Name ____________________________

ET-575 Project 4 – Array Manipulation

Code a functions for a partially filled integer array.

Use the following code as the base for your program:

```
#include <iostream>;
using namespace std;

int main( ) {
    const int CAPACITY = 100;
    int numbers[CAPACITY];
    int numElements = 0;

    cout << "Array Manipulation: " << endl;
    cout << "------------------" << endl;

    append(numbers, numElements, CAPACITY, 10);
    append(numbers, numElements, CAPACITY, 20);
    append(numbers, numElements, CAPACITY, 30);
    append(numbers, numElements, CAPACITY, 40);
    append(numbers, numElements, CAPACITY, 50);
    append(numbers, numElements, CAPACITY, 60);
    append(numbers, numElements, CAPACITY, 70);
    append(numbers, numElements, CAPACITY, 80);
    append(numbers, numElements, CAPACITY, 90);
    append(numbers, numElements, CAPACITY, 100);
    output(numbers, numElements);

    return 0;
}
```

For all functions keep in mind the following:
* CAPACITY should never be exceeded.
* numElements must be updated every time elements are added/removed
* Functions must match specifications for return type and behavior.
* Use appropriate indenting and format of code.

1) Implement a partially filled array of capacity 100.

2) *Append* function initializes the array with values:
   10,20,30,40,50,60,70,80,90,100

3) Looping program menu will output the following:

```
Array Manipulation
------------------
1. Get Index
2. Append
3. Insert
4. Remove
5. Remove First
6. Remove Last
7. Output
8. Exit program

Select: 3
```

4) Options that modify the array should also output the array:

```
Enter an integer value to insert: 2505
Enter a position: 5
Array: 10 20 30 40 50 2505 60 70 80 90 100  Size: 11
```

4) Program will implement all functions as specified on page 3.

5) Program will be tested with input specified on page 4.

Helper functions:

1) **isFull**
   a) returns true if array is full
2) **isEmpty**
   a) returns true if array is empty
3) **getIndex**
   a) returns element index if the element exists, -1 otherwise

Primary functions:

1) **output**
   a) read only parameters
   b) outputs the array as a horizontal row of elements with spaces
   c) outputs the size of the array on the same line
      Example: Array: 10 20 30 40 50 Size: 5

2) **append**
   a) checks if the array is full (use *isFull* function)
   b) if array is not full, add an element to the end of the array

3) **insertAt**
   a) checks if the array is full (use *isFull* function)
   b) if array is not full and the specified index is legal, insert element at index

4) **removeElement**
   a) check if element exists in the array (use *getIndex* function)
   b) if element exists, remove it from the array
   c) return true if element was removed, false otherwise

4) **removeFirst**
   a) remove and returns the first element in the array
   b) return -1 if element does not exist

5) **removeLast**
   a) remove and returns the last element in the array
   b) return -1 if element does not exist

For extra credit:

1. Convert *InsertAt* into a sorted insert function using one of the sorting algorithms discussed class.
2. Alter the *Append* function so that it will only add numbers that are larger then all previous numbers (to maintain sorting order).

Test the program with the following user input:

```
1) Remove First
2) Remove Last
3) Get Index 55
3) Insert 55 into position 4
4) Remove 70
5) Append 110
6) Output
7) End Program
```

Final array output should be:

```
Array: 20 30 40 50 55 60 80 90 110  Size: 9
```