Rush 01

—----

What to discuss with them:

1. Talk about some **combinatorics**)) How many possible ways are there to create a n-digit number with repeating/non repeating digits, brainstorm on how many possible ways there are to fill in a 2x2, 3x3, 4x4 squares with the given conditions. (Try to tell them something that's not right with a really serious face to see their reaction, i. e ….. There are 4!*3!*2!*1! Possible ways to fill in a 4x4 square (but there's actually more))
2. Would they be able to write a program that would count the number of unique squares like that? (They probably should, since if they've solved the problem - they have done more than that)
3. You can point out some random lines on the code (function declarations, prototypes, different data types used) and ask them to explain that to you.
4. Have they used **malloc**? What does it do? What does it return? Ask questions about malloc and free pair, you can go as deep as discussing how the OS releases all the used resources (memory, open files) after a process ends.
5. What are **memory leaks**? What are they "bad for" and how to avoid having them? Show them some ways of checking memory leaks.
6. What's **recursion**? Why have they used it, if they have? Can the **main** function be called recursively? Brainstorm on what could serve as a stop condition in that case.

Cases to check:

1. How well is malloc returning **NULL** case handled. What if they were to use the pointer returned in some other function, how well is that handled?
2. How have they handled user **input validation**? Try to break it, and brainstorm on better solutions.
3. If they have gone as far as solving the problem for **9x9**, discuss the possible **optimisations** that could've been done to the algorithm.