Michael Smith

CSE 460

Homework 3

2/13/18

Total Points: 60

1. (10 points)
   The aging algorithm with a = 1/2 is being used to predict run times. The previous four runs, from oldest to most recent are 40, 20, 40, and 15 msec. What is the next run time?

   Using the four previous run times, the prediction is able to be produced:

   (((40+20)/2+40)/2+15)/2

   = ((30+40)/2+15)/2

   =(35+15)/2

   = 25

   While using the previous two run times, the prediction becomes

   (40_15)/2

   = 27.5

2. (10 points)
   Measurement of a certain system have shown that the average process runs for a time T before blocking on I/O. A process switch requires a time S, which is effectively wasted (overhead). For round robin scheduling with quantum Q, give a formula for the CPU efficiency for each of the following.

   a. Q = infinity
   b. Q > T
   c. S < Q < T
   d. Q = S
   e. Q nearly 0

   Number of times switch = (T/Q)

   Time Wasted switching = (T/Q)*S

   So efficiency becomes = T / (T + ST/Q)

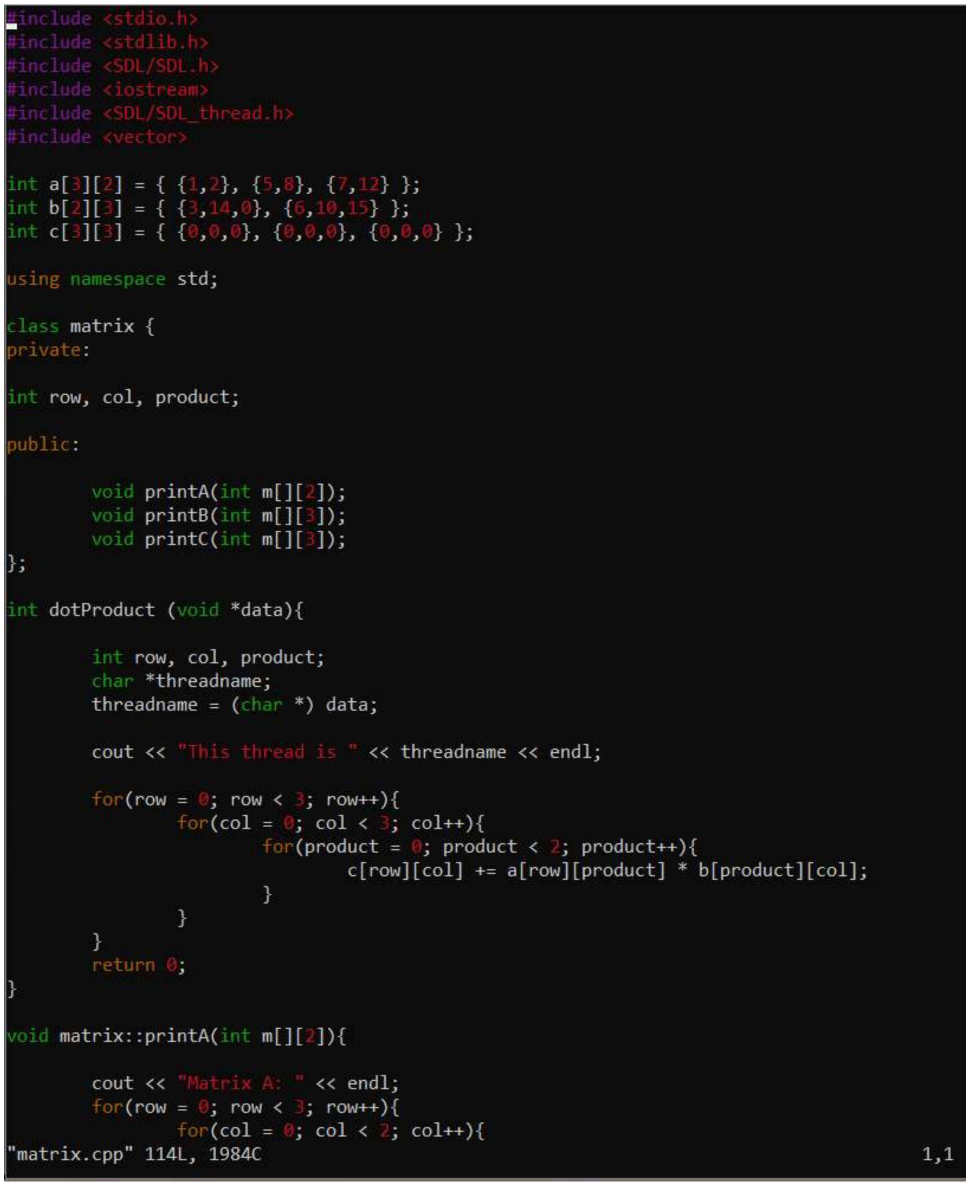
   a. S = 0 while the CPU efficiency is T/T = 100%
   b. S = 0 while the CPU efficiency is T/T = 100%

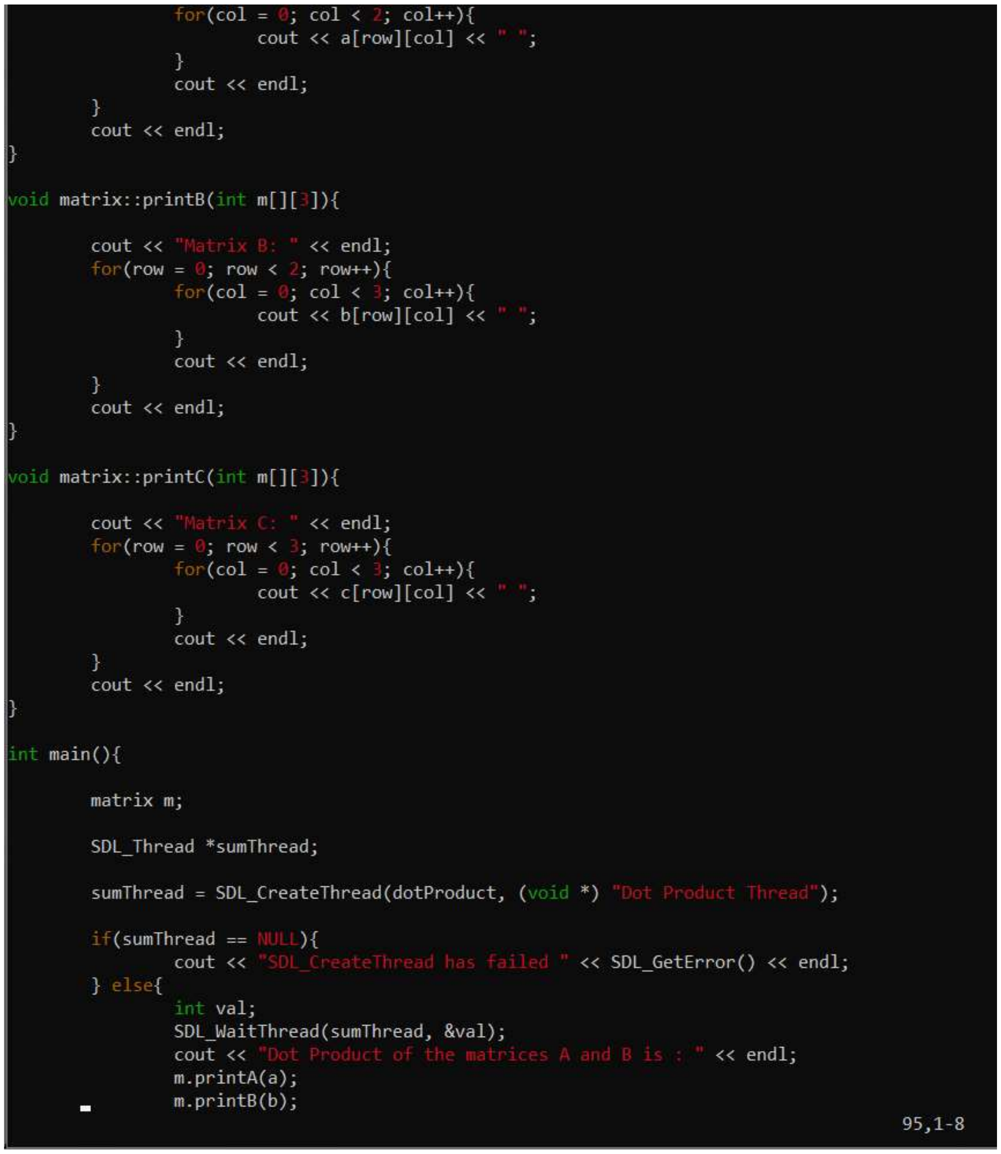
c. Using the efficiency equation, T / (T + ST/Q), the value would vary between 100% to 50% depending on how much less Q is than T.
d. S = Q so the efficiency becomes T / (T + ST/Q) = T /(2T) = 50%
e. With Q ~= 0 the efficiency becomes T / (T + ST/Q) = T / ~infinity = Undefined or 0%

3. (20 points)
Write a multithreaded program using SDL threads or POSIX threads. The program uses a number of threads to multiply two matrices. The multiplication of an M X L matrix A and an L X N matrix B gives an M X N matrix C, and is given by the formula,

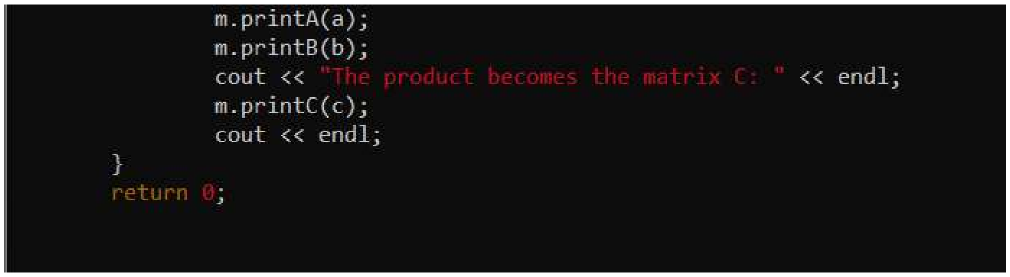
$$C_{ij} = \sum_{k=0}^{L-1} A_{ik} B_{kj} \quad 0 \leq i < M,\ 0 \leq j < N$$

Basically, each element C_{ij} is the dot product of the i-th row vector of A with the j-th column vector of B. The program uses one thread to calculate a dot product. Therefore, it totally needs M x N threads to calculate all the elements of matrix C.


```
#include <stdio.h>
#include <stdlib.h>
#include <SDL/SDL.h>
#include <iostream>
#include <SDL/SDL_thread.h>
#include <vector>

int a[3][2] = { {1,2}, {5,8}, {7,12} };
int b[2][3] = { {3,14,0}, {6,10,15} };
int c[3][3] = { {0,0,0}, {0,0,0}, {0,0,0} };

using namespace std;

class matrix {
private:

int row, col, product;

public:

        void printA(int m[][2]);
        void printB(int m[][3]);
        void printC(int m[][3]);
};

int dotProduct (void *data){

        int row, col, product;
        char *threadname;
        threadname = (char *) data;

        cout << "This thread is " << threadname << endl;

        for(row = 0; row < 3; row++){
                for(col = 0; col < 3; col++){
                        for(product = 0; product < 2; product++){
                                c[row][col] += a[row][product] * b[product][col];
                        }
                }
        }
        return 0;
}

void matrix::printA(int m[][2]){

        cout << "Matrix A: " << endl;
        for(row = 0; row < 3; row++){
                for(col = 0; col < 2; col++){
"matrix.cpp" 114L, 1984C                                                        1,1
```

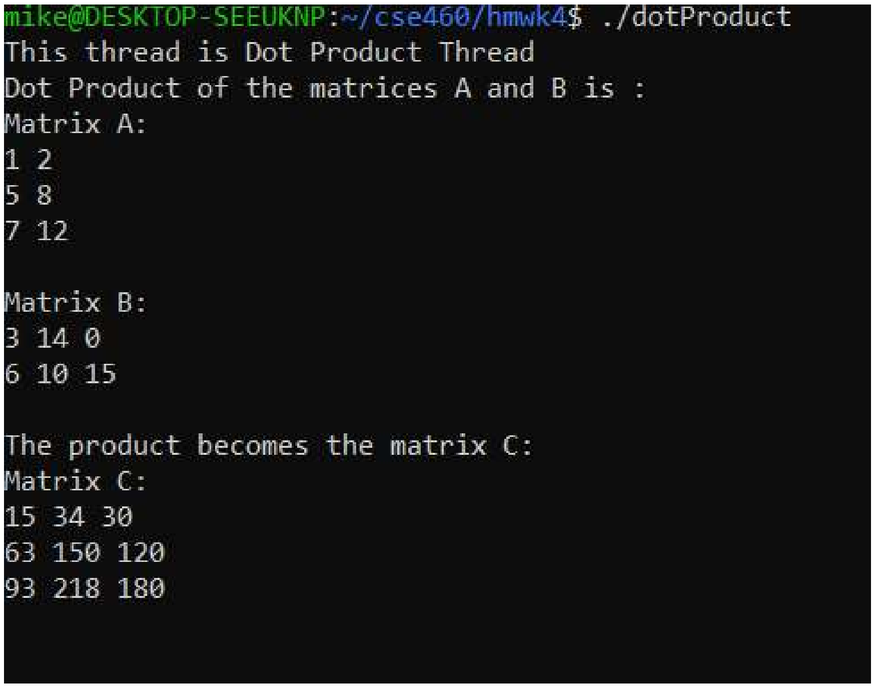


```
                for(col = 0; col < 2; col++){
                        cout << a[row][col] << " ";
                }
                cout << endl;
        }
        cout << endl;
}

void matrix::printB(int m[][3]){

        cout << "Matrix B: " << endl;
        for(row = 0; row < 2; row++){
                for(col = 0; col < 3; col++){
                        cout << b[row][col] << " ";
                }
                cout << endl;
        }
        cout << endl;
}

void matrix::printC(int m[][3]){

        cout << "Matrix C: " << endl;
        for(row = 0; row < 3; row++){
                for(col = 0; col < 3; col++){
                        cout << c[row][col] << " ";
                }
                cout << endl;
        }
        cout << endl;
}

int main(){

        matrix m;

        SDL_Thread *sumThread;

        sumThread = SDL_CreateThread(dotProduct, (void *) "Dot Product Thread");

        if(sumThread == NULL){
                cout << "SDL_CreateThread has failed " << SDL_GetError() << endl;
        } else{
                int val;
                SDL_WaitThread(sumThread, &val);
                cout << "Dot Product of the matrices A and B is : " << endl;
                m.printA(a);
                m.printB(b);
                                                                        95,1-8
```



```
                m.printA(a);
                m.printB(b);
                cout << "The product becomes the matrix C: " << endl;
                m.printC(c);
                cout << endl;
        }
        return 0;
```

Output



```
mike@DESKTOP-SEEUKNP:~/cse460/hmwk4$ ./dotProduct
This thread is Dot Product Thread
Dot Product of the matrices A and B is :
Matrix A:
1 2
5 8
7 12

Matrix B:
3 14 0
6 10 15

The product becomes the matrix C:
Matrix C:
15 34 30
63 150 120
93 218 180
```

4. (20 points)
   In the class the implementation of the readers-writers problem using SDL threads has been presented. However, the read and write tasks of the **reader** thread and the **writer** thread are not given. Implement these tasks as reading and writing of a file named *counter.txt*, which contains an integer counter.
   A **reader** thread

   reads the counter from the file, and

   prints out its thread name and the value of the counter.

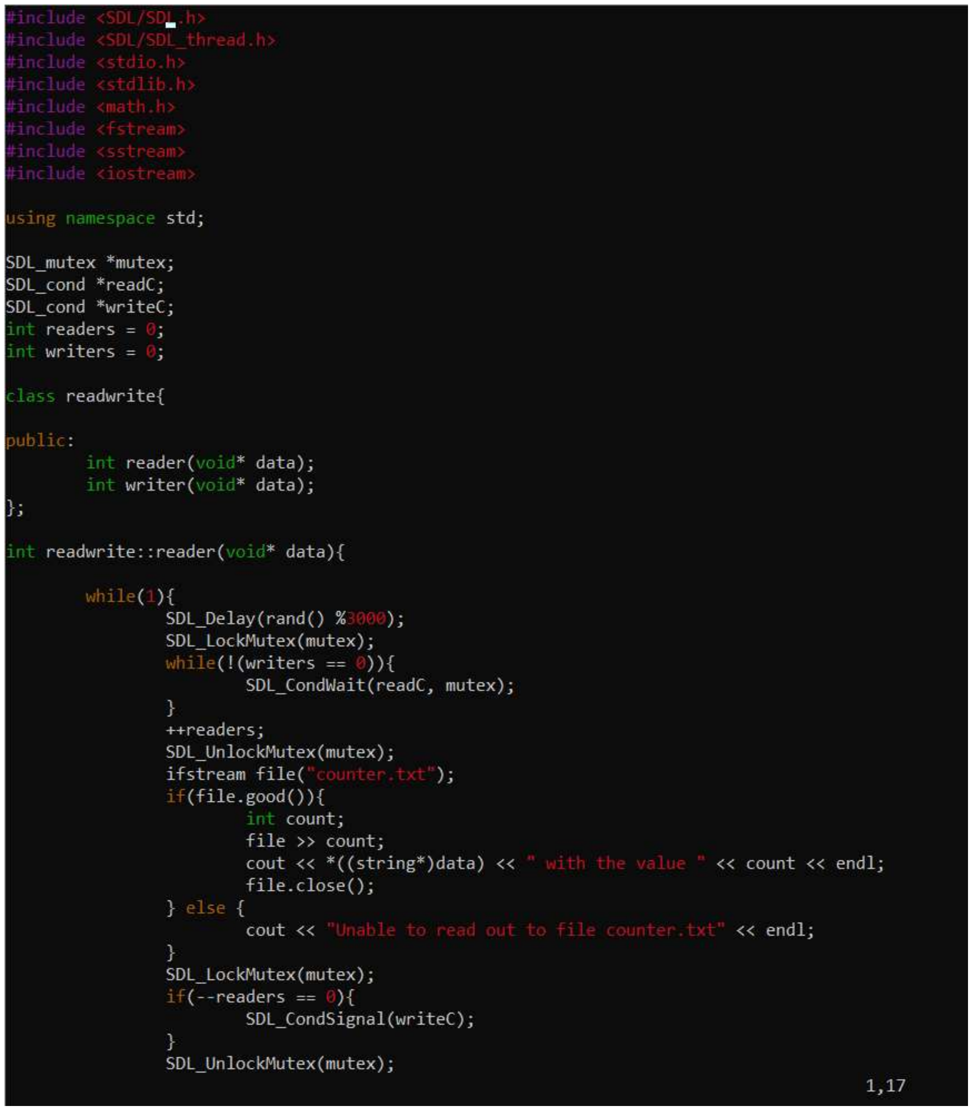
   A **writer** thread

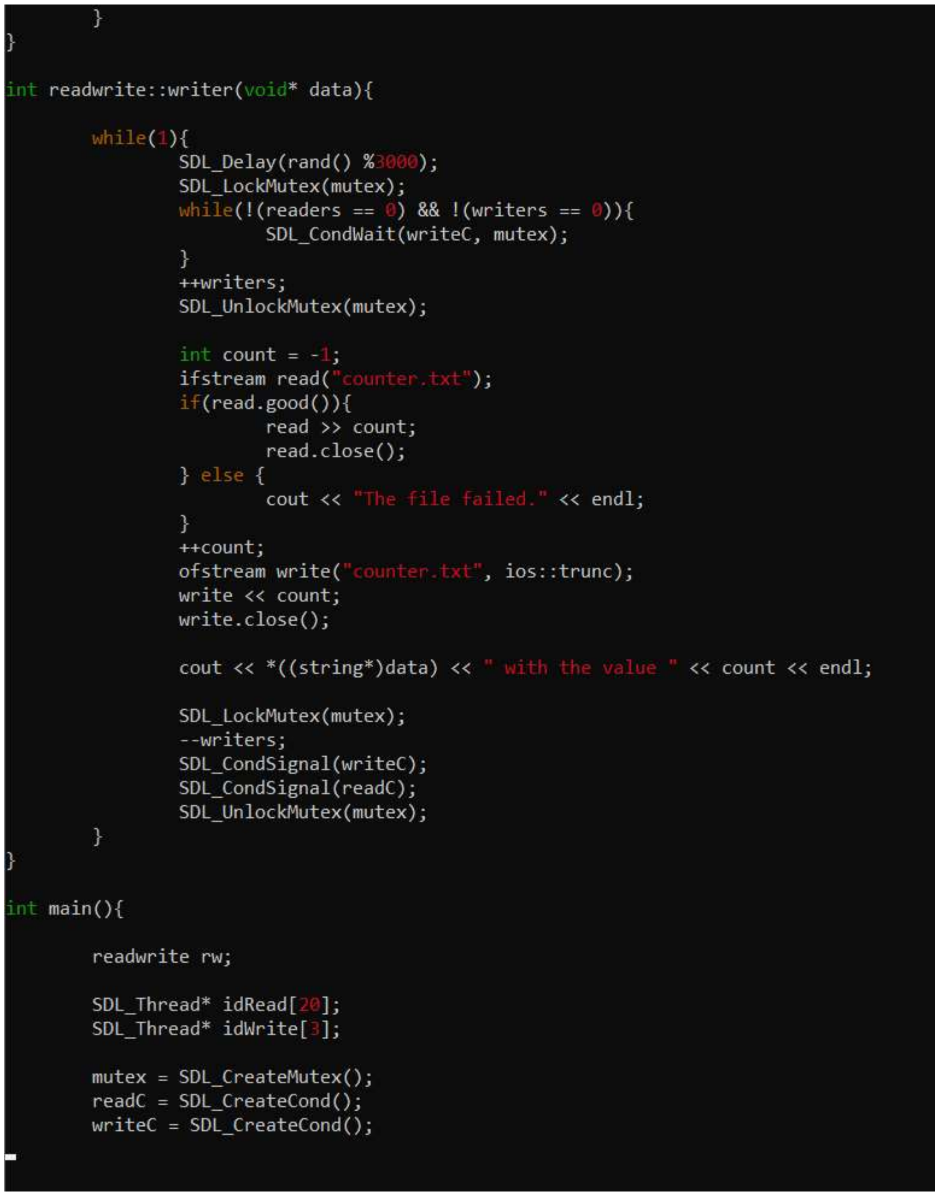
   increments the value of the counter in the file,

   prints out its thread name and the new value of the counter.

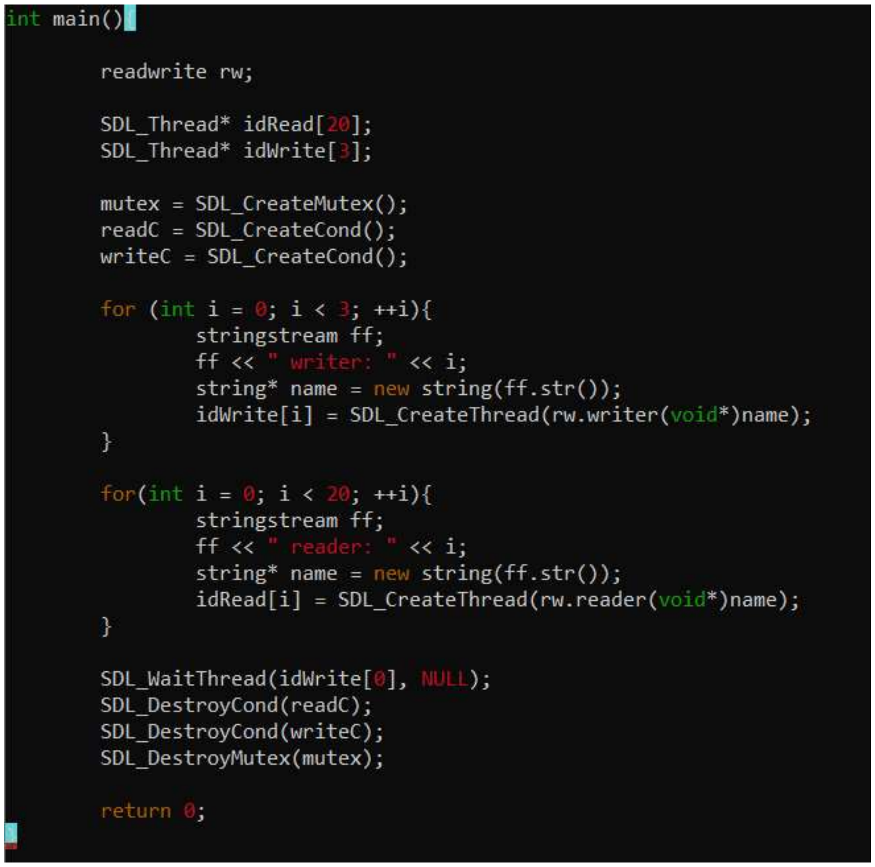
Each thread repeats its task indefinitely in a random amount of time between 0 and 3000 ms. Your **main** program should create 20 **reader** threads and 3 **writer** threads.

Code:


```
#include <SDL/SDL.h>
#include <SDL/SDL_thread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fstream>
#include <sstream>
#include <iostream>

using namespace std;

SDL_mutex *mutex;
SDL_cond *readC;
SDL_cond *writeC;
int readers = 0;
int writers = 0;

class readwrite{

public:
        int reader(void* data);
        int writer(void* data);
};

int readwrite::reader(void* data){

        while(1){
                SDL_Delay(rand() %3000);
                SDL_LockMutex(mutex);
                while(!(writers == 0)){
                        SDL_CondWait(readC, mutex);
                }
                ++readers;
                SDL_UnlockMutex(mutex);
                ifstream file("counter.txt");
                if(file.good()){
                        int count;
                        file >> count;
                        cout << *((string*)data) << " with the value " << count << endl;
                        file.close();
                } else {
                        cout << "Unable to read out to file counter.txt" << endl;
                }
                SDL_LockMutex(mutex);
                if(--readers == 0){
                        SDL_CondSignal(writeC);
                }
                SDL_UnlockMutex(mutex);
                                                                                1,17
```

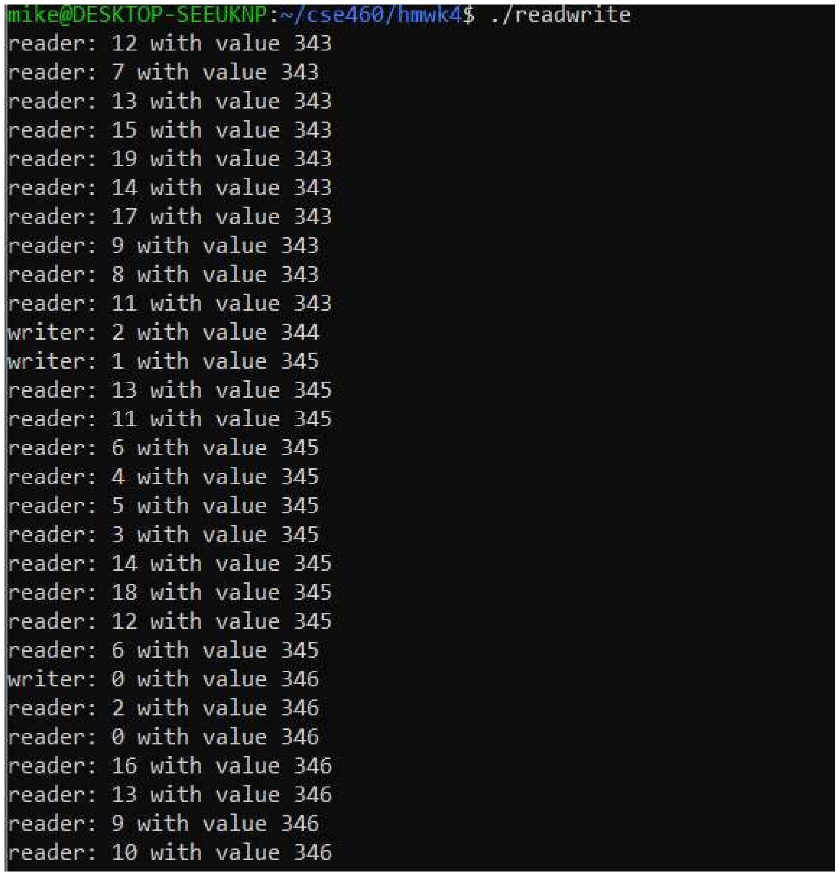


```
        }
}

int readwrite::writer(void* data){

        while(1){
                SDL_Delay(rand() %3000);
                SDL_LockMutex(mutex);
                while(!(readers == 0) && !(writers == 0)){
                        SDL_CondWait(writeC, mutex);
                }
                ++writers;
                SDL_UnlockMutex(mutex);

                int count = -1;
                ifstream read("counter.txt");
                if(read.good()){
                        read >> count;
                        read.close();
                } else {
                        cout << "The file failed." << endl;
                }
                ++count;
                ofstream write("counter.txt", ios::trunc);
                write << count;
                write.close();

                cout << *((string*)data) << " with the value " << count << endl;

                SDL_LockMutex(mutex);
                --writers;
                SDL_CondSignal(writeC);
                SDL_CondSignal(readC);
                SDL_UnlockMutex(mutex);
        }
}

int main(){

        readwrite rw;

        SDL_Thread* idRead[20];
        SDL_Thread* idWrite[3];

        mutex = SDL_CreateMutex();
        readC = SDL_CreateCond();
        writeC = SDL_CreateCond();
```



```
int main(){

        readwrite rw;

        SDL_Thread* idRead[20];
        SDL_Thread* idWrite[3];

        mutex = SDL_CreateMutex();
        readC = SDL_CreateCond();
        writeC = SDL_CreateCond();

        for (int i = 0; i < 3; ++i){
                stringstream ff;
                ff << " writer: " << i;
                string* name = new string(ff.str());
                idWrite[i] = SDL_CreateThread(rw.writer(void*)name);
        }

        for(int i = 0; i < 20; ++i){
                stringstream ff;
                ff << " reader: " << i;
                string* name = new string(ff.str());
                idRead[i] = SDL_CreateThread(rw.reader(void*)name);
        }

        SDL_WaitThread(idWrite[0], NULL);
        SDL_DestroyCond(readC);
        SDL_DestroyCond(writeC);
        SDL_DestroyMutex(mutex);

        return 0;
}
```

Output



```
mike@DESKTOP-SEEUKNP:~/cse460/hmwk4$ ./readwrite
reader: 12 with value 343
reader: 7 with value 343
reader: 13 with value 343
reader: 15 with value 343
reader: 19 with value 343
reader: 14 with value 343
reader: 17 with value 343
reader: 9 with value 343
reader: 8 with value 343
reader: 11 with value 343
writer: 2 with value 344
writer: 1 with value 345
reader: 13 with value 345
reader: 11 with value 345
reader: 6 with value 345
reader: 4 with value 345
reader: 5 with value 345
reader: 3 with value 345
reader: 14 with value 345
reader: 18 with value 345
reader: 12 with value 345
reader: 6 with value 345
writer: 0 with value 346
reader: 2 with value 346
reader: 0 with value 346
reader: 16 with value 346
reader: 13 with value 346
reader: 9 with value 346
reader: 10 with value 346
```

Report: I was able to complete each section of the homework assignment, while producing correct results. I believe I have earned a 60/60 grade.