Setting QML object's **z** property to negative value will cause that:  
 this object goes to top in case of overlapping siblings objects,  
 this object goes to bottom in case of overlapping sibling objects,  
 It impossible, **z** property can only have positive value.

What happens when you provide the signal body implementation?  
 Signals don't have implementation, so the code will be ignored by compiler,  
 The code may don't compile or don't link, depending on where the implementation was done in source code,  
 The implementation will overwrite usual implementation done by moc and the code will not work as intended.

What is impossible regarding signals and slots?  
 Signal to signal connection,  
 Connecting signal with no parameter to the slot with parameter, like:  
connect(slider, SIGNAL(updated()), spinbox, SLOT(valueChanged(int)));  
 Connecting signal with a parameter to the slot with no parameters, like:  
connect(slider, SIGNAL(valueChanged(int)), spinbox, SLOT(update()));

Choose false sentence about moc.  
 Moc is a Meta-Object Compiler,  
 Moc produces moc files, which are ordinary C++ files,  
 Moc is unable to implement signals and slots for class templates,  
 Moc expands #define-s normally, so type macros can be used as signal argument.

Qt Quick is a collection of several technologies. Which one is not a part of it?  
 QML,  
 html,  
 javascript,  
 Qt C++.

QML object's **id** attribute:  
 can be read and set during runtime just like any other QML object attribute,  
 can begin with a lowercase or uppercase letter or an underscore,  
 is optional, but recommended.

What happens, when you try connect signal and slot, which are already connected (the same sender and receiver)?  
 Duplicate connection will be created, and for every signal emit() the corresponding slot will be called two times,  
 connect() will fail and return false; the connection will remain single,  
 If the connection type was Qt::UniqueConnection, the code will not compile.

All visual items in Qt Declarative (QML) inherit from:  
 Item,  
 Component,  
 QWidget.

QGraphicsItem::ItemHasNoContents flag enabled means that:  
 the QGraphicsItem object has no properties defined,  
 the attempt to paint the QGraphicsItem failed,  
 the item does not paint anything,  
 the item is a panel.

Q\_GADGET macro:  
 allows to have signal and slots for the classes that do not inherit from QObject,  
 enables some features of QMetaObject, i.e. Q\_ENUM, Q\_PROPERTY and Q\_INVOKABLE,  
 can be put only into the QObject subclasses.

Qt5 introduces new style of connect with the use of function pointers, like presented below:

connect( slider, &QSlider::valueChanged, spinbox, &QSpinBox::setValue );

The old Qt4 style was using SIGNAL() and SLOT() macros:

connect(slider, SIGNAL(valueChanged(int)), spinbox, SLOT(setValue(int)));

One of the advantages of Qt5 style is that:  
 it allows to connect overloaded slots easier,  
 errors can be detected earlier, at compilation stage,  
 Q\_OBJECT macro is not needed for signals and slots anymore.

Choose true sentence about qmake:  
 It is possible to generate both - Makefile and .pro file - with the use of qmake.  
 Qmake can generate Makefile, but it is not possible to generate .pro file with qmake,  
 Qmake compiles the project.

What will happen if you instantiate QVector <QWidget>?  
 The compiler will complain that QWidget's copy constructor and assignment operators are disabled,  
 The compiler will complain that QWidget is not registrated as meta-object,  
 The code will compile, but the program will result in undefined behaviour.

Which Qt class allows to start external program from Qt application?  
 QProcess,  
 QPluginLoader,  
 QRunnable.

Which one class is not a part of the QtCore?  
 QFile,  
 QThread,  
 QColor.