```
/**
 * Simulates Page Tables
 *
 *
 * @author Palak Ojha
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include "PageTable.h"

struct page_table_entry {
    unsigned int dirty_valid;
    int frameNum;
    int pageNum;
    int order;
    int last;
    int freq;
    struct page_table_entry *next;
};

struct page_table {
    enum replacement_algorithm mode;
    int page_fault_count;
    int pageCount;
    int frameCount;
    int verbose;
    unsigned int dirty_valid;
    int frameNum;
    int pageNum;
    struct page_table_entry *pte_head;
};

/**
 * Prints a given table
 *
 * @param page_table to print
 */
char* getMode(struct page_table* p) {
    if (p->mode == FIFO) {
        return "FIFO";
    } else if (p->mode == LRU) {
        return "LRU";
    } else if (p->mode == MFU) {
        return "MFU";
    }
    return "unknown";
}

void mfu(struct page_table *pt, int page) {
    int frames[10];
    int pages[30];
    int time[10];
    int count;
    int flag_1;
    int flag_2;

    for(int i = 0; i < pt->frameCount; i++) {
        frames[i] = -1;
    }

    pt->page_fault_count = 0;
    for(int i = 0; i < pt->pageCount; i++) {
        count = 0;
        flag_1 = 0;
        flag_2 = 0;
        for(int j = 0; j < pt->frameCount; j++) {
            if(frames[j] == pages[i]) {
                count++;
                time[j] = count;
                flag_1 = 1;
                flag_2 = 1;
                break;
            }
        }

        if(flag_1 == 0) {
            for(int j = 0; j < pt->frameCount; j++) {
                if(frames[j] == -1) {
                    count++;
                    pt->page_fault_count++;
                    frames[j] = pages[i];
                    time[j] = count;
                    flag_2 = 1;
                    break;
                }
            }
        }

        if(flag_2 == 0) {
            int min = time[0];
            int pos = 0;
            for(int j = 0; j < pt->frameCount; j++) {
                if(time[j] < min) {
                    min = time[j];
                    pos = j;
                }
                count++;
                pt->page_fault_count++;
                frames[pos] = pages[j];
                time[pos] = count;
                break;
            }
        }
    }
}


void fifo(struct page_table *pt, int page) {
    int temp[pt->frameCount];
    for(int i = 0; i < pt->frameCount; i++) {
        temp[i] = -1;
    }
    int count;
    for(int i = 0; i < pt->pageCount; i++) {
        count = 0;
        for(int j = 0; j < pt->frameCount; j++) {
            if(page == temp[j]) {
                count++;
                pt->page_fault_count--;
            }
        }
        pt->page_fault_count++;
        if(count == 0) {
            if(pt->page_fault_count <= pt->frameCount) {
                temp[i] = page;
            } else {
                temp[(pt->page_fault_count - 1) % pt->frameCount] = page;
            }
        }
    }
}

void lru(struct page_table *pt, int page) {
    int frames[10];
    int pages[30];
    int time[10];
    int count;
    int flag_1;
    int flag_2;

    for(int i = 0; i < pt->frameCount; i++) {
        frames[i] = -1;
    }

    pt->page_fault_count = 0;
    for(int i = 0; i < pt->pageCount; i++) {
        count = 0;
        flag_1 = 0;
        flag_2 = 0;
        for(int j = 0; j < pt->frameCount; j++) {
            if(frames[j] == pages[i]) {
                count++;
                time[j] = count;
                flag_1 = 1;
                flag_2 = 1;
                break;
            }
        }

        if(flag_1 == 0) {
            for(int j = 0; j < pt->frameCount; j++) {
                if(frames[j] == -1) {
                    count++;
                    pt->page_fault_count++;
                    frames[j] = pages[i];
                    time[j] = count;
                    flag_2 = 1;
                    break;
                }
            }
        }

        if(flag_2 == 0) {
            int min = time[0];
            int pos = 0;
            for(int j = 0; j < pt->frameCount; j++) {
                if(time[j] < min) {
                    min = time[j];
                    pos = j;
                }
                count++;
                pt->page_fault_count++;
                frames[pos] = pages[j];
                time[pos] = count;
                break;
            }
        }
    }
}


/**
 * Creates a new page table object. Returns a pointer to created page table.
 *
 * @param pageCount Number of pages.
 * @param frameCount Numbers of frames.
 * @param algorithm Page replacement algorithm
 * @param verbose Enables showing verbose table contents.
 * @return A page table object.
 */
struct page_table* page_table_create(int pageCount, int frameCount, enum
replacement_algorithm algorithm, int verbose) {
    struct page_table *pt = (struct page_table*)malloc(sizeof(struct page_table));
    pt->pte_head = (struct page_table_entry*)malloc(sizeof(struct
page_table_entry*));
    pt->pageCount = pageCount;
    pt->frameCount = frameCount;
    pt->mode = algorithm;
    pt->verbose = verbose;
    return pt;
}

/**
 * Destorys an existing page table object. Sets outside variable to NULL.
 *
 * @param pt A page table object.
 */
void page_table_destroy(struct page_table** pt) {
    struct page_table* pt_temp = *pt;
    free(pt_temp->pte_head);
    free(pt_temp);
    pt = NULL;
}

/**
 * Simulates an instruction accessing a particular page in the page table.
 *
 * @param pt A page table object.
 * @param page The page being accessed.
 */
void page_table_access_page(struct page_table *pt, int page) {
    if(pt->mode == FIFO) {

        fifo(pt, page);
    } else if(pt->mode == LRU) {

        lru(pt, page);

    } else if(pt->mode == MFU) {

        mfu(pt, page);
    }
}

/**
 * Displays page table replacement algorithm, number of page faults, and the
 * current contents of the page table.
 *
 * @param pt A page table object.
 */
void page_table_display(struct page_table* pt) {
    printf("====Page Table====\n");
    printf("Mode: %s\n", getMode(pt));
    printf("Page Faults: %d\n", pt->page_fault_count);
    printf("page frame | dirty valid\n");
}

/**
 * Displays the current contents of the page table.
 *
 * @param pt A page table object.
 */
void page_table_display_contents(struct page_table *pt){
    struct page_table_entry* pte = pt->pte_head;
    printf("   %d     %d |     %d     %d\n",
           pt->pageCount, pt->frameNum,
           0, pt->dirty_valid);
}
```