**Study Questions**

1. Why is using a fixed-size array problematic in this case?
2. What could happen if the user enters more subjects than the array size?
3. Why is input validation necessary, and how would you handle invalid input?
4. Why should file paths not be hardcoded in programs?
5. What are the benefits of using vector over arrays in C++?
6. How can you make the databaseConnection function safer and more scalable?
7. What changes would you make to handle storing multiple DMC records for the same college?
8. How would you modify the program if each subject could have different weightage instead of equal marks?
9. What would happen if the user entered non-numeric data for marks? How would you handle it?
10. Can you think of a reason why suddenly a large number like 597661888 appeared in the output file?

**Suggestion for improvement**

1. **Dynamic Arrays / Vectors Instead of Fixed Arrays:**
   * Replace marksArray[3] and subjectNameArray[3] with vector<int> and vector<string> to avoid out-of-bounds issues and support dynamic subject counts.
2. **Input Validation:**
   * Add input checks for marks (e.g., between 0 and 100) and non-empty subject names or college names.
   * Handle invalid integers or unexpected input formats gracefully.
3. **File Path Flexibility:**
   * Replace hardcoded file paths ("d:\\2nd\\DMC.txt") with user input or relative paths.
   * Use cross-platform file handling approaches.
4. **Functions and Modularity:**
   * Break code into functions: getInput(), calculateGrade(), writeToFile(), and displayDMC() for better readability and reusability.
5. **Constants and Enums:**
   * Use const or #define for grade boundaries and messages, and consider enum for grade categories for clarity.
6. **Class-Based Structure (Optional Advanced Suggestion):**
   * Wrap subject details and DMC generation logic in a DMC class for OOP demonstration.
7. **Use of std::setw() and Formatting:**
   * Improve output readability by using iomanip for consistent spacing and tabular data.
8. **Add Timestamp:**
   * Add a timestamp when writing to DMC.txt for record-keeping.
9. **Improve databaseConnection() Function:**
   * Return a default string if the choice is invalid, or use a map for clarity.
10. **Handle Large Totals & Data Type Selection:**
    * Validate that data types are sufficient (the incorrect large number seen in DMC file could be input or type overflow).