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 01

You should evaluate 1 student in this team

Git repository

```
git@vogsphere-v2-bg.13
```

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, with the exception of cheating, you are encouraged to continue to discuss your work (even if you have not finished it) in order 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 defence, 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 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.

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 and C++20 members 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

ex00

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (For example, using console output, etc... The student has to be able to explain how it proves anything). If it does not, you MUST NOT grade this part.

ex00

There must be a ponyOnTheHeap function that allocates a new Pony
using "new", then deletes it (using "delete", obviously).
There must be a ponyOnTheStack function that allocates a new Pony on the stack (WITHOUT
using "new" or "malloc").

Yes | No

ex01

ex01

Theoretically, two choices here are right: Either changing the allocation
to be on the stack and not on the heap (So, don't use "new" anymore, and handle
a std::string without using the pointer), or adding a "delete panthere;" after
the std::cout.
Ask the student to explain WHY he did what he did before marking
this as done. He has to answer something other than "Meh, it just works".

Yes | No

ex02

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex02

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
The Zombies must be destroyed when appropriate.
In newZombie, it should be allocated on the heap, returned,
and then deleted in the main(). The student must explain why.
The Zombies created by randomChump must either be allocated on the stack (so implictly deleted
at the end of the function), or allocated on the heap then explicitly deleted.
The student must justify his choice.

Yes | No

ex03

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex03

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
The Zombies must be allocated

in the constructor of the ZombieHorde, and should be allocated as an array,
either on the stack, either explictly using new[], in which case they should
be deleted in the destructor. The student must explain his choice.

Yes | No

ex04

ex04

There is a string containing "HI THIS IS BRAIN", then a pointer
to it, then a reference to it, and it is displayed through the pointer then
through the reference. As the subject says, really, that's it, no tricks
or anything.

Yes | No

ex05

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex05

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
The "identify" function
in the "Brain" must return the representation of "this", or any other trick
that equates to "the adress of the current instance".
The "getBrain"
function should return a REFERENCE to the Brain of the current Human. With the
main() that the subject provides, it must, as the subject says, display two
identical adresses.
The student should be able to explain why he did this.

Yes | No

ex06

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc ...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex06

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
The student must choose to store the Weapon either as pointer
or as a reference in HumanA and HumanB.
In HumanA, BOTH are acceptable if justified, even if theoretically the reference
is better, since the Weapon exists from creation until destruction and never
changes.
In HumanB, only the pointer is acceptable, since the field is not
set at creation time, so it can not be a reference.
The student must justify his choices correctly.

Yes | No

ex07

ex07

The program must work as the subject specifies. A reasonable amount
of errors must be handled. If you can find an error that isn't handled, and
isn't completely esoteric, no points for this exercise.
The program must read from the file using an ifstream or equivalent,
and write using an ofstream or equivalent."

Yes | No

ex08

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex08

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
The "action" function must use an array of pointer to member functions
to choose which action should be called. Any if/elseif/elseif/else or
other crap like this counts as wrong.

Yes | No

ex09

As the subject says, this exercise requires the student to turn in a "main" function, and it must, when run, demonstrate that the exercise works as intended (for example, using console output, etc...). The student has to be able to explain how it proves anything. If it does not, you MUST NOT grade this part.

ex09

All the classes and functions required by the subject must exist
and work as specified, otherwise, no points for this exercise.
Must work exactly as the subject requires.
As with the previous exercise, the action to take when
using "log" must be determined using an array of pointers to member functions.

Yes | No

ex10

ex10

The program must work as the subject specifies. Any error that
isn't handled = no points for this exercise."

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

Finish evaluation