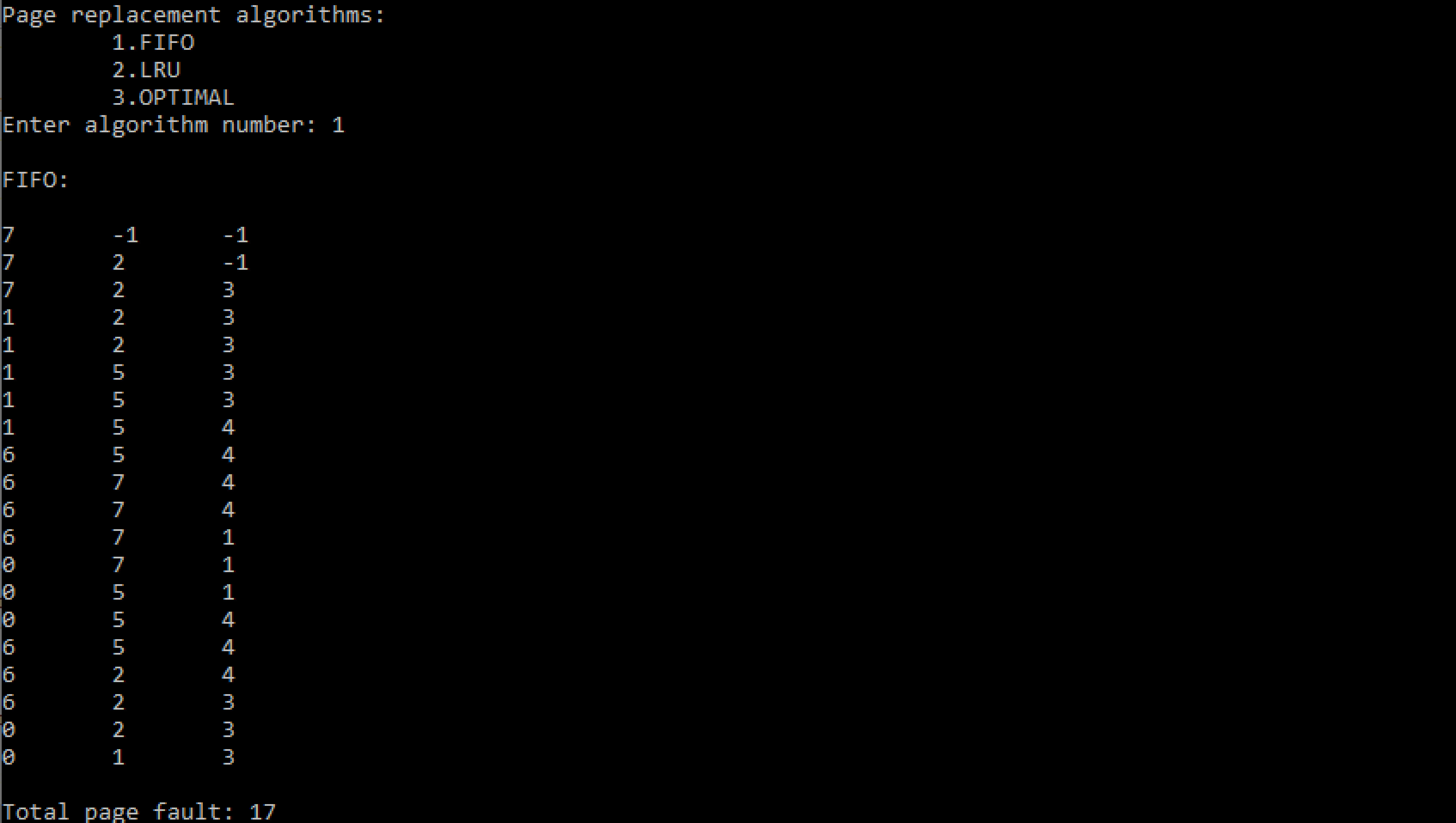
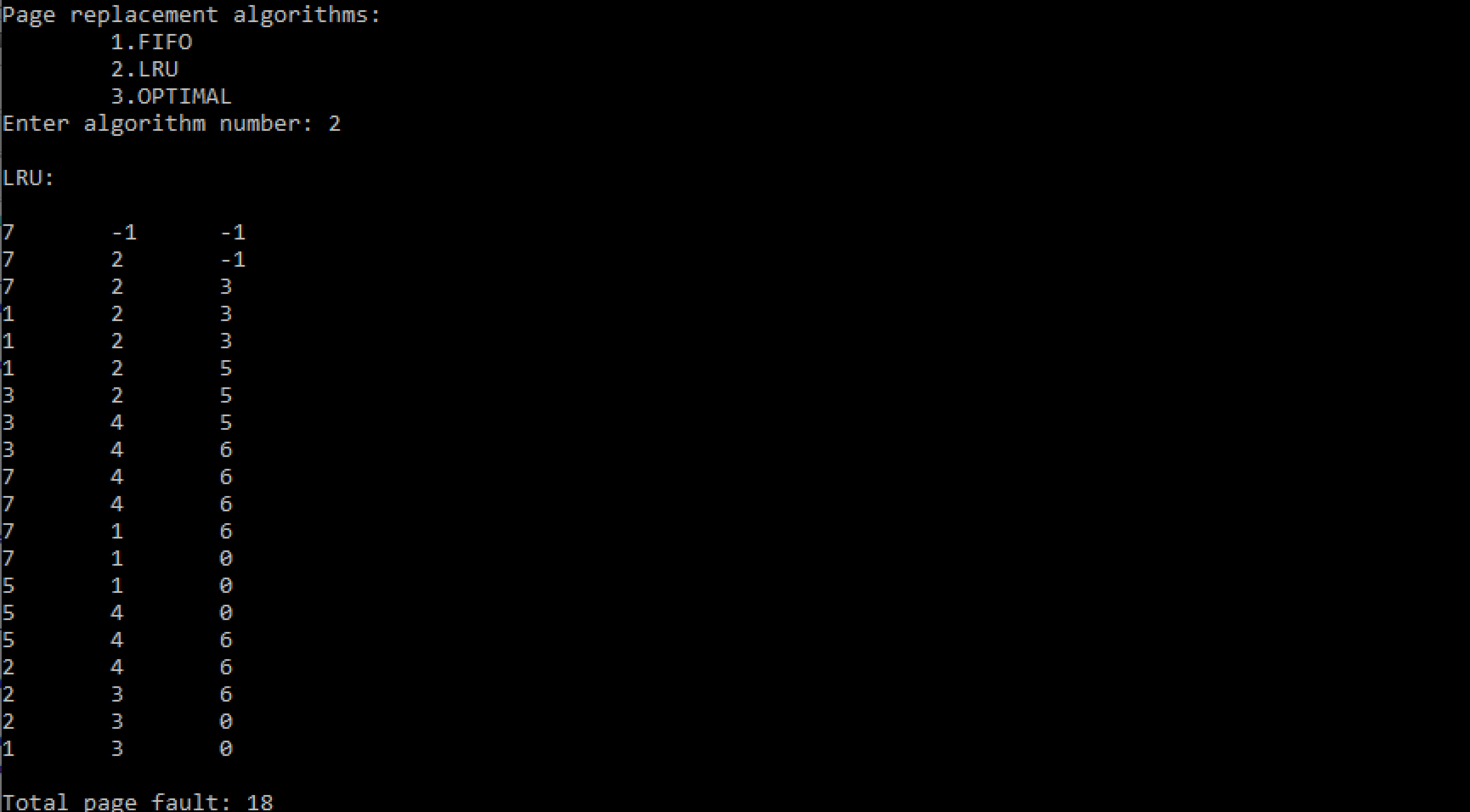
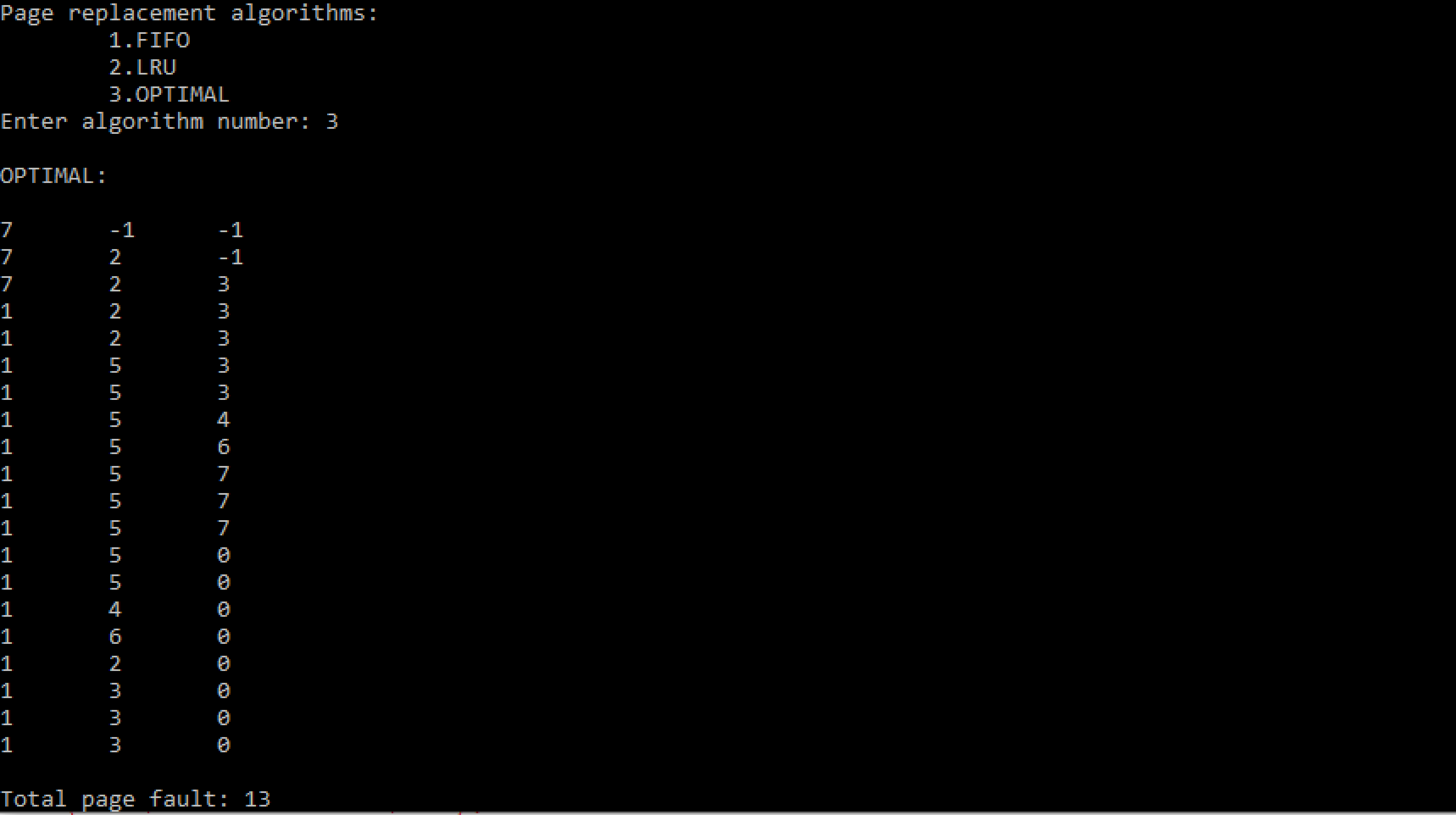
```
#include<stdio.h>

int findLRU(int time[], int n){
    int i, minimum = time[0], pos = 0;

    for(i = 1; i < n; ++i){
        if(time[i] < minimum){
            minimum = time[i];
            pos = i;
        }
    }
    return pos;
}

int fifo(int pages[], int no_of_pages, int no_of_frames){

    int faults = 0, m, n, s;

    int temp[no_of_frames];
    for(m = 0; m < no_of_frames; m++){
        temp[m] = -1;
    }

    for(m = 0; m < no_of_pages; m++){
        s = 0;
        for(n = 0; n < no_of_frames; n++){
            if(pages[m] == temp[n]){
                s++;
                faults--;
            }
        }
        faults++;
        if((faults <= no_of_frames) && (s == 0)){
            temp[m] = pages[m];
        }else if(s == 0){
            temp[(faults - 1) % no_of_frames] = pages[m];
        }
        printf("\n");
        for(n = 0; n < no_of_frames; n++){
            printf("%d\t", temp[n]);
        }
    }
    return faults;
}

int opt(int pages[], int no_of_pages, int no_of_frames){

    int flag1, flag2, flag3, i, j, k, pos, max, faults = 0;

    int frames[no_of_frames], temp[no_of_frames];

    for(i = 0; i < no_of_frames; ++i){
        frames[i] = -1;
    }

    for(i = 0; i < no_of_pages; ++i){
        flag1 = flag2 = 0;

        for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
                   flag1 = flag2 = 1;
                   break;
               }
        }
        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
                    faults++;
                    frames[j] = pages[i];
                    flag2 = 1;
                    break;
                }
            }
        }

        if(flag2 == 0){
            flag3 =0;

            for(j = 0; j < no_of_frames; ++j){
            temp[j] = -1;

            for(k = i + 1; k < no_of_pages; ++k){
                 if(frames[j] == pages[k]){
                     temp[j] = k;
                     break;
                     }
                 }
            }

            for(j = 0; j < no_of_frames; ++j){
                 if(temp[j] == -1){
                     pos = j;
                     flag3 = 1;
                     break;
                 }
            }

            if(flag3 ==0){
                max = temp[0];
                pos = 0;

                 for(j = 1; j < no_of_frames; ++j){
                     if(temp[j] > max){
                     max = temp[j];
                     pos = j;
                     }
                 }
            }
            frames[pos] = pages[i];
            faults++;
        }

        printf("\n");

        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }

   return faults;
}

int lru(int pages[], int no_of_pages, int no_of_frames){

    int counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;
    int frames[no_of_frames];
    for(i = 0; i < no_of_frames; ++i){
        frames[i] = -1;
    }

    for(i = 0; i < no_of_pages; ++i){
        flag1 = flag2 = 0;

        for(j = 0; j < no_of_frames; ++j){
            if(frames[j] == pages[i]){
                counter++;
                time[j] = counter;
                flag1 = flag2 = 1;
                break;
            }
        }

        if(flag1 == 0){
            for(j = 0; j < no_of_frames; ++j){
                if(frames[j] == -1){
                    counter++;
                    faults++;
                    frames[j] = pages[i];
                    time[j] = counter;
                    flag2 = 1;
                    break;
                }
            }
         }

        if(flag2 == 0){
             pos = findLRU(time, no_of_frames);
             counter++;
             faults++;
             frames[pos] = pages[i];
             time[pos] = counter;
        }

        printf("\n");

        for(j = 0; j < no_of_frames; ++j){
            printf("%d\t", frames[j]);
        }
    }
    return faults;
}

int main(){

    int referenceString[20] = {7, 2, 3, 1, 2, 5, 3, 4, 6, 7, 7, 1, 0, 5, 4, 6,
2, 3, 0, 1};
    int pageFaults = 0, m, n, s, frames = 3;
    int pages = sizeof(referenceString)/sizeof(int);


    printf("Page replacement algorithms: \n\t1.FIFO\n\t2.LRU\n\t3.OPTIMAL\n");
    int choose;
    printf("Enter algorithm number: ");
    scanf("%d", &choose);

    int pageFault = 0;
    printf("\n");
    switch(choose){
        case 1: {
            printf("FIFO: \n");
            pageFault = fifo(referenceString, pages, frames);
            break;
        }
        case 2: {
            printf("LRU: \n");
            pageFault = lru(referenceString, pages, frames);
            break;
        }
        case 3: {
            printf("OPTIMAL: \n");
            pageFault = opt(referenceString, pages, frames);
            break;
        }
        default: {
            printf("Wrong algorithm number. Try again.");
            break;
        }
    }

    printf("\n\nTotal page fault: %d\n", pageFault);

    return 0;
}
```



```
Page replacement algorithms:
        1.FIFO
        2.LRU
        3.OPTIMAL
Enter algorithm number: 1

FIFO:

7       -1      -1
7       2       -1
7       2       3
1       2       3
1       2       3
1       5       3
1       5       3
1       5       4
6       5       4
6       7       4
6       7       4
6       7       1
0       7       1
0       5       1
0       5       4
6       5       4
6       2       4
6       2       3
0       2       3
0       1       3

Total page fault: 17
```



```
Page replacement algorithms:
        1.FIFO
        2.LRU
        3.OPTIMAL
Enter algorithm number: 2

LRU:

7       -1      -1
7       2       -1
7       2       3
1       2       3
1       2       3
1       2       5
3       2       5
3       4       5
3       4       6
7       4       6
7       4       6
7       1       6
7       1       0
5       1       0
5       4       0
5       4       6
2       4       6
2       3       6
2       3       0
1       3       0

Total page fault: 18
```



```
Page replacement algorithms:
        1.FIFO
        2.LRU
        3.OPTIMAL
Enter algorithm number: 3

OPTIMAL:

7       -1      -1
7       2       -1
7       2       3
1       2       3
1       2       3
1       5       3
1       5       3
1       5       4
1       5       6
1       5       7
1       5       7
1       5       7
1       5       0
1       5       0
1       4       0
1       6       0
1       2       0
1       3       0
1       3       0
1       3       0

Total page fault: 13
```