Simulate the zombie cleaning activity as a multi-threaded program. You should have multiple threads, one representing you (slayer) and one for each friend controlling a door (doorMan).

●void zombieEntered(): Keeps track of number of zombies entered.

●void zombieKilled(): Keeps track of number of zombies killed.

●int tooManyZombiesInTheRoom(): Returns true if number of zombies in the room are greater than or equal to 100.

●int killed100Zombies(): Returns true if more than 100 zombies have been killed.

●int zombiesExist(); Returns true if there is at least one zombies in the room.

●int getKilledCount(); Returns the number of zombies killed.

●int getInTheRoomCount(); Returns the number of zombies in the room.

Since multiple threads access these functions, you want the counts to be consistent. You do not want any thread to get an inconsistent view of the data being accessed.

●void \*doorMan(void \*);

Each DoorMan thread lets in a zombie with a 50% chance every 2ms, keeping track of the number of zombies admitted (by calling the corresponding functions from the ZombieCounter). The DoorMan thread terminates if there are too many zombies (more than 100 zombies) in the room at any time or if the Slayer has killed more than 100 zombies.

●void \*slayer(void \*);

The Slayer kills a zombie every 2ms (but he/she has to check first to see if there is a zombie) keeping track of the number of zombies killed (by calling corresponding function). The Slayer thread terminates if there are too many zombies (more than 100 zombies) in the room at any time or if he/she has killed more than 100 zombies.

●int main();

The main function creates n noorMan threads (n is taken in as a command line argument) and one slayer thread. When all the threads complete their execution, the main thread checks and prints if you have killed 100 zombies or have been killed by the zombies.