**FirstName: Rashed**

**Last Name: Alyammahi**

1. **Which magic doors cannot be opened with valid magic codes? Explain with clear and precise justification.**

There are two magic codes: 13444 and 65234 cannot open magic doors. With 13444 case, which misses the unclock function. According to mutex knowledge, when the lock is set no other thread can access the shared resource. That is why need to release it. With 6534 case I believe there is something is preventing the magic door open and that things is buffer overflow.A buffer overflow (or buffer overrun) occurs when the volume of data exceeds the storage capacity of the memory buffer. ... They typically result from malformed inputs or failure to allocate enough space for the buffer.

This cause our code to be vulnerable and cause the reason to not open the magical door.

1. **Can you open any magic door without using its valid magic code? If yes, explain how? Give a clear and precise explanation including an invalid magic code that opens a door.**

In my opinion, there cannot open the magical door without using it’s valid magic code After analyzing the code, I found out that there is some if checks that helps strengthen the security that no false value can open the magical doors. First thing is that I noticed that int magic number is declared and defined at the top with the initial value of 0. There is only one way to open all the doors when the value of your variable will become null but on int the value can’t be null.

Therefore, we can’t open any magic door without using its valid magic code

1. **Is there any possible execution behavior in the function verifyMagicCode on which the lock() is not followed by unlock ()? if yes, write the execution sequence using line numbers**

Yes. When the lock starts and on no value is matched with magic number then the switch default statement will run which is not having a unlock function.

The execution sequence will be: 31,35,37,82,83,84,85

1. **Is there any possible execution behavior in the program on which lock () is followed by lock ()? if yes, explain the scenario.**

Yes, the Lock is followed by a Lock in int main () function where we want to continue the execution of our code and don’t want to release it until we got our output or some error. Mutex lock will only be released by the thread who locked it.

1. **What can go wrong if you have lock followed by lock?**

Only one of all the threads would make it to the execution but if this thread tries to execute the same region of code that is already locked then it will again go to sleep and we have to terminate the program.

1. **What can go wrong if the lock is not followed by unlock? Explain it**

All other threads calling pthread\_mutex\_lock will simply block and wait for the first thread to release the mutex by calling pthread\_mutex\_unlock

1. **Write a short paragraph about the aspects of program you examined to answer the above questions. (*For this part, you will be given credit based on the quality of your effort and how well you have described* it.)**

I tried my best to analyze the code. After analyzing it, I discovered that there has been a missing unclock function which is causing the magic door is not open and there is also have a case where the input is really high in compare to the data type of integer. That is issue relating buffer overflow.

On the second question, I tried everything I could to open the door but I couldn’t that make me realize that maybe there is no way we can breach this as there are many if checks for this condition. For the rest of the question I just run the code again and again and observe how it will work and what will happen we don’t release the thread or when we lock by lock and put my answers according to the results that I received and analysis that software verification can be really tricky if a developer’s forgot some unlock() or follow lock by lock