Exercise 2 C++ Classes

1. What are the top two advantages of using tuples?

a) As return types of functions.
b) As input arguments to functions.
c) To align data on word boundaries.
d) To avoid copying data.

2. In which of the following situations would we use tuples instead of structs or classes?

a) When we wish to create lists of logically-related data in client code.
b) When we are not concerned with nested data structures.
c) When the tuple data is just a repository.
d) A tuple is similar to a class but with no member functions.

3. Which of the following functionality can be used to construct tuples?

a) `std::make_tuple`.
b) Initializer lists.
c) Tuple concatenation.
d) Using `std::tie` to create a tuple of *lvalue* references.

4. (Explicit Constructors)

Consider the following class:

```
struct B
{
    explicit B(int) {}
    explicit B(int, int) {}
    explicit operator int() const { return 0; }
};
```

Create a program to test the class. Which lines of code below compile and which ones do not? How would you modify the class to ensure that they do compile? Otherwise, how can these lines be modified so that the code compiles error-free without modifying the class?

```
B b1 = 1;
B b2(3);
B b3{ 7,3 };
B b4 = (B)42;
```

5. Which of the following statements are true?

a) `constexpr` applies to both objects and functions.
b) All `const` objects are also `constexpr` objects.
c) The output from a `constexpr` function will be known at compile-time if its input arguments are known at compile-time.
d) `constexpr` functions need not necessarily produce values that are known at compile-time.

6. Explain why the following member function of class `Point` does not compile:

```
constexpr double Distance(const Point& pt2) const
{
   return std::sqrt((x - pt2.x)*(x - pt2.x) + (y - pt2.y)*(y - pt2.y));
}
```

Furthermore, does the following code compile?

```
constexpr Point p2(1.0, 2.0);
```

```
    double newVal = 3.0;
    p2.X(newVal);

    Point p3(1.0, 2.0);
    double newVal2 = 3.0;
    p3.X(newVal2);
```

7. Explain what the output of the following code is:

```
double S = 2.0; double K = 65.0; double h = 1.0;

double sum = 0.0;
for (int i = 1; i <= 100; ++i)
{
    sum += std::sqrt(S / K); S -= h;
    std::cout << i << ",";
}

std::cout << std::boolalpha << "Sum: " << sum << '\n';
```

8. Consider the following code:

```
// Exception handling
{
    std::cout << "STL exception time\n";
    double factor = 2.0;
    double val
          = factor*std::numeric_limits<double>::max(); // infinity
    val -= val; // NaN

    try
    {
       double b = std::exp(val);
    }
    catch (std::domain_error& e)
    {
       // 1.
       std::cout << "Error " << e.what() << std::endl;
    }
    catch (...)
    {
       // 2.
       std::cout << "Unexpected error " << std::endl;
    }
    std::cout << "end " << std::endl;
}
```

When this code is run, what happens?

a) A domain error occurs.
b) An unexpected error occurs.
c) The program crashes.
d) The program runs to completion without throwing an exception.