Question 1

Which statement is false about boost?

1.  Boost is an open-source library and can be used freely in non-commercial and commercial applications.
2.  The boost library is largely implemented using templates.
3.  Boost is cross-platform and has support for most modern C++ compilers.
4.  Boost is a C++ library that is standard available in C++ compilers.
5. You answered this question correctly.

Question 2

How many libraries does boost contain?

1.  76
2.  33
3.  Boost is one big library.
4.  More than 100
5. You answered this question correctly.

Question 3

Which two statements are true about boost::shared\_ptr?

1.  A shared pointer deletes the object it is pointing to automatically when the shared pointer object gets out of scope.
2.  Shared pointers use a garbage collector to clean up memory.
3.  A shared pointer deletes the object it is pointing to automatically when the last shared pointer pointing to that object gets out of scope.
4.  Shared pointers use reference counting to determine when an object is not referenced anymore.
5. You answered this question correctly.

Question 4

What is the correct syntax to create a boost shared pointer to an *MyClass* object?

1.  boost:shared\_ptr<MyClass> mc(MyClass());
2.  boost:shared\_ptr<MyClass\*> mc(new MyClass);
3.  boost:shared\_ptr<MyClass> mc(new MyClass);
4.  boost:shared\_ptr<MyClass> mc=new MyClass;

Question 5

Which statement is false about boost tuples?

1.  The tuple *get<>()* member function can be used to set the element values.
2.  A tuple is a fixed-sized collection of elements which can each have a different type.
3.  A tuple defined for *n* elements can contain zero till *n* values.
4.  A tuple makes it easy to make a function that returns more than one value.
5. You answered this question correctly.

Question 6

What is the incorrect syntax to create a tuple with a double, int and string?

1.  boost::tuple<double, int, string> t(3.14, 10, string("Hello"));
2.  boost::tuple<double, int, string> t;
3.  boost::tuple<double, int, string> t=boost::make\_tuple(3.14, 10, string("Hello"));
4.  boost::tuple t=boost::make\_tuple(3.14, 10, string("Hello"));
5. You must specify for each element the type as template argument to the tuple
6. You answered this question correctly.

Question 7

Which statement is false about boost variants?

1.  A variant can contain one value of any type.
2.  When retrieving a value from the variant, we can use the *get<T>()* global function.
3.  A variant can contain one value of a given collection of types.
4.  A variant is like a type-safe C union and boost variant can contain class types while a union can only contain the build-in data types.

Question 8

What is the incorrect syntax to create a variant for a double, int or string or the syntax to extract the value?

1.  boost::variant<double, int, string> v(40.0);
2.  double d=boost::get(v);
3.  double d=boost::get<double>(v);
4.  boost::variant<double, int, string> v=40.0;
5. You answered this question correctly.

Question 9

Which statement is false about the boost random library?

1.  The boost random library provides various algorithms for generating random numbers.
2.  To get random numbers in a specific range, you need to use a distribution in combination with the random generator.
3.  The numbers generated by boost random number generators are really random.
4.  To get different random numbers each time you need to set the seed of a random number generator.
5. You answered this question correctly.

Question 10

Which statement is false about random distributions?

1.  The *discrete\_distribution* transforms the random numbers to a set of numbers where the chance to get each value can be different.
2.  The *uniform\_real\_distribution* transforms the random numbers to floating point values in a specific range.
3.  We are required to use a distribution in combination with a random number generator.
4.  The *uniform\_int\_distribution* transforms the random numbers to integers in a range where the chance to get each integer value is the same.

Result