```c
#include "cachelab.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>   //strcpy()适用
#include <unistd.h>   //getopt()适用
#include <getopt.h>
#include <stdlib.h>

/*step2用：定义高速缓存cache结构体   */
typedef struct {
    int valid;      //有效位
    int tag;        //标识位
    int LruNumber;  //LRU算法计数
} Line;             //Line行格式

typedef struct {
    Line *lines;    //指向一组中的行
} Set;              //Set组格式

typedef struct {
    int SetNumber;  //组数
    int LineNumber; //行数
    Set *sets;      //指向一Cache中的组
} Sim_Cache;       //模拟的Cache格式

/*Step3用: LRU变量*/
#define MAX_NUM 2147483647

/*Step4用  统计数量 */
int misses;    //未命中
int hits;      //命中
int evictions; //驱逐

/*
 * Prototypes for Csim helper functions
 * ZHJ,C301,Yuelu,CS,2021Spr
 */

/*Step2 Functions----*/
/*打印help信息*/
void printHelpMenu() {
    printf("Z  Usage: ./csim-ref [-hv] -s <num> -E <num> -b <num> -t <file>\n");
    printf("Options:\n");
    printf("-h               Print this help message.\n");
    printf("-v               Optional verbose flag.\n");
    printf("-s <num>         Number of set index bits.\n");
    printf("-E <num>         Number of lines per set.\n");
    printf("-b <num>         Number of block offset bits.\n");
    printf("-t <file>        Trace file.\n\n\n");
    printf("Examples:\n");
    printf("linux>    ./csim -s 4 -E 1 -b 4 -t traces/yi.trace\n");
    printf("linux>    ./csim -v -s 8 -E 2 -b 4 -t traces/yi.trace\n");
}

/*检查参数合法性*/
void checkOptarg(char *curOptarg) {
    if(curOptarg[0]=='-') {
        printf("./csim :Missing required command line argument\n");
        printHelpMenu();
        exit(0);
    }
}

/*分析输入参数*/
int get_Opt(int argc,char **argv,int *s,int *E,int* b,char *tracefileName,int *
isVerbose) {
    int c;
    while((c= getopt(argc,argv,"hvs:E:b:t:"))!=-1) {
        switch(c) {
            case 'v':
                *isVerbose=1;
```

```
                break;
            case 's':
                checkOptarg(optarg);
                *s =atoi(optarg);
                break;
            case 'E':
                checkOptarg(optarg);
                *E =atoi(optarg);
                break;
            case 'b':
                checkOptarg(optarg);
                *b =atoi(optarg);
                break;
            case 't':
                checkOptarg(optarg);
                strcpy(tracefileName,optarg);
                break;
            case 'h':
            default:
                printHelpMenu();
                exit(0);
        }
    }
    return 1;
}

/*初始化cache */
void init_SimCache(int s,int E,int b,Sim_Cache *cache) {
    if(s<0) {
        printf("invald cache sets number!\n");
        exit(0);
    }
    cache->SetNumber=1<<s;
    cache->LineNumber=E;

    cache->sets=(Set *)malloc(cache->SetNumber*sizeof(Set)) ;
    if(!cache->sets) {
        printf("Set Memory error!\n");
        exit(0);
    }

    int i,j;
    for(i=0; i<cache->SetNumber; i++) {
        cache->sets[i].lines=(Line *)malloc(E*sizeof(Line));
        if(!cache->sets[i].lines) {
            printf("Line Memeory error!\n");
            exit(0);
        }

        for(j=0; j<E; j++) {
            cache->sets[i].lines[j].valid=0;
            cache->sets[i].lines[j].LruNumber=0;
        }
    }
    return;
}

/*释放函数*/
int free_SimCache(Sim_Cache *cache) {
    int i;
    for(i=0; i<cache->SetNumber; i++) {
        free(cache->sets[i].lines);
        cache->sets[i].lines=NULL;
    }
    free(cache->sets);
    cache->sets=NULL;
    return 0;
}

/*显示各组*/
int put_Sets(Sim_Cache *cache) {
```

```
142
143         int i,j;
144         for(i=0; i<cache->SetNumber; i++) {
145             for(j=0; j<cache->LineNumber; j++) {
146                 printf("set %d:  line %d:  valid=%d, LruNumber=%d\n",i,j,cache->sets[i].
                    lines[j].valid,cache->sets[i].lines[j].LruNumber);
147             }
148         }
149         return 0;
150     }
151
152     /*Step3 Functions----*/
153     /*更新LruNumber,hit的话为最大的MAX_NUM,其他行LRU均减1 */
154     void updateLruNumber(Sim_Cache *sim_cache,int setBits,int hitIndex) {
155         sim_cache->sets[setBits].lines[hitIndex].LruNumber=MAX_NUM;
156         int j;
157         for(j=0; j<sim_cache->LineNumber; j++) {
158             if(j!=hitIndex)
159                 sim_cache->sets[setBits].lines[j].LruNumber--;
160         }
161
162     }
163
164     /* 查找某组中当前最小的LruNumber行，作为牺牲行 */
165     int findMinLruNumber(Sim_Cache *sim_cache,int setBits) {
166         int i,t;
167         int minIndex=0;
168         int minLru=MAX_NUM;
169         for(i=0; i<sim_cache->LineNumber; i++) {
170             t=sim_cache->sets[setBits].lines[i].LruNumber;
171             if(t<minLru) {
172                 minIndex=i;
173                 minLru=t;
174             }
175         }
176         return minIndex;
177     }
178
179     /*判断是否命中*/
180     int isMiss(Sim_Cache *sim_cache,int setBits,int tagBits) {
181         int i;
182         int isMiss=1;
183         for(i=0; i<sim_cache->LineNumber; i++) {
184             if(sim_cache->sets[setBits].lines[i].valid==1 && sim_cache->sets[setBits].
                lines[i].tag ==tagBits) {
185                 isMiss=0;
186                 updateLruNumber(sim_cache,setBits,i);
187                 break;
188             }
189         }
190         return isMiss;
191     }
192
193     /*更新高速缓存数据*/
194     int updateCache(Sim_Cache *sim_cache,int setBits,int tagBits) {
195         int i;
196         int isfull=1;
197         for(i=0; i<sim_cache->LineNumber; i++) {
198             if(sim_cache->sets[setBits].lines[i].valid==0) {
199                 isfull=0;
200                 break;
201             }
202         }
203         if(isfull==0) {
204             sim_cache->sets[setBits].lines[i].valid=1;
205             sim_cache->sets[setBits].lines[i].tag=tagBits;
206             updateLruNumber(sim_cache,setBits,i);
207         } else {
208             int evictionIndex=findMinLruNumber(sim_cache,setBits);
209             sim_cache->sets[setBits].lines[evictionIndex].valid=1;
210             sim_cache->sets[setBits].lines[evictionIndex].tag=tagBits;
```

```
211             updateLruNumber(sim_cache,setBits,evictionIndex);
212         }
213         return isfull;
214     }
215 
216     /*验证LRU运行相关函数*/
217     int runLru(Sim_Cache *sim_cache,int setBits,int tagBits) {
218         if(isMiss(sim_cache,setBits,tagBits) )
219             updateCache(sim_cache,setBits,tagBits);
220         return 0;
221     }
222 
223     /*
224      * main function for Csim, cachelab part A.
225      * ZHJ,C301,Yuelu,CS,2021Spr
226      */
227     int main(int argc, char *argv[]) {
228         int s,E,b,isVerbose=0;
229         char tracefileName[100];   //追踪文件
230 
231         /*step2用：  用户补充检验代码  */
232 
233 
234         /*step3用：  用户补充检验代码  */
235 
236 
237         /*step4用：  用户补充检验代码   */
238 
239 
240         return 0;
241     }
242 
```