Data Structures
CS 246 - 001
Department of Physics and Computer Science
Medgar Evers College
Exam 1

Instructions:

- **The exam requires completing a set of tasks within 80 minutes.**
- **Each task requires creating a header file that defines classes, functions, and objects based on a case study.**
- **The header files must include a header guard, define classes and objects within the namespace *dse*.**
- **Only libraries '`iostream`', '`string`', '`iomanip`', '`sstream`', '`stdexcept`', '`Object.h`', '`Array.h`', and '`Comparable.h`' can be included in any header file.**
- **Documentation for '`Object.h`', '`Array.h`', and '`Comparable.h`' are provided.**
- **All classes created must define essential methods publicly and fields privately.**
- **All classes must define their special member functions, which must be public.**
- **All derived classes must have a virtual destructor.**
- **Notes are not allowed.**
- **Cheating of any kind is prohibited and will not be tolerated.**
- **Violating and failing to follow rules will result in an automatic zero (0) for the exam.**

TO ACKNOWLEDGE THAT YOU HAVE READ AND UNDERSTOOD THE INSTRUCTIONS ABOVE, PRINT YOUR NAME AND THE DATE ON BOTH THIS SHEET AND THE BLUE BOOK

Name: ______________________ **Date:** ____________

Grading

Problem	**Maximum Points**	**Points Earned**
01	6.0	
02	8.0	
03	6.0	
Total	20.0	

1. Case Study Name: Exam Form Widget
   An online exam form is built by assembling widgets that represent different types of questions. One commonly used widget is the Short-Answer Question. Its characteristics are as follows:
   - It contains three fields: Question (text), Answer (textbox), and Solution (text).
   - The question and solution fields are initialized at the time of object creation and remain immutable thereafter. If they are not explicitly initialized, their values default to empty strings.
   - The widget operates in two stages, each producing a different view: Stage One View displays the question and the answer only; meanwhile, Stage Two View displays the question, answer, and solution.
   - The answer field may be modified only during Stage One.

   Using the specifications above, implement a class named *ShortAnswer* that publicly inherits *Object* and provides:
   - A constructor to initialize the question and solution fields.
   - A method to toggle between Stage One and Stage Two.
   - A getter method for each of the three fields: question, answer, and solution.
   - No setter methods for the question and solution fields, but one for the answer.
   - Overrides the `toString()` method to generate the appropriate view based on the current stage such that each field is on its own line, and the order question, answer, and solution with the labels `"Q: "`, `"A: "`, and `"S: "`, respectively.

   Afterward, define a *ShortAnswer Array* function that takes no parameters and returns an array of *ShortAnswer* objects corresponding to the following questions and their solutions (provided by you):
   - What is a pointer?
   - What is a dynamic memory?
   - What is the pointer member access operator?
   - What is a null-terminated doubly linked list?
   - What is a circular doubly linked list?

2. Case Study Name: Playing Cards
   Card games are played with decks composed of individual cards. The cards share the following characteristics:
   - Cards are comparable to one another.
   - Each card has two attributes, suit and rank, selected from predefined sets.
   - The way cards are compared depends on the rules of the specific game.

   Using the specifications above, implement an abstract class *Card* that publicly inherits *Object* and *Comparable*, and provides:
   - A getter method for each field: suit and rank.
   - A pure virtual setter method for each field.
   - An override of `toString()` method so that a card is displayed in the format: `[`*rank*`:`*suit*`]`.

   Next, implement a class *Solitaire* that publicly inherits *Card* and provides:
   - A static constant string field initialized to `"A23456789TJQK"` to represent the collection of ranks.
   - A static constant string field initialized to `"CDHS"` to represent the collection of suits.
   - A default constructor that initializes the card's rank and suit to the first element of their respective collections
   - Overrides each setter method so that its field is updated only if the provided value exists in its corresponding collection.
   - Overrides the `hash()` method that returns the index of the card's rank within the valid rank string.

3. Case Study Name: Multiplayer
   In many multiplayer games and group activities, players must manage a collection of items that help them achieve game objectives. Each player is uniquely identified and maintains a collection of items that depends on the specific game. In general, a player has the following characteristics:
   - It has a unique fixed identification, which defaults to `"player `*x*`"` where *x* is the current count of players.
   - It maintains a collection of items with a fixed size of at least 3.
   - The type of items varies by game.

   Using the specifications above, implement a generic class *Player* that provides:
   - A constructor that provides an identification and uses the minimum capacity.
   - A constructor that provides a capacity and uses the default identification.
   - A constructor that provides an identification and a capacity.
   - A getter method for both the identification and the capacity.
   - A getter/setter method for the elements of the collection that are accessed with an index.