3.

```
hau@hau-Lenovo-Y50-70:~/Desktop/cse460/hw4$ g++ -o rwp rwp.cpp -lSDL -lpthread
hau@hau-Lenovo-Y50-70:~/Desktop/cse460/hw4$ ./rwp
reader: 18 with value: 150
reader: 10 with value: 150
_____writer: 1 with value: 151
reader: 9 with value: 151
reader: 17 with value: 151
reader: 7 with value: 151
reader: 16 with value: 151
reader: 7 with value: 151
_____writer: 2 with value: 152
reader: 18 with value: 152
reader: 19 with value: 152
reader: 2 with value: 152
reader: 1 with value: 152
_____writer: 1 with value: 153
reader: 11 with value: 153
reader: 10 with value: 153
_____writer: 0 with value: 154
reader: 7 with value: 154
reader: 17 with value: 154
reader: 16 with value: 154
reader: 14 with value: 154
reader: 4 with value: 154
reader: 3 with value: 154
reader: 1 with value: 154
reader: 13 with value: 154
_____writer: 0 with value: 155
reader: 9 with value: 155
```

```
/*
  readers_writers.cpp
  Compile:  g++ -o  readers_writers readers_writers.cpp -lSDL -lpthread
  Execute:  ./readers_writers
*/

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

using namespace std;

SDL_bool condition = SDL_FALSE;
SDL_mutex *mutex;
SDL_cond *readerQueue;   //condition variable
SDL_cond *writerQueue;   //condition variable
```

```
int readers = 0;
int writers = 0;
int active_writers = 0;
int reader ( void *data )
{
  while(1){

        SDL_Delay ( rand() % 3000);
        SDL_LockMutex ( mutex );
        while ( !(writers == 0) )
        SDL_CondWait ( readerQueue, mutex );
        readers++;
        SDL_UnlockMutex ( mutex );
        ifstream file("counter.txt");
        if(file.good()){

        int count;
        file >> count;
        cout << *((string*) data) << " with value: " <<count << endl;
        }
        else
        {
        cout << "Uable to read counter.txt" << endl;

        }
        SDL_LockMutex ( mutex );
        if ( --readers == 0 )
        SDL_CondSignal ( writerQueue );
        SDL_UnlockMutex ( mutex );
  }
}

int  writer ( void *data )
{
  while(1){
        SDL_Delay ( rand() % 3000);
        SDL_LockMutex(mutex);
        writers++;
        while ( !( (readers == 0) && (active_writers == 0) ) )
               SDL_CondWait ( writerQueue, mutex );
        active_writers++;

        SDL_UnlockMutex ( mutex );
        int count =-1;
        ifstream read("counter.txt");
        if(read.good()){
               read >> count;
               read.close();
        }
        else{
               cout <<"Write file failed" << endl;

        }
        ++count;
        ofstream write("counter.txt", ios::trunc);
        write << count;
        cout << *((string*) data) << " with value: " <<count << endl;

        SDL_LockMutex ( mutex );
```

```
        active_writers--;
        if(--writers == 0)
                SDL_CondSignal ( readerQueue );
        else
                SDL_CondSignal ( writerQueue );



        //SDL_CondBroadcast ( readerQueue );
        SDL_UnlockMutex ( mutex );
  }

}

int main ()
{
  SDL_Thread *idr[20];
  SDL_Thread *idw[3];                //thread identifiers
  mutex = SDL_CreateMutex();
  readerQueue = SDL_CreateCond();
  writerQueue = SDL_CreateCond();
  for (int i = 0; i < 20; i++ ){
       stringstream ss;
       ss<<"reader: " <<i;
       string *name = new string (ss.str());
       idr[i] = SDL_CreateThread (reader, name);

  }


  for (int i= 0; i < 3; i++ ){
       stringstream ss;
       ss<<"_____writer: " <<i;
       string *name = new string (ss.str());
       idw[i] = SDL_CreateThread (writer, name );


  }


  SDL_WaitThread ( idw[0], NULL );
  SDL_DestroyCond ( readerQueue );
  SDL_DestroyCond ( writerQueue );
  SDL_DestroyMutex ( mutex );
  return 0;

}
```