Ryland Nezil
V00157326
June 20, 2023
SENG 475 A01

Assignment 3

6.1

```
void func() {
     initialize(); // perform initialization
     do_work(); // do some work
     cleanup(); // perform any necessary cleanup
}
```

There is no mechanism to **ensure** that the *necessary cleanup* is performed 100% of the time. If anything goes wrong inside initialize() or do_work() and, God forbid, an exception is thrown out of one of these functions, then func() will immediately end without performing cleanup. One potentially likely result of this would be a memory leak.

6.2

The function analyze cannot throw an exception because it is specified as **noexcept**, which means that any attempt to throw an exception results in a fatal error causing the program to end immediately.

The function doWork can throw an exception because it passes analyze a Thing object by value, and thus invokes a copy constructor. This constructor, in turn, could throw an exception, which by technicality counts as doWork throwing an exception.

6.6

a)

The following objects are deleted in the following order, enumerated from first to last:

1. die3
2. die
3. countdown
4. hello
5. i
6. bjarne
7. herb
8. dv
9. u
10. z

b)

The following objects are deleted in the following order, enumerated from first to last:

1. s
2. x

6.8

a)

Functions that access memory are prone to throwing exceptions, so it is dangerous in this case to have the noexcept function which accesses memory not be defined where it is declared. Presumably it is defined in some library, wherein the programmer could easily make a mistake in regards to valid ways to use the pointers it has been passed. It would be much safer to define the function in the same place it is declared. One example of how this could go wrong is if useBuffers passed one of its pointers to another part of the code before returning, since as soon as it returns the memory pointed to by those pointers will be deallocated thus leaving invalid pointers in the code. Another way it could go wrong is if the buffers were deallocated within the useBuffers function, thus causing the memory to be deallocated a second time upon return which is undefined behaviour.

b)

If std::showbase or std::hex throws an exception then the original formatting flags will not be restored. To remedy this, figure out what exceptions these functions can throw (one that I am aware of is std::out_of_range), and put the stream insertion statement into a try block followed by appropriate catch blocks. Then have the line restoring the old flags outside of the catch blocks at the bottom of the function right before it returns so that the old flags are guaranteed to be restored.

c)

The creator of the Queue class should put some mechanism in place to deal with the event where push_back throws an exception. The user of the class might have an unexpected exception get thrown out at them when trying to put an item in the queue. To remedy this, the creator of the class should have put safeguards in to prevent this, such as having the push_back call inside a try block.