

```

1 #include <stdio.h>
2 #include <limits.h>
3 #include <stdbool.h>
4 #include <stdlib.h>
5
6 void fifo(int pages, int frame, int inputs[]){
7     printf("Incoming Frame 1 Frame 2 Frame 3\n");
8     int temp[frame];
9     int pagefault = 0;
10    for (int i = 0; i < frame; i++)
11    {
12        temp[i] = -1;
13    }
14    for (int i = 0; i < pages; i++)
15    {
16        int s = 0;
17        for (int j = 0; j < frame; j++)
18        {
19            if (temp[j] == inputs[i])
20            {
21                s++;
22                pagefault--;
23            }
24        }
25        pagefault++;
26        if (pagefault <= frame && s == 0)
27        {
28            temp[i % frame] = inputs[i];
29        }
30        else if (s == 0)
31        {
32            temp[(pagefault - 1) % frame] = inputs[i];
33        }
34        printf("%d\t", inputs[i]);
35        for (int j = 0; j < frame; j++)
36        {
37            if (temp[j] != -1)
38                printf(" %d\t", temp[j]);
39            else
40                printf(" - \t");
41        }
42        printf("\n");
43    }
44    printf("Total Page Faults:\t%d\n", pagefault);
45 }
46
47
48 int checkHit(int page, int queue[], int occupied){
49     for (int i = 0; i < occupied; i++)
50     {
51         if (queue[i] == page)
52         {
53             return 1;
54         }
55     }
56     return 0;
57 }

```

```

53         return 1;
54     }
55 }
56 return 0;
57 }
58 void printFrame(int queue[], int occupied, int currentPage){
59     printf("%d\t", currentPage);
60     for (int i = 0; i < occupied; i++)
61     {
62         if (queue[i] != -1)
63             printf(" %d\t", queue[i]);
64         else
65             printf(" - \t");
66     }
67     printf("\n");
68 }
69 void lru(int pages, int frames, int incomingStream[]){
70     int queue[frames];
71     int occupied = 0;
72     int pagefault = 0;
73     int distance[frames];
74     printf("\nIncoming Frame 1 Frame 2 Frame 3\n");
75     for (int i = 0; i < pages; i++)
76     {
77         if (checkHit(incomingStream[i], queue, occupied))
78         {
79             printFrame(queue, occupied, incomingStream[i]);
80             continue;
81         }
82         if (occupied < frames)
83         {
84             queue[occupied] = incomingStream[i];
85             pagefault++;
86             occupied++;
87             printFrame(queue, occupied, incomingStream[i]);
88         }
89         else
90         {
91             int max = INT_MIN;
92             int index = 0;
93             for (int j = 0; j < frames; j++)
94             {
95
96                 distance[j] = 0;
97                 for (int k = i - 1; k >= 0; k--)
98                 {
99                     distance[j]++;
100                     if (queue[j] == incomingStream[k])
101                         break;
102                 }
103                 if (distance[j] > max)
104                 {
105                     max = distance[j];
106                     index = j;

```

```

107         }
108     }
109     queue[index] = incomingStream[i]; // Replace LRU page pagefault++;
110     printFrame(queue, occupied, incomingStream[i]);
111 }
112 }
113 printf("Total Page Faults: %d\n", pagefault);
114 }
115 bool search(int key, int fr[], int size){
116     for (int i = 0; i < size; i++)
117         if (fr[i] == key)
118             return true;
119     return false;
120 }
121 int predict(int pg[], int fr[], int pn, int index, int fn){
122     int res = -1, farthest = index;
123     for (int i = 0; i < fn; i++)
124     {
125         int j;
126         for (j = index; j < pn; j++)
127         {
128             if (fr[i] == pg[j])
129             {
130                 if (j > farthest)
131                 {
132                     farthest = j;
133                     res = i;
134                 }
135                 break;
136             }
137         }
138         if (j == pn)
139             return i;
140     }
141     return (res == -1) ? 0 : res;
142 }
143
144 void optimal(int pg[], int pn, int fn){
145     int fr[fn];
146     int pagefault = 0;
147     int occupied = 0;
148     printf("\nIncoming Frame 1 Frame 2 Frame 3\n");
149     for (int i = 0; i < pn; i++)
150     {
151         if (search(pg[i], fr, occupied))
152         {
153             printFrame(fr, occupied, pg[i]);
154             continue;
155         }
156         if (occupied < fn)
157         {
158             fr[occupied] = pg[i];
159             pagefault++;

```

```

160         occupied++;
161         printFrame(fr, occupied, pg[i]);
162     }
163     else
164     {
165         int j = predict(pg, fr, pn, i + 1, fn);
166         fr[j] = pg[i];
167         pagefault++;
168         printFrame(fr, occupied, pg[i]);
169     }
170 }
171 printf("Total Page Faults: %d\n", pagefault);
172 }
173 int main(){
174     int pages;
175     int frames;
176     int op;
177     printf("Enter the number of pages: ");
178     scanf("%d", &pages);
179     int arr[pages];
180     printf("Enter the number of frames: ");
181     scanf("%d", &frames);
182     for (int i = 0; i < pages; i++)
183     {
184         printf("Enter the input %d: ", i + 1);
185
186         scanf("%d", &arr[i]);
187     }
188     do
189     {
190         printf("\n===== \n");
191         printf("1. FIFO\n2. LRU\n3. Optimal\n4. Exit\n");
192         printf("Enter the option: ");
193         scanf("%d", &op);
194         switch (op)
195         {
196             case 1:
197                 printf("\nFirst In First Out\n");
198                 fifo(pages, frames, arr);
199                 break;
200             case 2:
201                 printf("\nLeast Recently Used\n");
202                 lru(pages, frames, arr);
203                 break;
204             case 3:
205                 printf("\nOptimal Page Replacement\n");
206                 optimal(arr, pages, frames);
207                 break;
208             case 4:
209                 break;
210             default:
211                 printf("Enter a valid option\n");
212         }
213     } while (op != 4);

```

```
213     } while (op != 4);  
214     return 0;  
215 }
```