

```

1  /*
2   * Name: Preston Tighe
3   * Program 6
4   *
5   * Command: gcc program6.c -o program6 && ./program6 &&
   cat results.txt
6   */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10 #define CLOCK_SIZE 4
11
12 typedef struct tag {
13     int s[CLOCK_SIZE][4];
14 }
15
16 QUEUE;
17 QUEUE queue;
18 int nf = 0;
19
20 int inblock(int a[CLOCK_SIZE][4], int page, char operation
   ) {
21     for (int i = 0; i < CLOCK_SIZE; i++) {
22         if (a[i][1] == page) {
23             a[i][2] = 1;
24             if (operation == 'w') {
25                 a[i][3] = 1;
26             }
27             return (1);
28         }
29     }
30     return (0);
31 }
32
33 int search(int a[CLOCK_SIZE][4]) {
34     for (int i = 0; i < CLOCK_SIZE; i++) {
35         if (a[(nf + i) % 4][2] == 0 && a[(nf + i) % 4][3]
   == 0) {
36             nf = (nf + i) % 4;
37             return (nf);
38         }
39     }
40     for (int i = 0; i < CLOCK_SIZE; i++) {
41         if (a[(nf + i) % 4][2] == 0 && a[(nf + i) % 4][3]
   == 1) {

```

```

42         nf = (nf + i) % 4;
43         return (nf);
44     }
45     a[(nf + i) % 4][2] = 0;
46 }
47 return (search(a));
48 }
49
50 void writeClockToFile(FILE * filePtr, int page, char
    operation) {
51     int pa;
52     char op;
53     pa = page;
54     op = operation;
55     fprintf(filePtr, "FRAME          PAGE          USE
    MODIFY\n");
56     if (inblock(queue.s, pa, op)) {
57         return;
58     } else if (queue.s[CLOCK_SIZE - 1][1] != -1) {
59         int j = search(queue.s);
60         queue.s[j][1] = page;
61         queue.s[j][2] = 1;
62         if (operation == 'w') {
63             queue.s[j][3] = 1;
64         } else {
65             queue.s[j][3] = 0;
66         }
67         nf = (j + 1) % 4;
68         return;
69     } else {
70         for (int i = 0; i < CLOCK_SIZE; i++) {
71             if (queue.s[i + nf][1] == -1) {
72                 queue.s[i + nf][1] = page;
73                 queue.s[i + nf][2] = 1;
74                 if (operation == 'w') {
75                     queue.s[i + nf][3] = 1;
76                 }
77                 nf = (i + nf + 1) % 4;
78                 return;
79             }
80         }
81     }
82 }
83
84 int main() {

```

```

85     int page;
86     char operation;
87     for (int i = 0; i < CLOCK_SIZE; i++) {
88         queue.s[i][0] = i;
89         queue.s[i][1] = -1;
90     }
91     for (int i = 0; i < CLOCK_SIZE; i++) {}
92     char inFileName[] = "testdata.txt";
93     FILE * inFilePtr = fopen(inFileName, "r");
94     if (inFilePtr == NULL) {
95         printf("File %s could not be opened.\n",
inFileName);
96         exit(1);
97     }
98     char outFileName[] = "results.txt";
99     FILE * outFilePtr = fopen(outFileName, "w");
100    if (outFilePtr == NULL) {
101        printf("File %s could not be opened.\n",
outFileName);
102        exit(1);
103    }
104    fscanf(inFilePtr, "%d%c", & page, & operation);
105    while (!feof(inFilePtr)) {
106        fprintf(outFilePtr, "Page referenced: %d %c\n",
page, operation);
107        writeClockToFile(outFilePtr, page, operation);
108        for (int i = 0; i < CLOCK_SIZE; i++) {
109            char spa = queue.s[i][1] < 0 ? '\0' : ' ';
110            if (i == nf) {
111                fprintf(outFilePtr, " %d          %c%d
%d          %d <- next frame\n", queue.s[i][0],
spa, queue.s[i][1], queue.s[i][2], queue.s[i][3]);
112            } else {
113                fprintf(outFilePtr, " %d          %c%d
%d          %d \n", queue.s[i][0], spa, queue.
s[i][1], queue.s[i][2], queue.s[i][3]);
114            }
115        }
116        fprintf(outFilePtr, "\n");
117        fscanf(inFilePtr, "%d%c", & page, & operation);
118    }
119    /* end while */
120    fclose(inFilePtr);
121    fclose(outFilePtr);
122    return (0);

```

```
123 };
```