OBJECT-ORIENTED PROGRAMMING
HOMEWORK #3
SPRING 2023

Due Monday, March 20

This question is based on the IntArray class code, which can be found at https://pages.ramapo.edu/~ldant/oop/hand2.txt. Take the **IntArray** example and "improve" it. In doing this problem, a major change to the IntArray class that you must make is that the class must keep track of two different size data members. One is the number of array elements dynamically allocated (size) and the other is the number of array elements actually assigned (capacity). This means that you must modify the IntArray code in Handout 2 to reflect this change. In addition, you must write the following functions. You must turn in all the code for this problem. In cases where you must write functions, you need to determine if the functions should be member or global functions. Each function that takes an argument pos (a position in the array) throws a out_of_range error if the position is invalid.

(a) Write a move constructor and a move assignment operator.

(b) Write a constructor that takes an initializer list. It should work like this:

```
IntArray a{3,6,5,8};
```

(c) Write a range constructor. The following illustrates how this constructor works.

```
vector<int> v{5,3,8,6};
list<int> w{4,9,2,3};
IntArray a(v.begin(), v.end());
IntArray b(w.begin(), w.end());
```

(d) Write the const and nonconst versions of the following access functions:

```
at(unsigned pos); //Returns the value at position pos
front(); //Returns the first element of the array
back(); //Returns last element of the array
```

(e) Write the following insert and delete member functions:

```
void insert(unsigned pos, int x); //insert x before 'pos'
void insert(unsigned pos, unsigned n, int x); //n x's
void push_back(int x); //add x to end of array
void pop_back(); // Removes the last element in array
void erase(unsigned pos); //remove the element at 'pos'
void erase(unsigned first, unsigned last); //erase [first,last)
void clear(); //Clears the contents of the array, but the capacity stays the same
```

(f) Write the following member functions

```
unsigned capacity(); //returns the number of possible elements
void reserve(unsigned n); //enlarges capacity to n
void shrink_to_fit(); //Shrinks capacity to equal size
void swap(IntArray &a); //swaps data with IntArray a
```

(g) Write the following overloaded operators which compare arrays using dictionary order: <, <=, >, >=.

(h) Write a main program that tests your class. Your main must show that the following STL functions work on **your** IntArray class: sort, copy, reverse_copy, copy_backward You must email the code for your answer as one file.