3],table[4],table[5],table[6],table[7],table[8]);

break;

case 10:

printf("Final state of memory: %d %d %d %d %d %d %d %d %d %d\n",table[0],table[1],table[2],

table[3],table[4],table[5],table[6],table[7],table[8],table[9]);

break;

}

free(frames);

return 0;

}