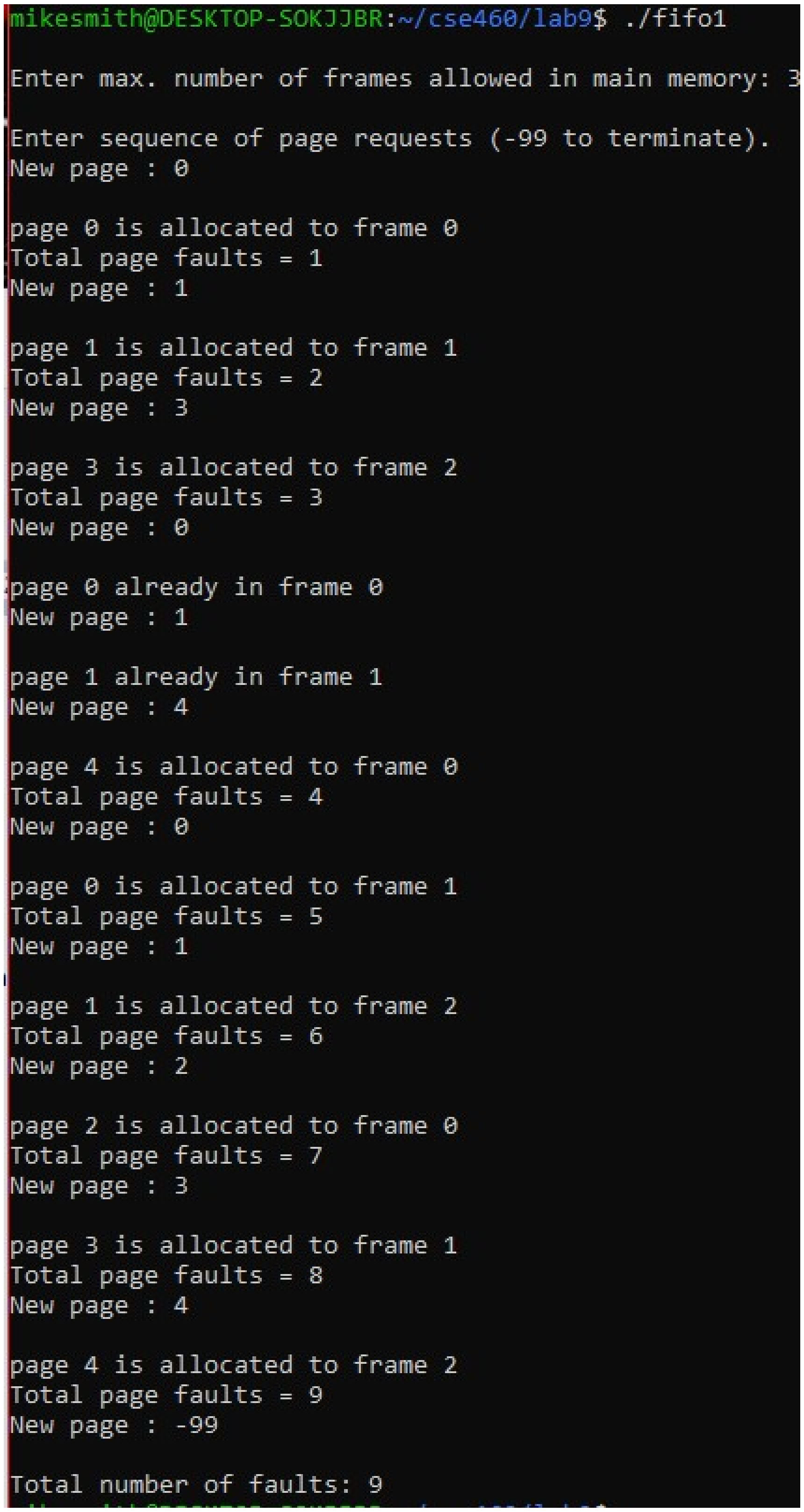
Michael Smith

Hon Huynh

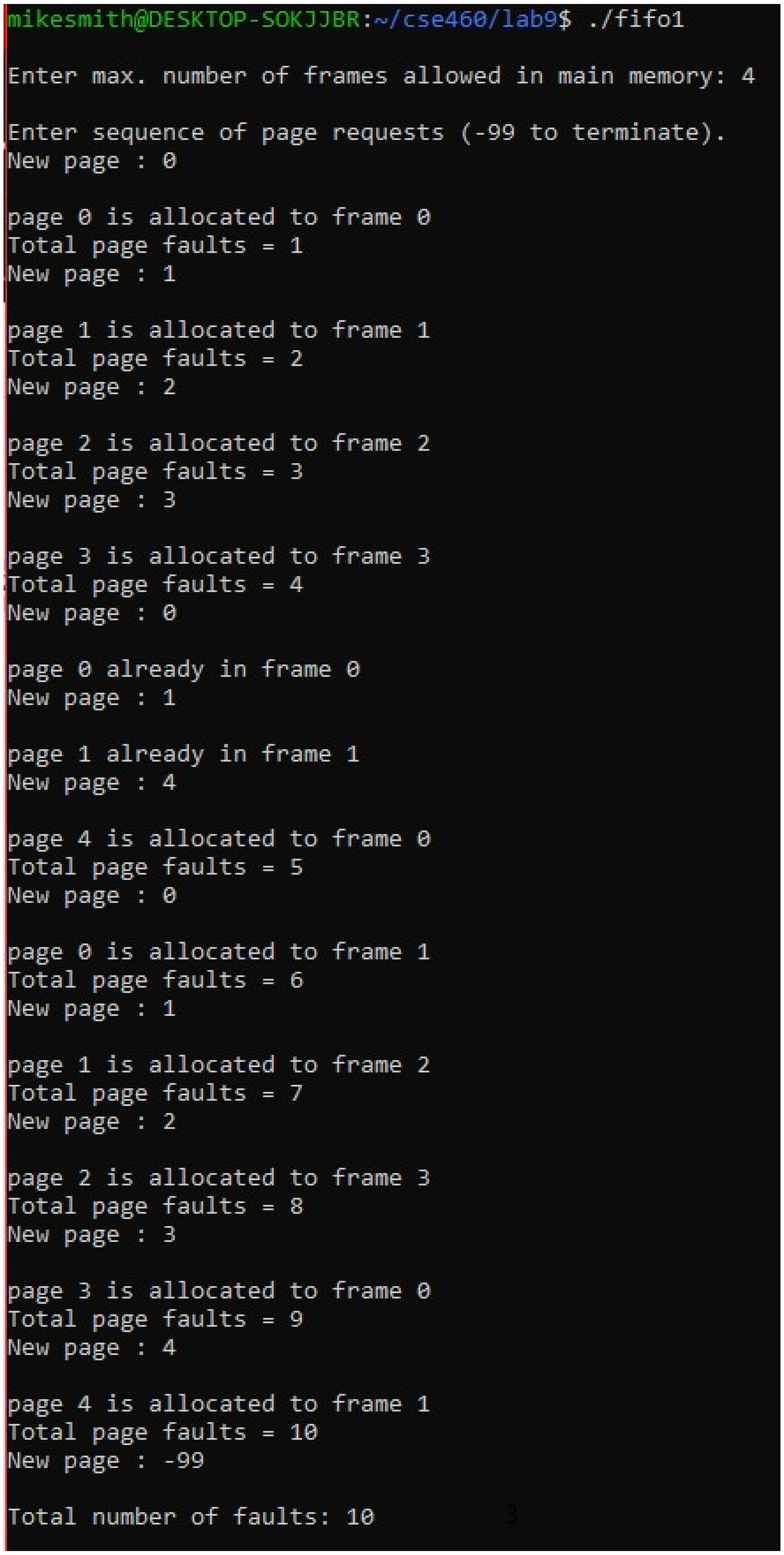
CSE 460

Lab 9

1. First in First Out (FIFO) Replacement



```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./fifo1

Enter max. number of frames allowed in main memory: 3

Enter sequence of page requests (-99 to terminate).
New page : 0

page 0 is allocated to frame 0
Total page faults = 1
New page : 1

page 1 is allocated to frame 1
Total page faults = 2
New page : 3

page 3 is allocated to frame 2
Total page faults = 3
New page : 0

page 0 already in frame 0
New page : 1

page 1 already in frame 1
New page : 4

page 4 is allocated to frame 0
Total page faults = 4
New page : 0

page 0 is allocated to frame 1
Total page faults = 5
New page : 1

page 1 is allocated to frame 2
Total page faults = 6
New page : 2

page 2 is allocated to frame 0
Total page faults = 7
New page : 3

page 3 is allocated to frame 1
Total page faults = 8
New page : 4

page 4 is allocated to frame 2
Total page faults = 9
New page : -99

Total number of faults: 9
```

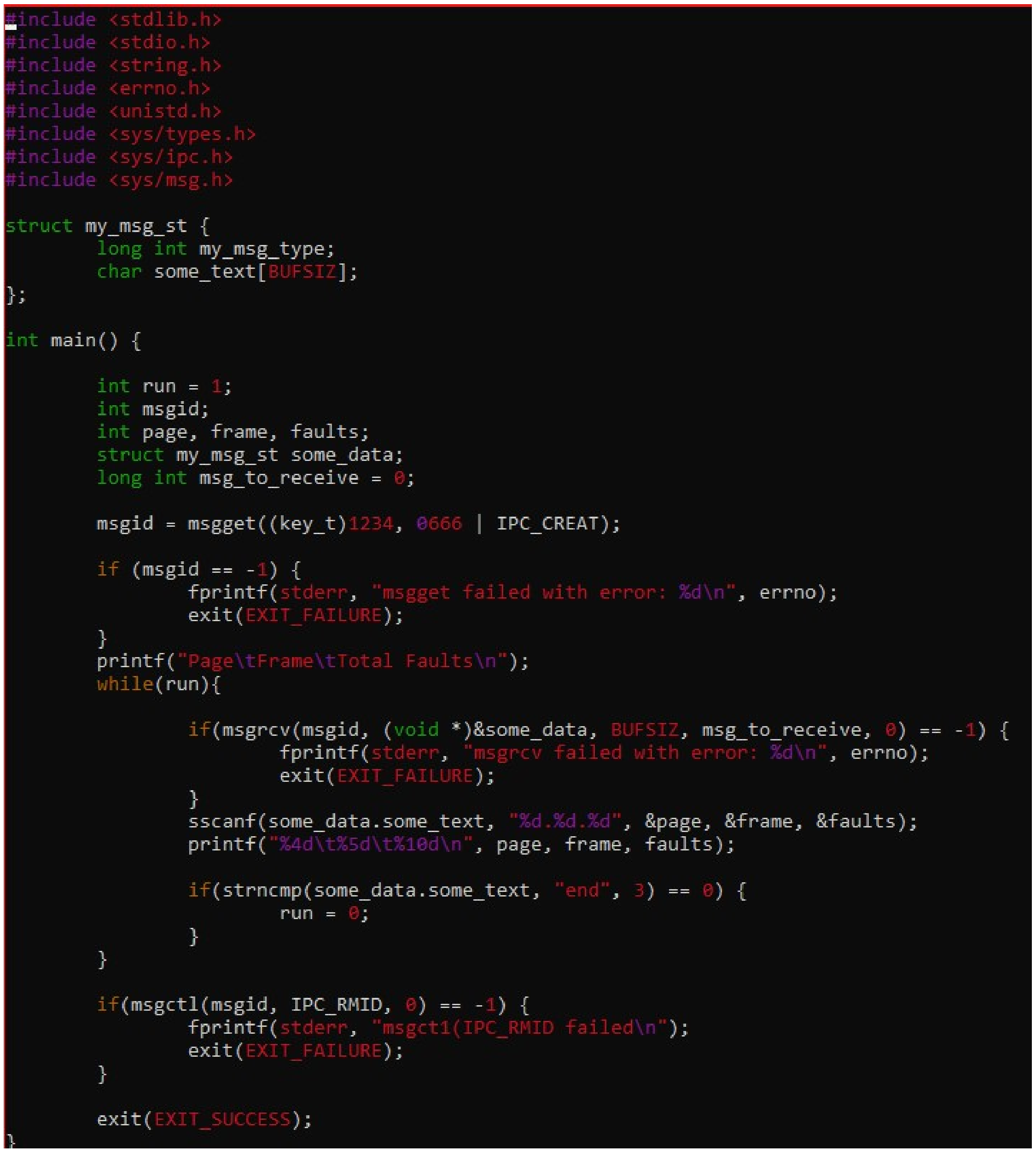


```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./fifo1

Enter max. number of frames allowed in main memory: 4

Enter sequence of page requests (-99 to terminate).
New page : 0

page 0 is allocated to frame 0
Total page faults = 1
New page : 1

page 1 is allocated to frame 1
Total page faults = 2
New page : 2

page 2 is allocated to frame 2
Total page faults = 3
New page : 3

page 3 is allocated to frame 3
Total page faults = 4
New page : 0

page 0 already in frame 0
New page : 1

page 1 already in frame 1
New page : 4

page 4 is allocated to frame 0
Total page faults = 5
New page : 0

page 0 is allocated to frame 1
Total page faults = 6
New page : 1

page 1 is allocated to frame 2
Total page faults = 7
New page : 2

page 2 is allocated to frame 3
Total page faults = 8
New page : 3

page 3 is allocated to frame 0
Total page faults = 9
New page : 4

page 4 is allocated to frame 1
Total page faults = 10
New page : -99

Total number of faults: 10
```

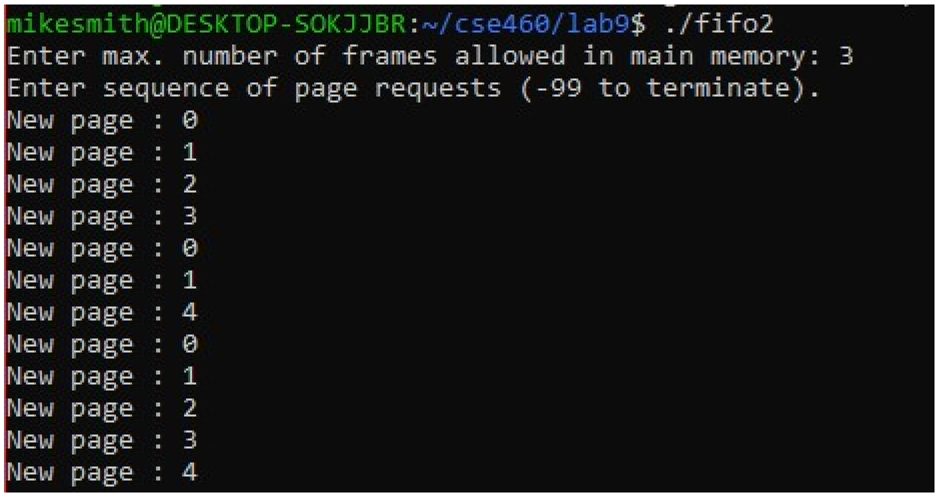
Yes the Belady's anomaly was observed when the 3 page fram gave 9 faults while teh 4 page fault gave 10. As well as the specific page reference when there wasn't a page fault.

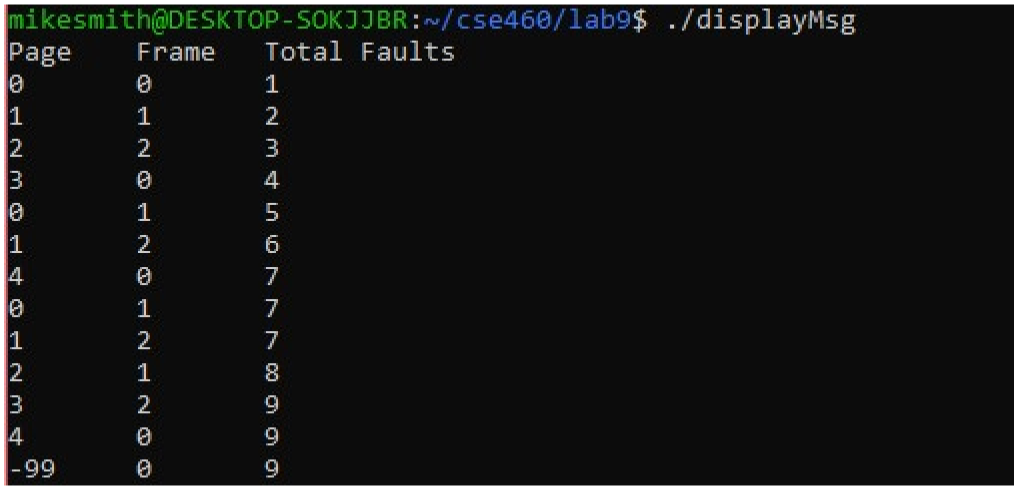
2. Multithreads for FIFO Program

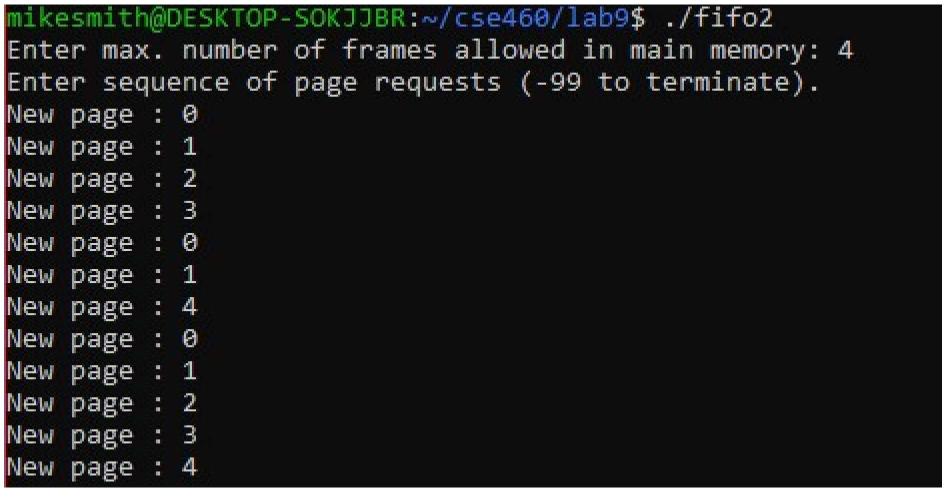
displayMessage

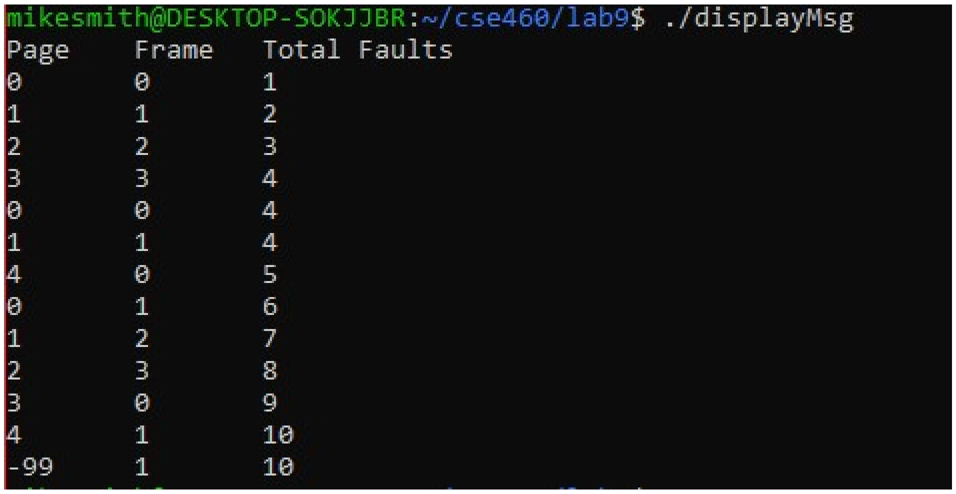


```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>

struct my_msg_st {
        long int my_msg_type;
        char some_text[BUFSIZ];
};

int main() {

        int run = 1;
        int msgid;
        int page, frame, faults;
        struct my_msg_st some_data;
        long int msg_to_receive = 0;

        msgid = msgget((key_t)1234, 0666 | IPC_CREAT);

        if (msgid == -1) {
                fprintf(stderr, "msgget failed with error: %d\n", errno);
                exit(EXIT_FAILURE);
        }
        printf("Page\tFrame\tTotal Faults\n");
        while(run){

                if(msgrcv(msgid, (void *)&some_data, BUFSIZ, msg_to_receive, 0) == -1) {
                        fprintf(stderr, "msgrcv failed with error: %d\n", errno);
                        exit(EXIT_FAILURE);
                }
                sscanf(some_data.some_text, "%d.%d.%d", &page, &frame, &faults);
                printf("%4d\t%5d\t%10d\n", page, frame, faults);

                if(strncmp(some_data.some_text, "end", 3) == 0) {
                        run = 0;
                }
        }

        if(msgctl(msgid, IPC_RMID, 0) == -1) {
                fprintf(stderr, "msgctl(IPC_RMID failed\n");
                exit(EXIT_FAILURE);
        }

        exit(EXIT_SUCCESS);
}
```


```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./fifo2
Enter max. number of frames allowed in main memory: 3
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
```


```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page    Frame   Total Faults
0       0       1
1       1       2
2       2       3
3       0       4
0       1       5
1       2       6
4       0       7
0       1       7
1       2       7
2       1       8
3       2       9
4       0       9
-99     0       9
```

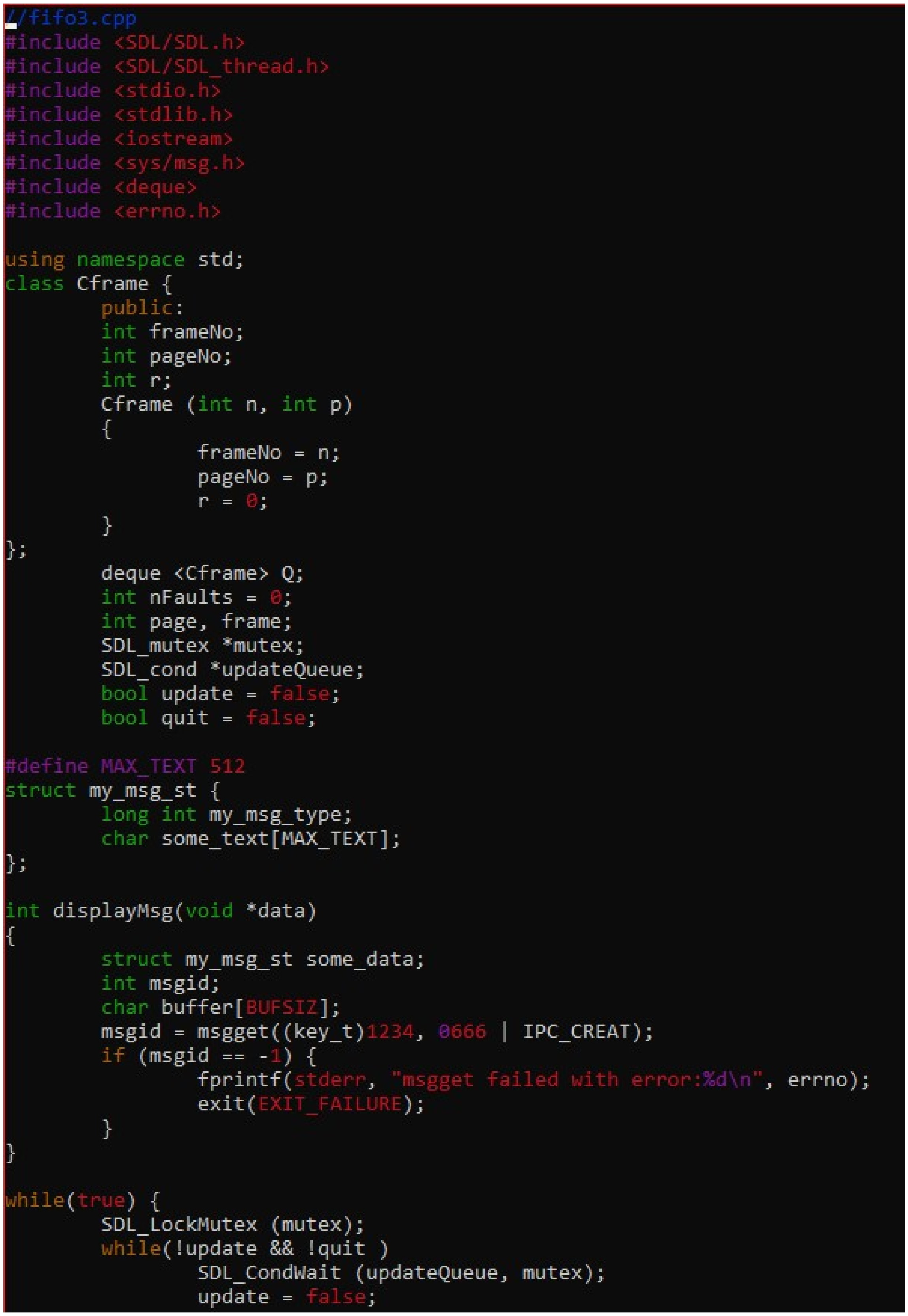

```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./fifo2
Enter max. number of frames allowed in main memory: 4
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
```



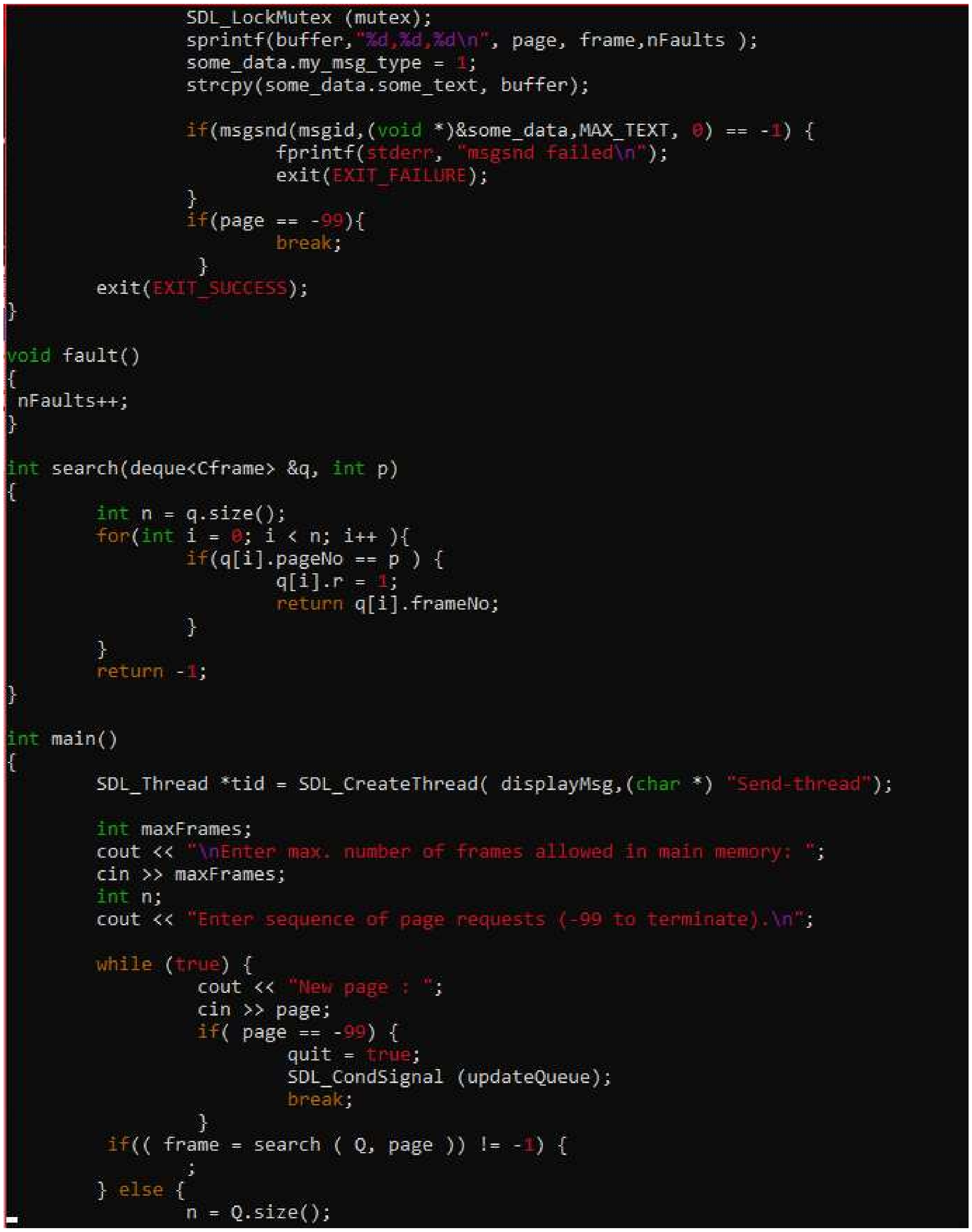
```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page    Frame   Total Faults
0       0       1
1       1       2
2       2       3
3       3       4
0       0       4
1       1       4
4       0       5
0       1       6
1       2       7
2       3       8
3       0       9
4       1       10
-99     1       10
```

3. Implement one of the following, second chance or LRU:

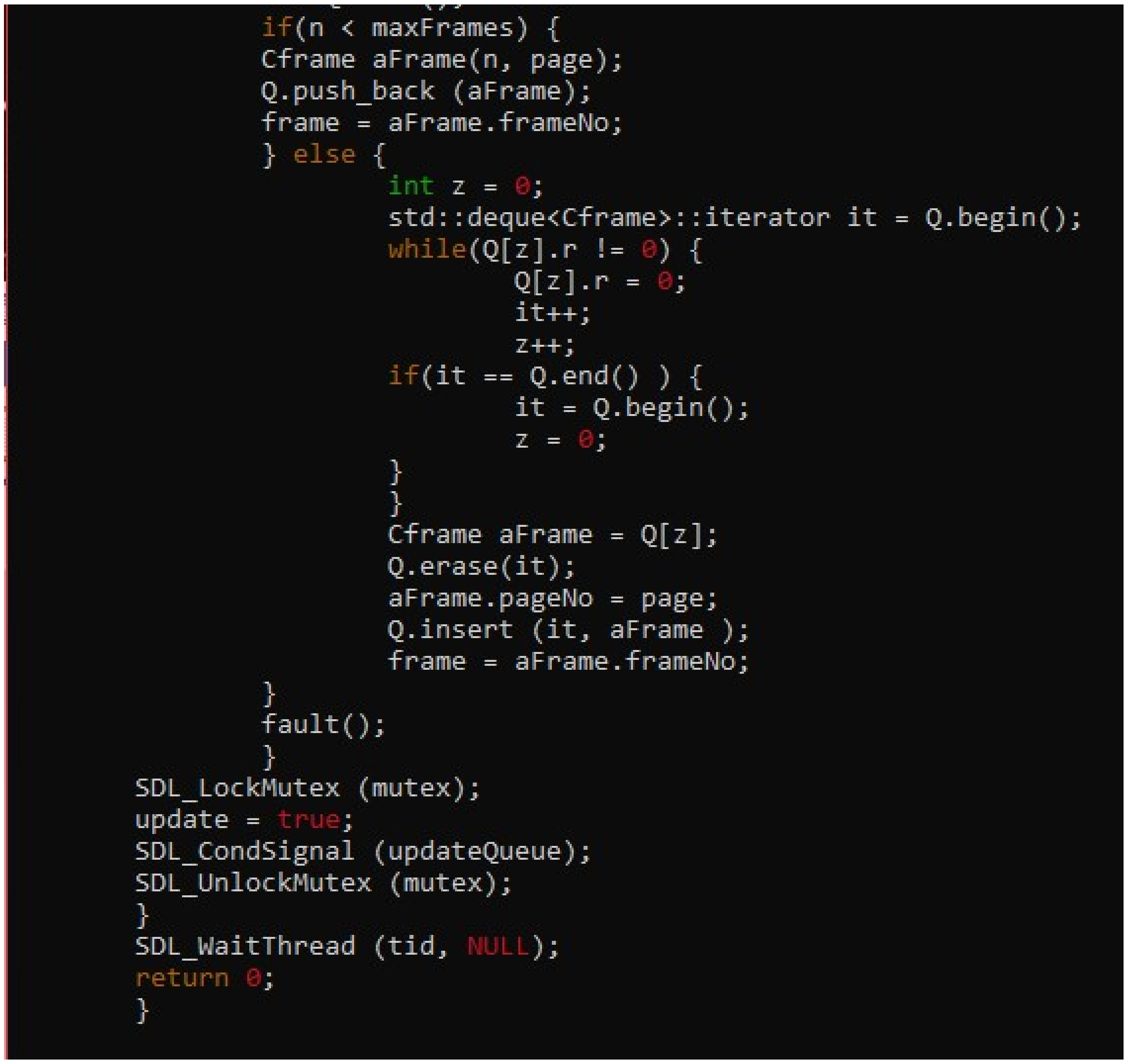
a. Second Chance



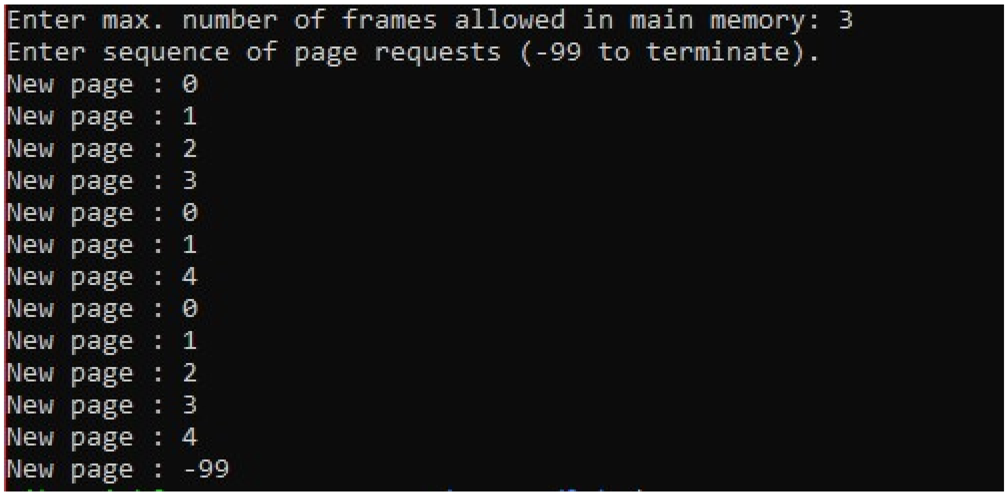
```
//fifo3.cpp
#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <sys/msg.h>
#include <deque>
#include <errno.h>

using namespace std;
class Cframe {
        public:
        int frameNo;
        int pageNo;
        int r;
        Cframe (int n, int p)
        {
                frameNo = n;
                pageNo = p;
                r = 0;
        }
};
        deque <Cframe> Q;
        int nFaults = 0;
        int page, frame;
        SDL_mutex *mutex;
        SDL_cond *updateQueue;
        bool update = false;
        bool quit = false;

#define MAX_TEXT 512
struct my_msg_st {
        long int my_msg_type;
        char some_text[MAX_TEXT];
};

int displayMsg(void *data)
{
        struct my_msg_st some_data;
        int msgid;
        char buffer[BUFSIZ];
        msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
        if (msgid == -1) {
                fprintf(stderr, "msgget failed with error:%d\n", errno);
                exit(EXIT_FAILURE);
        }
}

while(true) {
        SDL_LockMutex (mutex);
        while(!update && !quit )
                SDL_CondWait (updateQueue, mutex);
                update = false;
```



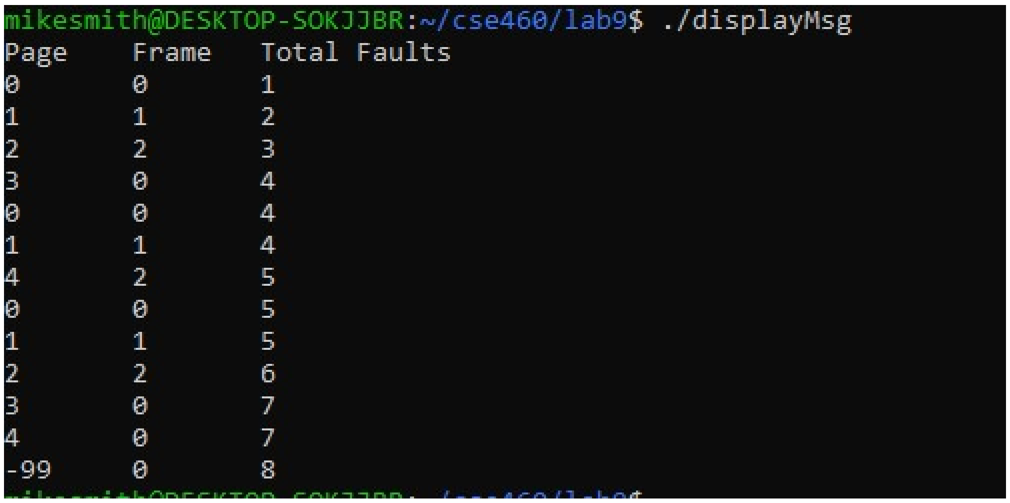
```
                SDL_LockMutex (mutex);
                sprintf(buffer,"%d,%d,%d\n", page, frame,nFaults );
                some_data.my_msg_type = 1;
                strcpy(some_data.some_text, buffer);

                if(msgsnd(msgid,(void *)&some_data,MAX_TEXT, 0) == -1) {
                        fprintf(stderr, "msgsnd failed\n");
                        exit(EXIT_FAILURE);
                }
                if(page == -99){
                        break;
                 }
        exit(EXIT_SUCCESS);
}

void fault()
{
 nFaults++;
}

int search(deque<Cframe> &q, int p)
{
        int n = q.size();
        for(int i = 0; i < n; i++ ){
                if(q[i].pageNo == p ) {
                        q[i].r = 1;
                        return q[i].frameNo;
                }
        }
        return -1;
}

int main()
{
        SDL_Thread *tid = SDL_CreateThread( displayMsg,(char *) "Send-thread");

        int maxFrames;
        cout << "\nEnter max. number of frames allowed in main memory: ";
        cin >> maxFrames;
        int n;
        cout << "Enter sequence of page requests (-99 to terminate).\n";

        while (true) {
                 cout << "New page : ";
                 cin >> page;
                 if( page == -99) {
                         quit = true;
                         SDL_CondSignal (updateQueue);
                         break;
                 }
         if(( frame = search ( Q, page )) != -1) {
                ;
        } else {
                n = Q.size();
```



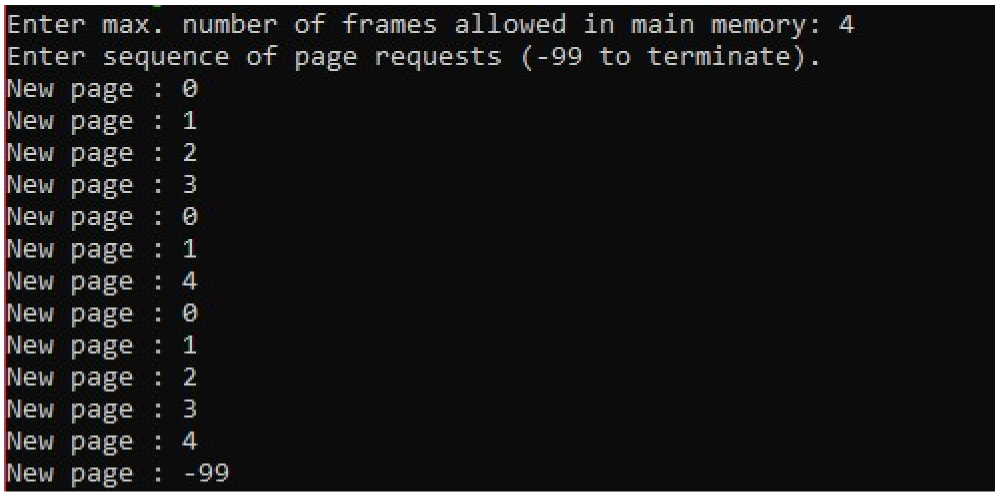
```
                if(n < maxFrames) {
                Cframe aFrame(n, page);
                Q.push_back (aFrame);
                frame = aFrame.frameNo;
                } else {
                        int z = 0;
                        std::deque<Cframe>::iterator it = Q.begin();
                        while(Q[z].r != 0) {
                                Q[z].r = 0;
                                it++;
                                z++;
                        if(it == Q.end() ) {
                                it = Q.begin();
                                z = 0;
                        }
                        }
                        Cframe aFrame = Q[z];
                        Q.erase(it);
                        aFrame.pageNo = page;
                        Q.insert (it, aFrame );
                        frame = aFrame.frameNo;
                }
                fault();
                }
        SDL_LockMutex (mutex);
        update = true;
        SDL_CondSignal (updateQueue);
        SDL_UnlockMutex (mutex);
        }
        SDL_WaitThread (tid, NULL);
        return 0;
        }
```



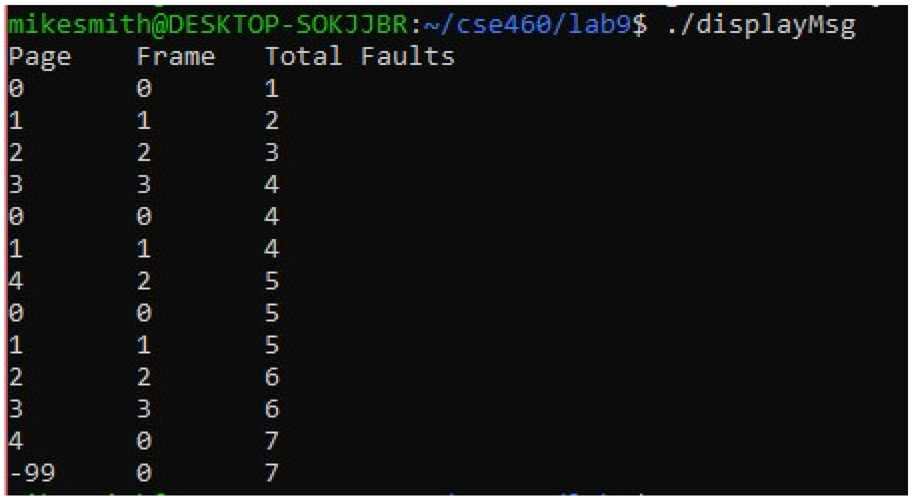
```
Enter max. number of frames allowed in main memory: 3
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
New page : -99
```



```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page    Frame   Total Faults
0       0       1
1       1       2
2       2       3
3       0       4
0       0       4
1       1       4
4       2       5
0       0       5
1       1       5
2       2       6
3       0       7
4       0       7
-99     0       8
```

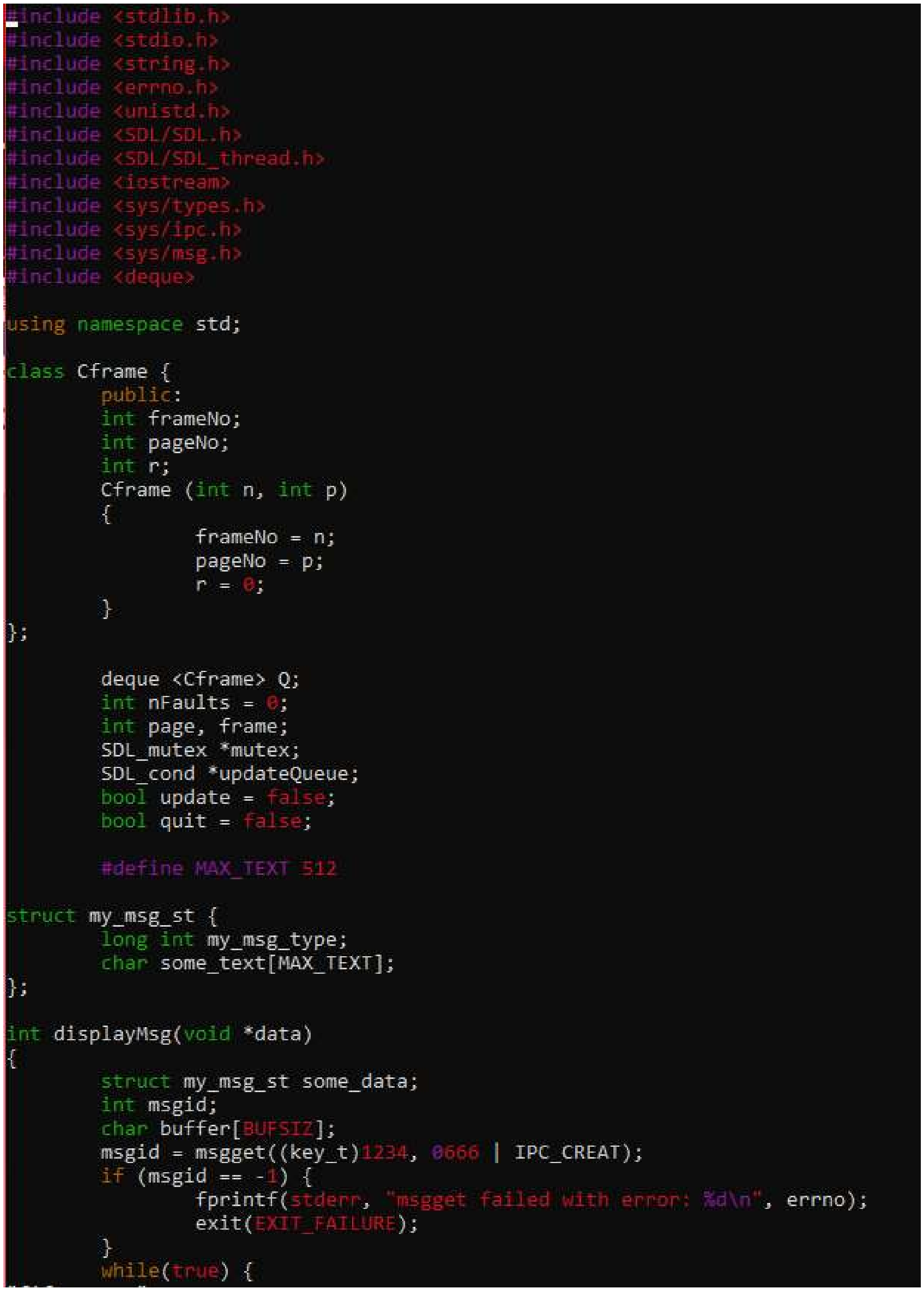


```
Enter max. number of frames allowed in main memory: 4
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
New page : -99
```

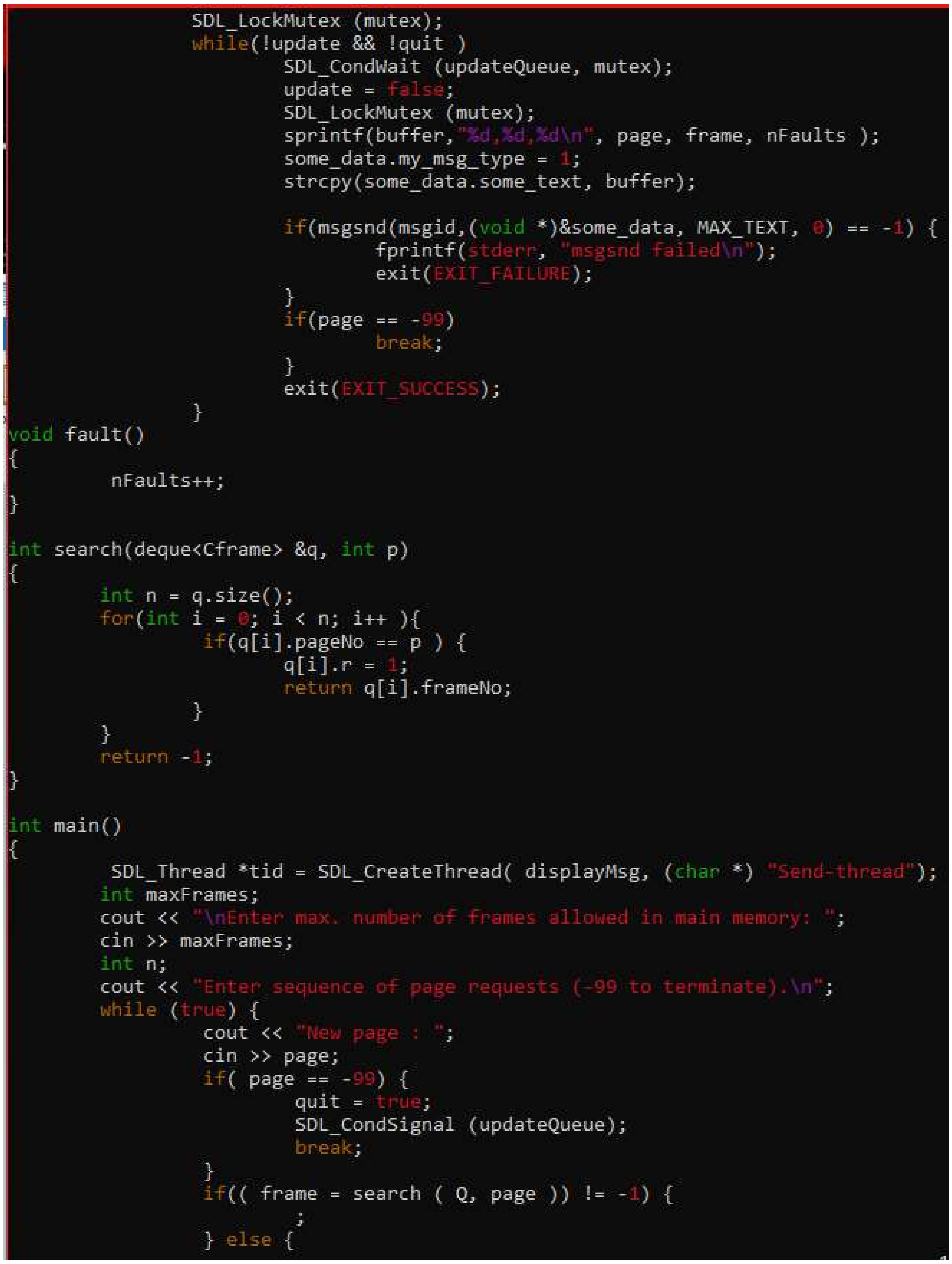


```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page    Frame   Total Faults
0       0       1
1       1       2
2       2       3
3       3       4
0       0       4
1       1       4
4       2       5
0       0       5
1       1       5
2       2       6
3       3       6
4       0       7
-99     0       7
```

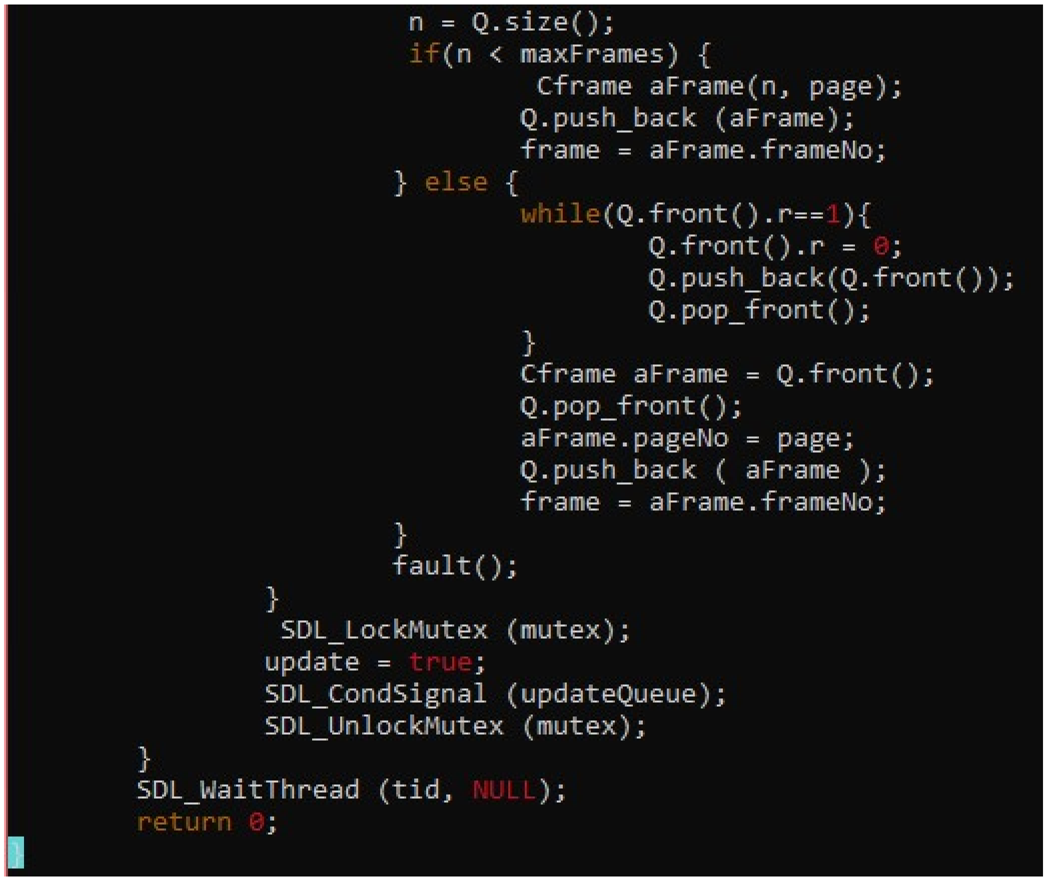
b. Least Recently Used Page Replacement



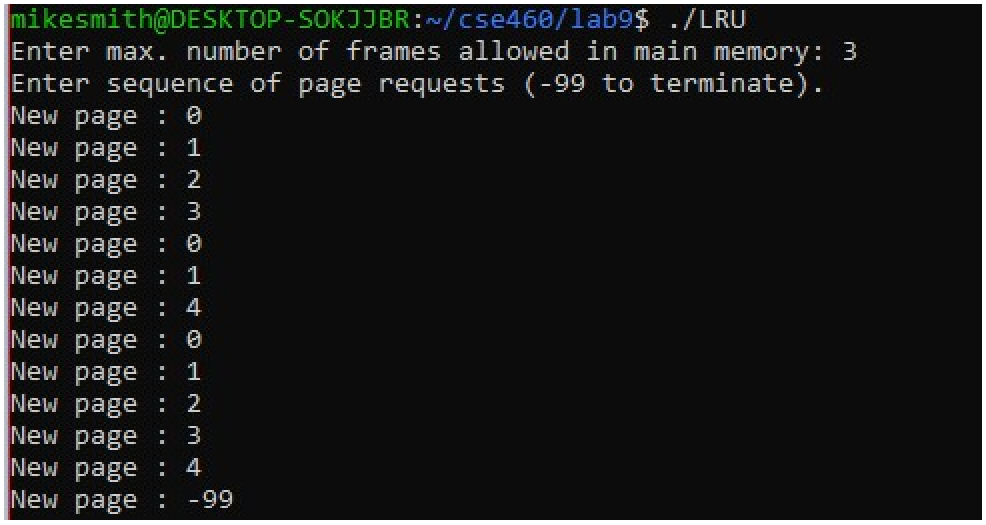
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <unistd.h>
#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <iostream>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/msg.h>
#include <deque>

using namespace std;

class Cframe {
        public:
        int frameNo;
        int pageNo;
        int r;
        Cframe (int n, int p)
        {
                frameNo = n;
                pageNo = p;
                r = 0;
        }
};

        deque <Cframe> Q;
        int nFaults = 0;
        int page, frame;
        SDL_mutex *mutex;
        SDL_cond *updateQueue;
        bool update = false;
        bool quit = false;

        #define MAX_TEXT 512

struct my_msg_st {
        long int my_msg_type;
        char some_text[MAX_TEXT];
};

int displayMsg(void *data)
{
        struct my_msg_st some_data;
        int msgid;
        char buffer[BUFSIZ];
        msgid = msgget((key_t)1234, 0666 | IPC_CREAT);
        if (msgid == -1) {
                fprintf(stderr, "msgget failed with error: %d\n", errno);
                exit(EXIT_FAILURE);
        }
        while(true) {
```



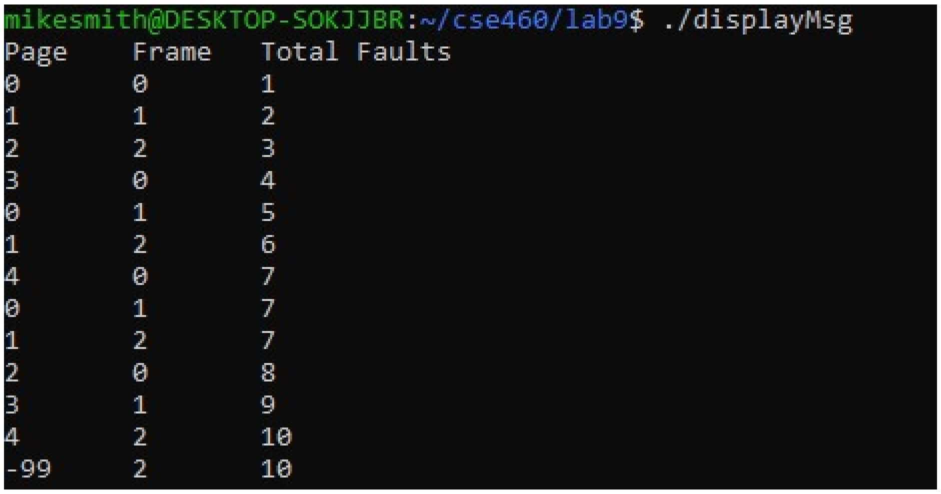
```
                SDL_LockMutex (mutex);
                while(!update && !quit )
                        SDL_CondWait (updateQueue, mutex);
                        update = false;
                        SDL_LockMutex (mutex);
                        sprintf(buffer,"%d,%d,%d\n", page, frame, nFaults );
                        some_data.my_msg_type = 1;
                        strcpy(some_data.some_text, buffer);

                        if(msgsnd(msgid,(void *)&some_data, MAX_TEXT, 0) == -1) {
                                fprintf(stderr, "msgsnd failed\n");
                                exit(EXIT_FAILURE);
                        }
                        if(page == -99)
                                break;
                        }
                        exit(EXIT_SUCCESS);
                }
void fault()
{
         nFaults++;
}

int search(deque<Cframe> &q, int p)
{
        int n = q.size();
        for(int i = 0; i < n; i++ ){
                 if(q[i].pageNo == p ) {
                        q[i].r = 1;
                        return q[i].frameNo;
                }
        }
        return -1;
}

int main()
{
         SDL_Thread *tid = SDL_CreateThread( displayMsg, (char *) "Send-thread");
        int maxFrames;
        cout << "\nEnter max. number of frames allowed in main memory: ";
        cin >> maxFrames;
        int n;
        cout << "Enter sequence of page requests (-99 to terminate).\n";
        while (true) {
                 cout << "New page : ";
                 cin >> page;
                 if( page == -99) {
                         quit = true;
                         SDL_CondSignal (updateQueue);
                         break;
                 }
                 if(( frame = search ( Q, page )) != -1) {
                         ;
                 } else {
```



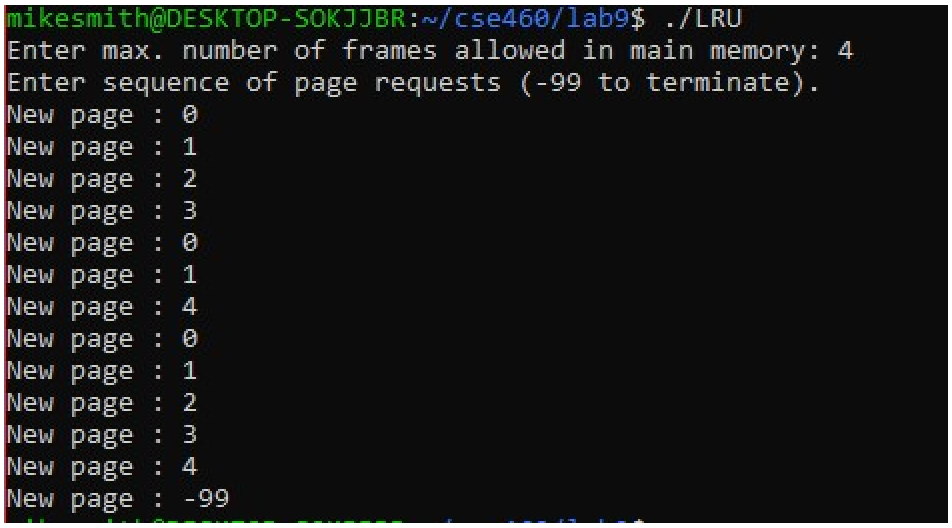
```
                         n = Q.size();
                         if(n < maxFrames) {
                                 Cframe aFrame(n, page);
                                Q.push_back (aFrame);
                                frame = aFrame.frameNo;
                        } else {
                                while(Q.front().r==1){
                                        Q.front().r = 0;
                                        Q.push_back(Q.front());
                                        Q.pop_front();
                                }
                                Cframe aFrame = Q.front();
                                Q.pop_front();
                                aFrame.pageNo = page;
                                Q.push_back ( aFrame );
                                frame = aFrame.frameNo;
                        }
                        fault();
                }
                 SDL_LockMutex (mutex);
                update = true;
                SDL_CondSignal (updateQueue);
                SDL_UnlockMutex (mutex);
        }
        SDL_WaitThread (tid, NULL);
        return 0;
}
```



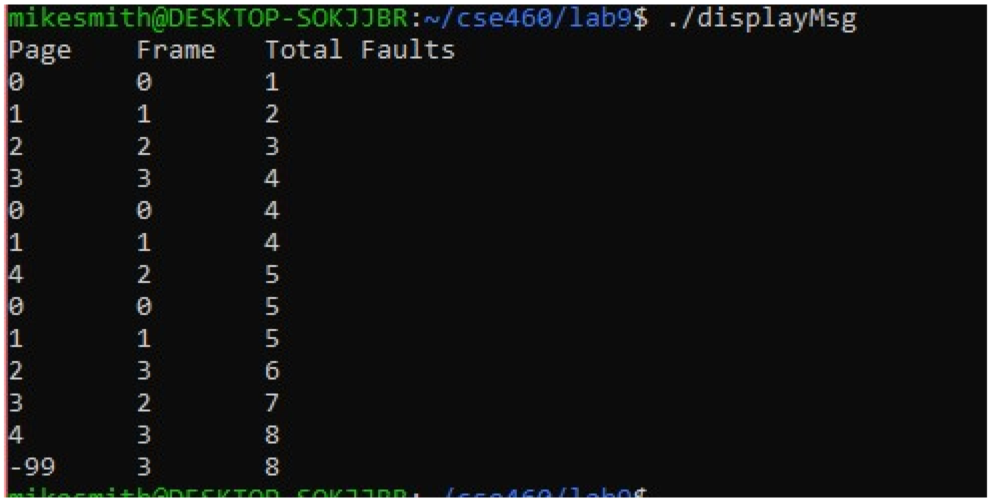
```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./LRU
Enter max. number of frames allowed in main memory: 3
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
New page : -99
```



```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page     Frame    Total Faults
0        0        1
1        1        2
2        2        3
3        0        4
0        1        5
1        2        6
4        0        7
0        1        7
1        2        7
2        0        8
3        1        9
4        2        10
-99      2        10
```



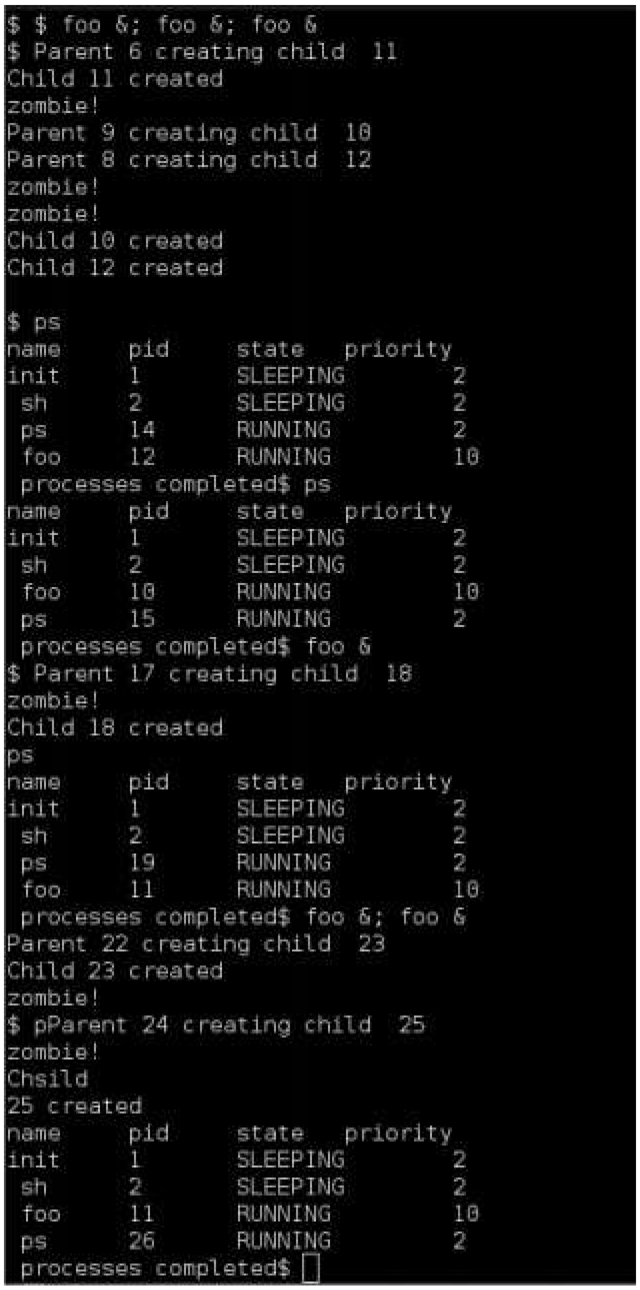
```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./LRU
Enter max. number of frames allowed in main memory: 4
Enter sequence of page requests (-99 to terminate).
New page : 0
New page : 1
New page : 2
New page : 3
New page : 0
New page : 1
New page : 4
New page : 0
New page : 1
New page : 2
New page : 3
New page : 4
New page : -99
```



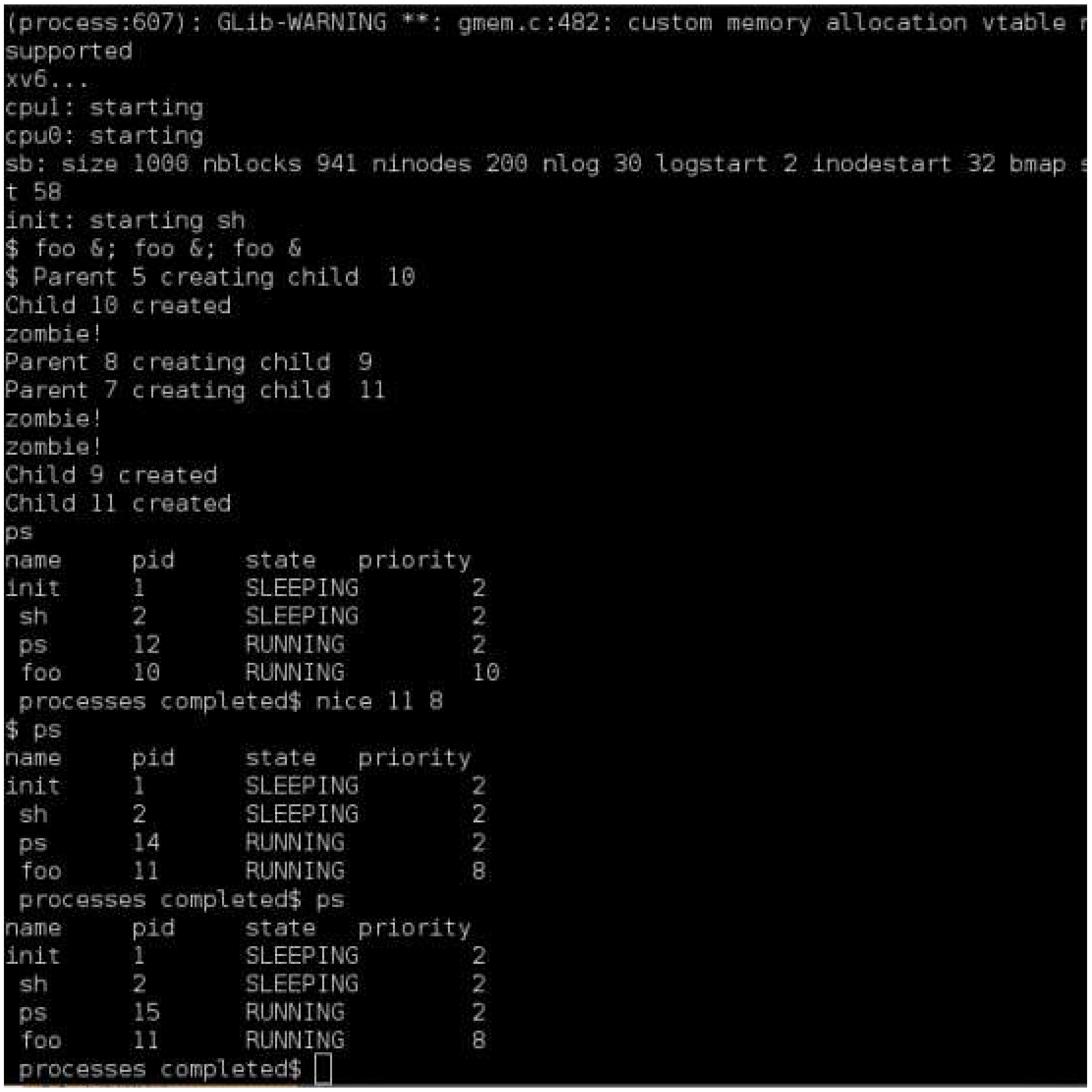
```
mikesmith@DESKTOP-SOKJJBR:~/cse460/lab9$ ./displayMsg
Page     Frame    Total Faults
0        0        1
1        1        2
2        2        3
3        3        4
0        0        4
1        1        4
4        2        5
0        0        5
1        1        5
2        3        6
3        2        7
4        3        8
-99      3        8
```

When comparing all the programs together, the faults are the least in Fifo3 when compared to fifo2, fifo, and lru.

4. XV6 Process Priority



```
$ $ foo &; foo &; foo &
$ Parent 6 creating child  11
Child 11 created
zombie!
Parent 9 creating child  10
Parent 8 creating child  12
zombie!
zombie!
Child 10 created
Child 12 created

$ ps
name     pid     state   priority
init     1       SLEEPING       2
 sh      2       SLEEPING       2
 ps      14      RUNNING        2
 foo     12      RUNNING        10
 processes completed$ ps
name     pid     state   priority
init     1       SLEEPING       2
 sh      2       SLEEPING       2
 foo     10      RUNNING        10
 ps      15      RUNNING        2
 processes completed$ foo &
$ Parent 17 creating child  18
zombie!
Child 18 created
ps
name     pid     state   priority
init     1       SLEEPING       2
 sh      2       SLEEPING       2
 ps      19      RUNNING        2
 foo     11      RUNNING        10
 processes completed$ foo &; foo &
Parent 22 creating child  23
Child 23 created
zombie!
$ pParent 24 creating child  25
zombie!
Chsild
25 created
name     pid     state   priority
init     1       SLEEPING       2
 sh      2       SLEEPING       2
 foo     11      RUNNING        10
 ps      26      RUNNING        2
 processes completed$ 
```



```
(process:607): GLib-WARNING **: gmem.c:482: custom memory allocation vtable r
supported
xv6...
cpu1: starting
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap s
t 58
init: starting sh
$ foo &; foo &; foo &
$ Parent 5 creating child  10
Child 10 created
zombie!
Parent 8 creating child  9
Parent 7 creating child  11
zombie!
zombie!
Child 9 created
Child 11 created
ps
name     pid     state   priority
init     1       SLEEPING        2
 sh      2       SLEEPING        2
 ps      12      RUNNING         2
 foo     10      RUNNING         10
 processes completed$ nice 11 8
$ ps
name     pid     state   priority
init     1       SLEEPING        2
 sh      2       SLEEPING        2
 ps      14      RUNNING         2
 foo     11      RUNNING         8
 processes completed$ ps
name     pid     state   priority
init     1       SLEEPING        2
 sh      2       SLEEPING        2
 ps      15      RUNNING         2
 foo     11      RUNNING         8
 processes completed$ 
```

Evalutation: We were able to complete each step of the Lab, without any errors and with correct outputs. Including the expected outputs from the Xv6 project as well.