C++ Coding Challenges - Chapter 2

Challenge 1: Basic Functions

Create utility functions for common operations:

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

// TODO: Create function 'square' that takes an int and returns its square

// TODO: Create function 'isPositive' that takes an int and returns true if
positive

// TODO: Create function 'getMin' that takes two ints and returns the smaller
one

int main() {
    // Test your functions
    cout << square(6) << endl;           // Should print 36
    cout << isPositive(-5) << endl;      // Should print 0 (false)
    cout << getMin(10, 7) << endl;       // Should print 7

    return 0;
}
```

Challenge 2: Array Functions

Create functions that work with arrays:

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

// TODO: Create function 'printArray' that prints all elements in an int
array

// TODO: Create function 'findSum' that returns sum of all elements

// TODO: Create function 'findMin' that returns the smallest element

int main() {
    int numbers[5] = {12, 7, 23, 4, 18};
    int size = 5;

    printArray(numbers, size);           // Should print all numbers
    cout << "Sum: " << findSum(numbers, size) << endl;     // Should print 64
    cout << "Min: " << findMin(numbers, size) << endl;     // Should print 4

    return 0;
}
```

Challenge 3: String Processing

Work with arrays of strings:

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

// TODO: Create function 'countStrings' that returns number of strings in
array

// TODO: Create function 'findShortest' that returns the shortest string

// TODO: Create function 'hasString' that checks if array contains a specific
string

int main() {
    string words[4] = {"cat", "elephant", "dog", "butterfly"};
    int size = 4;

    cout << "Count: " << countStrings(words, size) << endl;        // Should
print 4
    cout << "Shortest: " << findShortest(words, size) << endl;    // Should
print "cat"
    cout << "Has 'dog': " << hasString(words, size, "dog") << endl; // Should
print 1

    return 0;
}
```

Challenge 4: Student Grade System

Create a complete grade management system:

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

// TODO: Create function 'calculateAverage' that takes array of grades and
size
// Returns the average as a double

// TODO: Create function 'getLetterGrade' that takes a double average
// Returns char: A (90+), B (80+), C (70+), D (60+), F (<60)

// TODO: Create function 'countPassing' that takes array of grades and size
// Returns number of grades >= 60

int main() {
    int grades[6] = {85, 92, 78, 96, 58, 87};
    int size = 6;

    double avg = calculateAverage(grades, size);
    char letter = getLetterGrade(avg);
```

```
    int passing = countPassing(grades, size);

    cout << "Average: " << avg << endl;
    cout << "Letter Grade: " << letter << endl;
    cout << "Passing Grades: " << passing << endl;

    return 0;
}
```

Solutions

Challenge 1 Solutions:

```
int square(int num) {
    return num * num;
}

bool isPositive(int num) {
    return num > 0;
}

int getMin(int a, int b) {
    if (a < b) {
        return a;
    } else {
        return b;
    }
    // Alternative: return (a < b) ? a : b;
}
```

Challenge 2 Solutions:

```
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;
}

int findSum(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += arr[i];
    }
    return sum;
}

int findMin(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }
```

```
    return min;
}
```

Challenge 3 Solutions:

```
int countStrings(string arr[], int size) {
    return size;
}

string findShortest(string arr[], int size) {
    string shortest = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i].length() < shortest.length()) {
            shortest = arr[i];
        }
    }
    return shortest;
}

bool hasString(string arr[], int size, string target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return true;
        }
    }
    return false;
}
```

Challenge 4 Solutions:

```
double calculateAverage(int grades[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += grades[i];
    }
    return (double)sum / size;
}

char getLetterGrade(double average) {
    if (average >= 90) return 'A';
    else if (average >= 80) return 'B';
    else if (average >= 70) return 'C';
    else if (average >= 60) return 'D';
    else return 'F';
}

int countPassing(int grades[], int size) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (grades[i] >= 60) {
            count++;
        }
    }
    return count;
}
```

Bonus Challenge: Array Manipulation

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

// Bonus: Create function 'reverseArray' that reverses an array in place
void reverseArray(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
        int temp = arr[i];
        arr[i] = arr[size - 1 - i];
        arr[size - 1 - i] = temp;
    }
}

// Bonus: Create function 'sortArray' that sorts array in ascending order
void sortArray(int arr[], int size) {
    // Simple bubble sort
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

// Test the bonus functions
int main() {
    int numbers[5] = {64, 34, 25, 12, 22};
    int size = 5;

    cout << "Original: ";
    printArray(numbers, size);

    reverseArray(numbers, size);
    cout << "Reversed: ";
    printArray(numbers, size);

    sortArray(numbers, size);
    cout << "Sorted: ";
    printArray(numbers, size);

    return 0;
}
```