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

You should evaluate 1 student in this team

★

Git repository

Introduction

Please comply with 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 student or group whose work is evaluated the possible dysfunctions in their project. Take the time to discuss and debate the problems that may have been identified.

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

Guidelines

- Only grade the work that was turned in the Git repository of the evaluated student or group.

- Double-check that the Git repository belongs to the student(s). Ensure that the project is the one expected. Also, check that 'git clone' is used in an empty folder.

- Check carefully that no malicious aliases was used to fool you and make you evaluate something that is not the content of the official repository.

- To avoid any surprises and if applicable, review together any scripts used

to facilitate the grading (scripts for testing or automation).

- If you have not completed the assignment you are going to evaluate, you have to read the entire subject prior to starting the evaluation process.

- Use the available flags to report an empty repository, a non-functioning program, a Norm error, cheating, and so forth.
In these cases, the evaluation process ends and the final grade is 0, or -42 in case of cheating. However, except for cheating, student are strongly encouraged to review together the work that was turned in, in order to identify any mistakes that shouldn't be repeated in the future.

- 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 properly 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.

Attachments

subject.pdf (https://cdn.intra.42.fr/pdf/pdf/47334/en.subject.pdf)

main.cpp (/uploads/document/document/8357/main.cpp)

Preliminary tests

If cheating is suspected, the evaluation stops here. Use the "Cheat" flag to report it. Take this decision calmly, wisely, and please, use this button with caution.

Prerequisites

The code must compile with c++ and the flags -Wall -Wextra -Werror
Don't forget this project has to follow the C++98 standard. Thus, C++11 (and later) 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 file (except for template functions).
- A Makefile compiles without the required flags and/or another compiler than c++.

Any of these means that you must flag the project with "Forbidden Function":

- Use of a "C" function (*alloc, *printf, free).
- Use of a function not allowed in the exercise guidelines.
- Use of "using namespace " or the "friend" keyword.
- Use of an external library, or features from versions other than C++98.

☑ Yes ☒ No

Exercise 00: Start with a few functions

This exercise is about writing 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 is about writing a generic function to iterate through arrays.

Does it work?

Test the following code with the evaluated 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

This exercise is about writing 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 evaluated student to prove the class template works with arrays of both simple and complex types before grading the exercise.

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 reading only if the instance is const). Access to an element which is out of range must throw an std::exception.

☑ Yes ☒ No

Ratings

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

✔ Ok ★ Outstanding project

Empty work Incomplete work Invalid compilation Cheat Crash

Concerning situation Leaks Forbidden function

Conclusion

Leave a comment on this evaluation

Finish evaluation