COMP710: Studio Session 07 – Exercise:

EXERCISE NAME: *C++ – Detecting and Fixing Memory Leaks*

Beware, you may have added memory leaks to the "GP Framework" as you have developed various features – try to fix these as soon as you notice them.

In your "GP Framework", **`#include <crtdbg.h>`** at the top of the **`main.cpp`** source file.

Next, add the following as the first instruction of the **`main`** function:

```
_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
```

Build (compile and link) and then run your "GP Framework".

Then quit the "GP Framework", and once the program exits, check the "Output" window in Visual Studio. Examine the output, and note if memory leaks are detected. If so, try to determine what Free Store allocations are leaking, and then resolve the issues.

An example of the output after quitting the framework could be:

```
Detected memory leaks!
Dumping objects ->
{176} normal block at 0x00547520, 1 bytes long.
 Data: < > CD
{175} normal block at 0x005474E0, 4 bytes long.
 Data: <    > 00 00 00 00
{174} normal block at 0x00547498, 8 bytes long.
 Data: < sT     > DC 73 54 00 00 00 00 00
{173} normal block at 0x00547428, 48 bytes long.
 Data: <(tT (tT (tT     > 28 74 54 00 28 74 54 00 28 74 54 00 01 01 CD CD
{172} normal block at 0x005473D8, 16 bytes long.
 Data: < '   tT (tT     > E0 27 0F 00 98 74 54 00 28 74 54 00 00 00 00 00
Object dump complete.
The program '[0x1A0C] Framework.exe' has exited with code 0 (0x0).
```

To fix the leaks, ensure each call to **`new`** is paired with an associated call to **`delete`** at some point in the execution of the framework/game. Remember to set the wild pointer to null after deletion. Similarly, any call to **`new []`** must be paired with **`delete[]`**. As you resolve missing **`delete`** and **`delete[]`** calls, recompile and test to check if the number of memory leaks has decreased.