1 Flags

- "pedantic" flag forces you to use vanilla C++.
- "Wall" flag will give you a bunch of warnings if they appear.

2 Defensive coding

- We can put "const" on methods that only return (a la a getter). This is useful as we can track if we accidentally tried modifying something with a constant method, which claims that the function will NOT modify any fields.
- Overloading versus overriding. If you're doing one, make sure you don't accidentally do the other with with the override keyword (think virtual keyword!).
- Be careful with arrays! Like C, you can do dumb shit with that and its elements.
- Same with Vectors, but you can instead use the ".at()" function to insure that you're checking only valid memory.

3 Debugging problems

- Print statements can only get you so far, but they do work. Kinda.
- In Unix, remember that outputs are buffered. If the program crashes, it may not output exactly where it crashed. Instead, print to stderr.
- Note that file outputs are also buffered. If you forget to close the file, whether it be shit code, crash, etc, the buffer may not be flushed, so your output files may be empty!
- This is because constant writing to a file is really inefficient.
- In Bash, typing in 'script' will save your inputs/outputs to a file.
- Hate removing and adding in print statements? What you can do is the 'DDEBUG=true' flag (debug).
- Assertions: Recall that we can use a string as an extra "condition" to give us information to what went wrong.
- Use the flag 'DNDEBUG' to turn assertions off.

4 GDB

- Use the '-g' flag while compiling and linking.
- We can read in a sequence of commands from a file whilst compiling with gdb.

5 Valgrind

- C++ is retarded sometimes. When checking leaks, as long as ALL the lost catagories are 0, you're fine.