Remember that the quality of the defenses, hence the quality of the of the school on the labor market depends on you. The remote defences during the Covid crisis allows more flexibility so you can progress into your curriculum, but also brings more risks of cheat, injustice, laziness, that will harm everyone's skills development. We do count on your maturity and wisdom during these remote defenses for the benefits of the entire community.

# SCALE FOR PROJECT CPP MODULE 07 (/PROJECTS/CPP-MODULE-07)

You should evaluate 1 student in this team

★

Git repository

```
git@vogsphere-v2.42.fr:vogsphere/intra-uuid-45211463-6576
```

## Introduction

- Only grade the work that is in the student or group's GiT repository.

- Double-check that the GiT repository belongs to the student or the group. Ensure that the work is for the relevant project and also check that "git clone" is used in an empty folder.

- Check carefully that no malicious aliases were used to fool you and make you evaluate something other than the content of the official repository.

- To avoid any surprises, carefully check that both the evaluating and the evaluated students have reviewed the possible scripts used to facilitate the grading.

- If the evaluating student has not completed that particular project yet, it is mandatory for this student to read the entire subject prior to starting the defence.

- Use the flags available on this scale to signal an empty repository, non-functioning program, a norm error, cheating etc. In these cases, the grading is over and the final grade is 0 (or -42 in case of cheating). However, except for cheating, you are encouraged to continue to discuss your work (even if you have not finished it) to identify any issues that may have caused this failure and avoid repeating the same mistake in the future.

- Remember that for the duration of the defense, no segfault,
no other unexpected, premature, uncontrolled or unexpected
termination of the program, else the final grade is 0. Use the
appropriate flag.
You should never have to edit any file except the configuration file if it exists.
If you want to edit a file, take the time to explicit the reasons with the
evaluated student and make sure both of you are okay with this.

- You must also verify the absence of memory leaks. Any memory allocated on the heap must
be freed before the end of execution.
You are allowed to use any of the different tools available on the computer, such as
leaks, valgrind, or e_fence. In case of memory leaks, tick the appropriate flag.

# Disclaimer

Please respect the following rules:

- Remain polite, courteous, respectful and constructive
throughout the evaluation process. The well-being of the community
depends on it.

- Identify with the person (or the group) evaluated the eventual
dysfunctions of the work. Take the time to discuss
and debate the problems you have identified.

- You must consider that there might be some difference in how your
peers might have understood the project's instructions and the
scope of its functionalities. Always keep an open mind and grade
him/her as honestly as possible. The pedagogy is valid only and
only if peer-evaluation is conducted seriously.

# Guidelines

You must compile with clang++, with -Wall -Wextra -Werror
As a reminder, this project is in C++98
C++11 (and later) member functions or containers are NOT expected.

Any of these means you must not grade the exercise in question:
- A function is implemented in a header (except in a template)
- A Makefile compiles without flags and/or with something other than clang++

Any of these means that you must flag the project as Forbidden Function:
- Use of a "C" function (*alloc, *printf, free)
- Use of a function not allowed in the subject
- Use of "using namespace" or "friend"
- Use of an external library, or C++20 features

---

# Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/27902/en.subject.pdf)

main.cpp (/uploads/document/document/4566/main.cpp)

# Exercise 00: A few functions

*In this exercise, the student must write 3 simple function templates: swap, min and max.*

**Simple types**

Refer to the subject for the expected output with simple types, such as int.

Yes No

**Complex types**

Do the functions also work with complex types such as:

```
class Awesome
{
public:
Awesome(void) : _n(0) { }
Awesome( int n ) : _n( n ) { }
Awesome & operator= (Awesome & a) { _n = a._n; return *this; }
bool operator==( Awesome const & rhs ) const { return (this->_n == rhs._n); }
bool operator!=( Awesome const & rhs ) const{ return (this->_n != rhs._n); }
bool operator>( Awesome const & rhs ) const { return (this->_n > rhs._n); }
bool operator<( Awesome const & rhs ) const { return (this->_n < rhs._n); }
bool operator>=( Awesome const & rhs ) const { return (this->_n >= rhs._n); }
bool operator<=( Awesome const & rhs ) const { return (this->_n <= rhs._n); }
int get_n() const { return _n; }
private:
int _n;
};
std::ostream & operator<<(std::ostream & o, const Awesome &a) { o << a.get_n(); return o; }

int main(void)
{
Awesome a(2), b(4);

swap(a, b);
std::cout << a << " " << b << std::endl;
std::cout << max(a, b) << std::endl;
std::cout << min(a, b) << std::endl;
return (0);
}
?
```

Yes No

# Exercise 01: Iter

*This exercise aims to write a generic iteration function through arrays.*

---

**Does it work???**

Test the following code with the student's iter:

```
class Awesome
{
public:
Awesome( void ) : _n( 42 ) { return; }
int get( void ) const { return this->_n; }
private:
int _n;
};
std::ostream & operator<<( std::ostream & o, Awesome const & rhs ) { o << rhs.get(); return o; }
template< typename T >
void print( T const & x ) { std::cout << x << std::endl; return; }
int main() {
int tab[] = { 0, 1, 2, 3, 4 }; // <--- I never understood why you can't write int[] tab. Wouldn't that make more sense?
Awesome tab2[5];

iter( tab, 5, print );
iter( tab2, 5, print );

return 0;
}
```

If everything went well, it should display:

```
0
1
2
3
4
42
42
42
42
42
```

☑ Yes ☒ No

---

# Exercise 02: Array

*In this exercise, the student must write a class template that behaves like an array. If the inner allocation of the actual array does not come from a use of new[], don't grade this exercise. Ask the student to prove the class template works with arrays of simple and complex types before grading.*

---

**Constructors**

Is it possible to create an empty array and an array of a specific size?

Yes No

**Access**

Elements must be accessible for reading and writing through the operator[] (or just for reading if the instance is const). Access to an element out of the limits must throw an std::exception.

Yes No

# Ratings

**Don't forget to check the flag corresponding to the defense**

Ok Outstanding project

Empty work No author file Invalid compilation Norme Cheat Crash

Leaks Forbidden function

# Conclusion

**Leave a comment on this evaluation**

Tres bien, module bien compris et expliqué,

**Finish evaluation**