ratio	letter		
100.00%		**1**	**Part 1 : Player Strategy Pattern**
10.00%		**1.1**	**Knowledge/Correctness of Game Rules**
5.00%	A	1.1.1	Students were fully aware and did not have any misconception about Warzone game rules during the presentation
5.00%	A	1.1.2	Code is implementing game mechanics that is according the Warzone game and the assignment description
60.00%		**1.2**	**Compliance of solution with Stated Problem**
6.67%	A	1.2.1	The Human player uses user interaction to implement all decisions available in the game (all kinds of orders, play cards).
6.67%	A	1.2.2	The Benevolent player deploys on its weaker territories.
6.67%	A	1.2.3	The Aggressive player strategy uses all its available armies to attack when possible
6.67%	A	1.2.4	The Aggressive player strategy always deploys on its strongest territory
6.67%	A	1.2.5	The Neutral player strategy cannot issue orders
6.67%	A	1.2.6	The solution to implement the player strategies is using the structure and behavior of the strategy pattern.
6.67%	A	1.2.7	All classes implement a correct copy constructor, assignment operator, and stream insertion operator.
6.67%	A	1.2.8	Absence of memory leaks.
6.67%	A	1.2.9	All data members of user-defined class type are pointers
10.00%		**1.3**	**Modularity of Solution**
1.67%	A	1.3.1	The player strategies are implemented in a file duo named PlayerStrategies.cpp/PlayerStrategies.h
1.67%	A	1.3.2	Each strategy is implemented as a subclass of a Strategy class, forllowing the Strategy design pattern
1.67%	A	1.3.3	The issueOrder() method of the player delegates to the issueOrder() method of the player's Strategy data member
1.67%	A	1.3.4	The player contains a data member of type PlayerStrategy.
1.67%	A	1.3.5	The Player constains a member function setStrategy() that enables it to change its strategy dynamically at runtime.
1.67%	A	1.3.6	The Player class is still in the file duo named Player.cpp/Player.h
10.00%		**1.4**	**Mastery of Language/Tools/Libraries**
5.00%	A	1.4.1	The program never crashed during the demonstration or code review
5.00%	A	1.4.2	Students were very clear in technical discussions during the demonstration
10.00%		**1.5**	**Code readability: name conventions, clarity of code, use of comments**
5.00%	A	1.5.1	All user-defined classes, methods, free functions, or operators are documented
5.00%	A	1.5.2	Code is clear and there is zero presence of commented-out code

100.00%		**2**	**Part 2 : File Reader Adapter**
10.00%		**2.1**	**Knowledge/Correctness of Game Rules**
5.00%	A	2.1.1	Students were fully aware and did not have any misconception about Warzone game rules during the presentation
5.00%	A	2.1.2	Code is allowing to read both Domination and Conquest files that enables the game to run using the Warzone game rules
60.00%		**2.2**	**Compliance of solution with Stated Problem**
12.00%	A	2.2.1	The GameEngine can now read either Domination or Conquest map files and play a game using either of the map files.
12.00%	A	2.2.2	The solution to implement the reading of Conquest files is using the structure and behavior of the Adapter pattern.
12.00%	A	2.2.3	All classes implement a correct copy constructor, assignment operator, and stream insertion operator.
12.00%	A	2.2.4	Absence of memory leaks.
12.00%	A	2.2.5	All data members of user-defined class type are pointers
10.00%		**2.3**	**Modularity of Solution**
2.50%	A	2.3.1	The adapter and adaptee classes are implemented in the pre-existing MapLoader.cpp/MapLoader.h file duo
2.50%	A	2.3.2	There is an adaptee class named ConquesFileReader that implements the reading of Conquest map files.
2.50%	A	2.3.3	There is an adapter class names ConquestFileReaderAdapter that contains a member of type ConquestFileReader
2.50%	A	2.3.4	The ConquestFileReaderAdapter is a subclass of the original Domination file reader class.
10.00%		**2.4**	**Mastery of Language/Tools/Libraries**
5.00%	A	2.4.1	The program never crashed during the demonstration or code review
5.00%	A	2.4.2	Students were very clear in technical discussions during the demonstration
10.00%		**2.5**	**Code readability: name conventions, clarity of code, use of comments**
5.00%	A	2.5.1	All user-defined classes, methods, free functions, or operators are documented
5.00%	A	2.5.2	Code is clear and there is zero presence of commented-out code