C++ Exception Handling Practice Problems

Problem 1: Bank Account Manager

Create a `BankAccount` class with methods for deposit, withdraw, and balance checking. Handle these exceptions:

- **Negative amount**: Throw `invalid_argument` when deposit/withdraw amount is negative
- **Insufficient funds**: Throw `runtime_error` when withdrawal exceeds balance
- **Account not found**: Throw `out_of_range` when accessing invalid account ID

Write a main function that demonstrates all three exception scenarios with proper try-catch blocks.

Problem 2: Array Statistics Calculator

Write a function `calculateStats(vector<int>& arr, int index)` that:

- Calculates mean, median, and mode of array elements
- Accesses element at given index for additional processing
- Allocates dynamic memory for temporary calculations

Handle these exceptions:

- **Empty array**: Throw `logic_error` if array is empty
- **Index out of bounds**: Handle `out_of_range` when accessing invalid index
- **Memory allocation failure**: Handle `bad_alloc` during dynamic allocation
- **Division by zero**: Handle when calculating mean with zero elements

Problem 3: File Processing System

Create a program that reads numbers from a file and performs mathematical operations. Handle:

- **File not found**: Throw `runtime_error` if file cannot be opened
- **Invalid format**: Throw `invalid_argument` for non-numeric data
- **Arithmetic errors**: Handle division by zero in calculations
- **Memory issues**: Handle `bad_alloc` when storing large datasets

The program should continue processing valid data even after encountering errors.

Problem 4: Student Grade Manager

Build a grade management system with these requirements:

- Store student grades in a map (student_id -> grades vector)
- Calculate GPA, find highest/lowest grades
- Support grade curves and weighted averages

Handle these exceptions:

- **Invalid grade**: Throw `out_of_range` for grades outside 0-100

- **Student not found**: Throw `runtime_error` for non-existent student IDs
- **Empty grade list**: Throw `logic_error` when calculating stats with no grades
- **Invalid weights**: Throw `invalid_argument` for negative or zero weights

Problem 5: Matrix Operations Library

Implement a `Matrix` class that supports addition, multiplication, and determinant calculation. Handle:

- **Dimension mismatch**: Throw `invalid_argument` for incompatible matrix operations
- **Non-square matrix**: Throw `logic_error` when calculating determinant of non-square matrix
- **Singular matrix**: Throw `runtime_error` for matrices with zero determinant during inversion
- **Memory allocation**: Handle `bad_alloc` for large matrix operations
- **Index bounds**: Handle `out_of_range` for invalid matrix element access