Your code must follow the general principles outlined in the C++ Coding Guidelines document at
https://www.student.cs.uwaterloo.ca/~cs343/CPPCodingGuidelines.shtml, though you should use C++14 style where appropriate, and favour C++ mechanisms over those in C.

We will be using a subtractive marking scheme when marking your program design and style. This means that you start with full marks for that portion, and then marks are subtracted based upon the violations found in your code/design.

Q1 [30 marks] ADT Design
  a) [3x2=6 marks] Entity- vs Value-Based ADT
  b) [3x3=9 marks] Impact on Design
  c) [3x5=15 marks] Special Member Functions

Q2: [20 marks] - Correctness
Auto-marked by Marmoset tests.

Q2: [20 marks] - Design and Programming Style
Subtractive marking scheme.
- 20 Modularity of ADT family: organization of code among base class and derived classes such that common code is in the base class and derived classes use the parent's code.
- 10 "Don't repeat yourself": Code that is at least 2 statements long, implements a coherent function, and appears in more than one place should be packaged as a helper function.
- 10 Information hiding: all data members should be private; avoid global variables and functions.
- 10 Comments of variables, routines.
- 5 Effective and correct use of C++ dynamic memory allocation to ensure no leaks or dangling pointers.
- 5 Constants: use mnemonic constants, const parameters, const member functions.
- 5 Static: uses static data members and member functions where appropriate.
- 5 Override: uses override keyword on member functions where appropriate.
- 5 Variable names are self-documenting.
- 5 Indentation is consistent.

Q3: [80 marks] - Correctness
[50 marks] for auto-marked test cases
[30 marks] for manual code inspection.

Q3: [20 marks] - Design and Programming Style
Subtractive marking scheme.
- 10 organization of code in classes/structs such that functions are associated with the data they manipulate.
- 10 "Don't repeat yourself": Code that is at least 2 statements long, implements a coherent function, and appears in more than one place should be packaged as a helper function.
- 10 Information hiding: all data members should be private; avoid global variables and functions.
- 10 Comments of variables, routines.
- 5 Effective and correct use of C++ dynamic memory allocation to ensure no leaks or dangling pointers.
- 5 Constants: use mnemonic constants, const parameters, const member functions.
- 5 Static: uses static data members and member functions where appropriate.
- 5 Override: uses override keyword on member functions where appropriate.
- 5 Variable names are self-documenting.
- 5 Indentation is consistent.