Exercise 00:

Write a static class ScalarConverter that will contain a method "convert" takes as parameter a string representation of a C++ literal in its most common form. This literal must belong to one of the following a scalar types:

- char
- int
- float
- double

Except for char parameters, only the decimal notation will be used.
Examples of char literals: ’c’, ’a’, ...
To make things simple, please note that non displayable characters shouldn’t be used as inputs. If a conversion to char is not displayable, prints an informative message.
Examples of int literals: 0, -42, 42...
Examples of ﬂoat literals: 0.0f, -4.2f, 4.2f...
You have to handle these pseudo literals as well (you know, for science): -inff, +inff and nanf.
Examples of double literals: 0.0, -4.2, 4.2...
You have to handle these pseudo literals as well (you know, for fun): -inf, +inf and nan.
Write a program to test that your class works as expected.
You have to first detect the type of the literal passed as parameter, convert it from string to its actual type, then convert it explicitly to the three other data types. Lastly, display the results as shown below.
If a conversion does not make any sense or overflows, display a message to inform the user that the type conversion is impossible. Include any header you need in order to handle numeric limits and special values.

Exercise 01:

Implement a static class Serializer with the following methods:
uintptr_t serialize(Data* ptr);
It takes a pointer and converts it to the unsigned integer type uintptr_t.
Data* deserialize(uintptr_t raw);
It takes an unsigned integer parameter and converts it to a pointer to Data.
Write a program to test that your class works as expected.
You must create a non-empty (it means it has data members) Data structure.
Use serialize() on the address of the Data object and pass its return value to deserialize(). Then, ensure the return value of deserialize() compares equal to the original pointer.
Do not forget to turn in the files of your Data structure.

Exercise 02:

Implement a Base class that has a public virtual destructor only. Create three empty classes A, B and C, that publicly inherit from Base.
Implement the following functions:
Base * generate(void);
It randomly instanciates A, B or C and returns the instance as a Base pointer. Feel free to use anything you like for the random choice implementation.
void identify(Base* p);
It prints the actual type of the object pointed to by p: "A", "B" or "C".
void identify(Base& p);
It prints the actual type of the object pointed to by p: "A", "B" or "C". Using a pointer inside this function is forbidden.
Including the typeinfo header is forbidden.
Write a program to test that everything works as expected.