Assignment 1

Complete the following programming exercises in C/C++. Aim to make your code as concise and logical as possible. Write functions to be self-contained units of code, i.e. the code inside your functions must **not** rely on variables you define outside the function (i.e. no *global variables*). You can test your code by calling your functions from your main() method, but no code written in main will be graded.

Submit your answers as a ***single text .TXT document***, with **each answer clearly numbered and your code properly formatted/indented** (use Ctrl+A then Ctrl+I in Code::Blocks to properly format your code).

This assignment is due 6:00 pm Friday 14th February. No late submissions accepted under any circumstances.

Q1. Write a function of the form

```
void printSpacedString(char str[]) { /* your code here */ }
```

which takes as input any C-style string (i.e. an array of char terminated with a `'\0'` character) and prints each character in the string separated by a space. At the end of the line, your function should then print the number of characters in the string. Test your function by calling it from main() as `printSpacedString("Hello World!");` which should print out exactly the following

```
H e l l o   W o r l d ! 12
```

Q2. Write a function of the form

```
int largest(int a, int b, int c, int d) { /* your code */ }
```

which returns one of the values a, b, c or d, whichever is the largest.

Q3. Write a function of the form

```
void minMax(const double arr[], const int size, double & minVal, double & maxVal) { ... }
```

which finds the minimum and maximum values in the array arr, using the **pass-by-reference** parameters minVal and maxVal as the outputs from the function. The size parameter specifies the length the array. You can test your function by putting the following code into your main() function:

```
double arr[] = {-4.3,7.0,10.4,3.9,1.2,9.8,11.3};
double min, max;
minMax(arr, sizeof(arr)/sizeof(arr[0]), min, max);
cout << "Q3: array min is " << min << " and max is " << max << endl;
```

Q4. By using ***pass-by-reference***, write a function of the form

```
void swap( ... ) { ... }
```

which takes two integer parameters, passed by reference, and swaps their values. For example, the following test code (written in man()) should print the line "Swapped, now a = 20 and b = 10"

```
int a = 10;
int b = 20;
swap(a,b); // calling your swap function
cout << "Swapped, now a = " << a << " and b = " << b << endl;
```

Note: The standard C++ library (in the **std** namespace) also defines a method called *swap*. To avoid any conflict with your method, don't make a global directive "using namespace std;" in your file. Instead, use "using std::cout;", etc., to declare only what is needed from the std namespace. (see the notes on namespaces).

Q5. Write a new version of the swap function **using pointers** to pass the two integers to the function ("pass-by-pointer"). Your function should work correctly when you run the following test code from main():

```
    int c = 100;
    int d = 200;
    swap(&c,&d);
    cout << "Swapped, now c = " << c << " and d = " << d << endl;
```

We can write the two functions of Q4 and Q5, both called “swap”, in the same C/C++ file and same namespace without causing any naming conflict. **Why is there no conflict here?**

Q6. Write a method

```
double range(double data[], int size)
```

which returns the *range* of the values contained in the double array and where **size** specifies the size of the array passed to the function. For example, passing the array of values -1.0, 6.3, -2.5, 10.7, 8.7 (which has size 5) should return the value 13.2 (equal to the max value minus the min value in the array). **Hint**: Your function code can call your function from **Q3** to help do this.

Q7. By **using the bit-wise AND operator &**, write a function of the form

```
unsigned char getLowerByte(int num) { … }
```

which takes the positive integer value **num** and returns the least significant (right-most) byte of the number. Note: only one line of code is needed to implement this function. To test your function and print the value as an integer, use the following test code in your main() method:

```
cout << "Lower byte = " << (int)getLowerByte(0xFFFFFF) << endl;
```

Q8. By using **the right bit shift operator >> and a loop**, write a function of the form

```
unsigned short significantBits(unsigned int num) { ... }
```

which returns the number of bits needed to represent a given decimal number **num**. For example, passing the decimal number 1000 to your function should return the value 10, as the binary representation of 1000 in base 10 is equal to 1111101000 in base 2.

Q9. Write a method

```
bool isOrdered(int a[], int size) { ... }
```

which checks if the elements of an array (which contains **size** elements) are all in order. The function returns true if all elements are all in ascending order and returns false otherwise.

Q10. Write a method

```
void checkers(int N) { ... }
```

that prints an **N x N** checkered pattern, as shown below. Note: you can use the special ASCII character 178 (decimal) to print one type of square and the space character to the other. (To get a square pattern on the screen you may need to print two of each character type side by side for each individual square).

```
▓▓  ▓▓  ▓▓  ▓▓  
  ▓▓  ▓▓  ▓▓  ▓▓
▓▓  ▓▓  ▓▓  ▓▓  
  ▓▓  ▓▓  ▓▓  ▓▓
▓▓  ▓▓  ▓▓  ▓▓  
  ▓▓  ▓▓  ▓▓  ▓▓
▓▓  ▓▓  ▓▓  ▓▓  
  ▓▓  ▓▓  ▓▓  ▓▓
```

The pattern above is produced by calling `checkers(8);`

Submission status

Submission status	Submitted for grading