Q1. Write a method in C++

```
void reverseArray(double a[], int size) { ... }
```

which reverses the order of the elements in an array. For example, this test code in main()

```
double array[] = {1.2, 2.3, 3.4, 4.9, 5.3};
reverseArray(array,5);
for (int i=0; i<5; i++)
   cout << array[i] << ",";
```

should print out:

```
5.3,4.9,3.4,2.3,1.2
```

Q2. Write a method in C++ which calls the putLargestFirst() method from Q9 of Assignment 1 to sort an array into ascending order (smallest number first). You may also call your reverseArray() method. Your method should have the following form:

```
void sort(double a[], int size) {
   // code that calls putLargestFirst(), multiple times and reverseArray(), if required
}
```

You should write some test code in main() to check your method works, but you do not need to submit your test code.

Q3. A segment of code is given below. Explain each line of code and its affect on the array contents, if any

```
1.    double arr[] = {1.0,2.0,3.0,4.0,5.0};
2.    double* ptr1 = arr;
3.    double* ptr2 = &arr[4];
4.    *ptr1 = *ptr2;
5.    *++ptr1 = 5.0;
6.    *--ptr2 = *arr;
```

Q4. Using the bit shift operators, write a method in C++

```
unsigned int getByte(unsigned int num, unsigned int byteIndex) { . . .}
```

which takes an integer num and a byteIndex and returns the corresponding byte of the number. The rightmost byte is index 0 and the leftmost index 3. For example, the following test code in main():

```
cout << hex << getByte(0x87654321, 0) << endl;
cout << hex << getByte(0x87654321, 1) << endl;
cout << hex << getByte(0x87654321, 2) << endl;
cout << hex << getByte(0x87654321, 3) << endl;
```

Should print out:

```
21
43
65
87
```

Q5. Write a C++ function:

```
void swap(int& a, int& b) { … }
```

which swaps the values of two integer variables passed to the function. For example, calling the following test code from main()

```
int a = 1, b = 2;
swap(a, b);
cout << a << "," << b << endl;
```

should print: 2,1

Q. Explain why there is are &s in the parameter list of the function. Why are they needed for this function to work?

Q6. Write a C++ function:

```
void sortAscending(int& a, int& b, int& c) { . . . }
```

Which swaps the values of the integer arguments a, b and c so that they are in ascending order. Implement the function using your swap() method (i.e. call swap() from your sortAscending() method code).

For example the following test code in main()

```
int a=8, b=24, c=2;
sortAscending(a,b,c);
cout << a << "," << b << "," << c << endl;
```

Should print out:

```
2,8,24
```

Q7. Write a C++ function:

```
string removeChar(const string & s, const char & c) { . . . }
```

which removes all occurrences of the character **c** in the given string **s** and returns the result as a new string. Use only basic string features in your implementation: string.length() and the indexing operator [].

Q. What does it mean to define the function parameters as "const string & s" and "const char & c"? I.e. what does the combination of const and & mean in this context?

Q8. Write a C++ class called Complex, which represents a complex number. Your class should have the following features:

- Two private data members to store the real and imaginary parts of a complex number
- A parameterless constructor, which initializes a new Complex number to the number 0 + 0j
- A constructor which allows a new Complex number to be initialized with given real and imaginary values
- A public setValue() method, which allows a Complex number's value to be changed.

Q9. Add a new method to your Complex class, called print() which, when called on a Complex object, prints out the complex number value in a neat form. For example, the following test code in main()

```
Complex c1(4,5), c2(2,-4), c3(3,0);
c1.print();
cout << endl;
c2.print();
cout << endl;
c3.print();
cout << endl;
```

should print out exactly:

```
4 + 5j
2 - 4j
3
```

Note: Add your new method to your Complex class by defining it separately outside the class, in the form:

```
void Complex::print() { . . . }
```

Q10. Write a function in C++:

```
int quadraticRoots(double a, double b, double c, Complex &r1, Complex &r2) { . . . }
```

which solves a quadratic equation of the form $\mathbf{a}x^2 + \mathbf{b}x + \mathbf{c} = 0$. The roots are returned via the parameters **r1** and **r2** and the method should return the number of roots (either 1 or 2) using a return statement.

For example, the following test code in main():

```
Complex r1, r2;
int num_roots = quadraticRoots(11, 3, 10, r1, r2);
cout << "The quadratic has " << num_roots << " root(s): ";
r1.print();
if (num_roots == 2) { cout << " and "; r2.print(); }
cout << endl;
```

Should give the printout:

```
The quadratic has 2 root(s): -0.136364 - 0.943661j and -0.136364 + 0.943661j
```