Programming Project

Instructions:

- **The project requires completing tasks by December 03, 2024, by 1:00 pm.**
- **Several programs accomplish tasks by transferring tokens between or within collections such as the following games:**

  - **Ball Sort**
  - **FreeCell**
  - **Solitaire**
  - **Spider Solitaire**
  - **Pyramid**
  - **TriPeaks**

  Your objective is to create one of the games listed above or an approved game with an undo capability.
- **You must define two classes that must be named *Deck* and *Game*. They must be written in separate header files that must be named 'Deck.h' and 'Game.h' respectively.**
- **Your header files can only include the libraries *iostream*, *string*, *sstream*, *cctype*, *iomanip*, *cstdlib*, *ctime*, *stdexcept* and *cmath*, and the header files 'Util.h' and 'Tools.h' as well.**
- **All fields of all classes must be private, but there is no limitation on how many fields can be included in the classes.**
- **The *Deck* class must use a *Token* array or list field as its dataset.**
- **The *Game* class must have a *Deck* array field.**
- **The methods listed in the tasks below and additional constructors must be public; otherwise, all additional methods must be private, but there is no limitation on how many additional methods can be included in the classes.**
- **Methods for validating or determining an aspect of the game must be constant methods.**
- **No additional files can be created for the project.**
- **You may work in groups of at most two members. Each member must define a single class if the group has two members.**
- **All classes must be defined to receive credit regardless of the group size.**
- **A typed document must be submitted by October 24 that provides the group members' names, class tasks, and the name and instructions of the selected game.**
- **Points for a task will only be awarded if completed accurately.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and failing to follow any rules will result in an automatic zero (0) for the project.**

Deck Grading

Task	Points	Earned
01	0.5	
02	0.5	
03	2.0	
04	2.0	
05	1.5	
06	0.5	
07	0.5	
08	0.5	
09	0.5	
10	0.5	
11	1.0	
Total	10.0	

Game Grading

Task	Points	Earned
01	0.5	
02	0.5	
03	1.0	
04	3.0	
05	1.0	
06	1.0	
07	0.5	
08	0.5	
09	2.0	
Total	10.0	

- Tasks highlighted in **red** must be submitted by October 24. Their grades are final.
- Tasks highlighted in **orange** must be submitted by November 14. Their grades are final.

Tasks:

- For the class *Deck*:
  1. Publicly inherit the class *Collection*.
  2. Define its special member functions so the deck is initially empty.
  3. Override `Insert()` method from *Collection* so that it adds the *Token* parameter into the deck based on the flag (string) parameter.
  4. Override `Remove()` method from *Collection* so that it removes a token from the deck based on the flag (string) parameter.
  5. Override `View()` methods from *Collection* so that they return a token from the deck based on the flag (string) parameter if the deck is not empty and the flag is valid; otherwise, it throws an error.
  6. Override `Extend()` method from *Collection* so that it sets if the display of the deck is extended or compressed.
  7. Override `Size()` method from *Collection* so that it returns the size of the deck.
  8. Override `Empty()` method from *Collection* so that it returns true if the deck is empty; otherwise, it returns false.
  9. Override `Clear()` method from *Collection* so that it empies the deck.
  10. Override `Info()` method from *Collection* so that it returns a string of the list of flags for the `Insert()`, `Remove()`, and `View()` methods.
  11. Override `ToString()` method from *Collection* so that it returns a string in the format

$$\begin{cases} [O] & \text{if the deck is empty} \\ [t]\ldots[b] & \text{if the deck view is extended} \\ [t] & \text{if the deck view is compressed} \end{cases}$$

  where O is a string used to represent an empty deck, t is the top token of the deck, and `[`t`]...[`b`]` is the list of the tokens of the deck from the top to the bottom token with each token on a separate line.

- For the class *Game*:
  1. Publicly inherit the class *Program*.
  2. Define its special member functions but delete its copy constructor and assignment operator.
  3. Override `Initialize()` method from *Program* so that it configures the game into its initial state.
  4. Override `Move()` method from *Program* so that it performs a task based on the flag (string) parameter.
  5. Override `Undo()` method from *Program* so that it undoes the previous moves performed.
  6. Override `Completed()` method from *Program* so that it returns true only if the game has ended.
  7. Override `State()` method from *Program* so that it returns a string that indicates the status of the game.
  8. Override `Info()` method from *Program* so that it returns a string of the list of flags for the `Move()` method.
  9. Override `ToString()` method from *Program* so that it returns a string of the game board.

Hints & Suggestions:

- A Boolean field would be useful for tracking the display mode of the *Deck* object.
- It should be obvious that most fields of the *Game* class are all *Deck* objects; the question is how many are needed.
- The `ToString()` method of *Game* may require more than invoking the `ToString()` method of each *Deck* objects. You can define the `View()` methods from *Deck* to access individual elements like an array.