Part 1: Assumptions

1.1 General Assumptions

To develop this Music Ensemble Management System (MEMS), I made several assumptions based on the assignment requirement PDF.

1.1.1 User Input Assumptions

- User will input command by typing single character like "C", "A", "M" etc.
- When user need to input ensemble ID or musician ID, they will enter a valid integer number
- If user enter wrong format input, the system will just ignore it and ask again
- User can type "X" anytime to exit the program

1.1.2 Ensemble Management Assumptions

- Each ensemble must have unique ID that is auto-generated by system
- Ensemble ID start from 1 and increase by 1 for each new ensemble created
- There can be two type of ensemble only: Orchestra and Jazz Band
- User need to "set current ensemble" before they can add musician or do other operations on it
- If no current ensemble is set, some commands like add musician will show error message

1.1.3 Musician Management Assumptions

- Each musician have unique ID within the ensemble
- Musician ID also start from 1 and auto-increment
- When add musician to ensemble, system will auto-assign the ID
- One musician can only belong to one ensemble (cannot share between ensembles)
- When delete a musician, their ID will not be reused for new musicians

1.1.4 Role/Instrument Assumptions

- For Orchestra ensemble, there are 2 valid roles (instruments):
    1. Violinist
    2. Cellist
- For Jazz Band ensemble, there are 3 valid roles:
    1. Pianist
    2. Saxophonist
    3. Drummer
- User must input valid role number when add or modify musician
- If input invalid role number, system will show error and not proceed

1.1.5 Undo/Redo Assumptions

- System keep track of all commands that can be undone (not include read-only commands like Show, Display, List)
- User can undo multiple times until there is no more history

- After undo, user can redo to restore the previous operation
- If user execute new command after undo, the redo stack will be cleared
- Read-only commands (Show ensemble, Display all ensembles, List undo/redo) should not appear in undo list

1.2 Design Pattern Assumptions

1.2.1 Command Pattern

- All 12 functions in the system should be implemented as separate Command classes
- Each command have execute() and undo() method
- Some commands don't need undo (like display commands) so their undo() method can be empty

1.2.2 Factory Pattern

- CommandFactory is responsible for creating all command objects
- Main class should not directly create command objects using "new" keyword
- Factory will handle the complexity of creating different types of commands

1.2.3 Memento Pattern

- Need to save state of musician or ensemble before making changes
- This allows proper undo functionality
- Memento objects store the previous state and can restore it when needed

1.3 Data Storage Assumptions

- This is a console-based application, no database or file storage needed
- All data exist in memory only during program execution
- When program exits, all data will be lost
- No need to implement save/load functionality

1.4 Error Handling Assumptions

- System should handle invalid input gracefully without crashing
- Error messages should be clear and helpful for user
- System continue running even after error occurs
- No need complex exception handling, simple if-else checks are enough

1.5 Output Format Assumptions

- All output should match exactly with the sample output in PDF
- Ensemble information displayed in format: "Type Name (ID)"
- Musician information displayed in format: "Name (ID), Role"
- List should be displayed in order they were added (maintain insertion order)

1.6 Constraints from Assignment

- Cannot modify the design of existing classes: Ensemble, OrchestraEnsemble, JazzBandEnsemble, Musician

- Must implement exactly 12 functions as specified
- Must use the three design patterns mentioned
- Code should be well-commented and easy to understand