For each of these write what happens when you do the following things. If the step executes without reporting an issue write "No Problem", if there was a problem, explain what the issue was.

Before you do the each of the steps you should output something and then again after you are done. This will let you know if the code ran successfully

You will need the following class:

class MemLabClass {

int \* i;

public:

MemLabClass( ) : i(new int) { std::cout << "MemLabClass Constructor" << std::endl; }

~MemLabClass( ) { std::cout << "MemLabClass Destructor" << std::endl; delete i; }

};

1) dynamically allocate a new integer, delete it twice

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: No problem**

**G++ Run: Crashed**

**DR Memory: UNADDRESSABLE ACCESS, INVALID HEAP ARGUMENT to free, Possible Leak**

2) dynamically allocate a new MemLabClass, delete it twice

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: No problem**

**G++ Run: Crahsed**

**DR Memory: UNADDRESSABLE ACCESS,** **UNADDRESSABLE ACCESS of freed memory,** **INVALID HEAP ARGUMENT to free**

**From the output, number of constructor calls? 1**

**From the output, number of destructor calls? 2**

3) dynamically allocate a new MemLabClass, then free it (do not delete it!)

**VS Compile: No problem**

**VS Run: No problem**

**G++ Compile: No problem**

**G++ Run: No problem**

**DR Memory: UNADDRESSABLE ACCESS,** **INVALID HEAP ARGUMENT,** **LEAK 4 direct bytes**

**From the output, number of constructor calls? 1**

**From the output, number of destructor calls? 0**

4) Allocate an integer pointer, don't delete it

**VS Compile: No problem**

**VS Run: No problem**

**G++ Compile: No problem**

**G++ Run: No problem**

**DR Memory: UNADDRESSABLE ACCESS,** **LEAK 4 direct bytes**

5) Declare an integer pointer, DON'T allocate it, dereference it assign 5 to it.

**VS Compile: Using uninitialized memory, uninitialized local variable**

**VS Run: Can not run it**

5b) Place the following before you function:

**#pragma warning(push)**

**#pragma warning(disable : 4700)**

And place the following after your function:

**#pragma warning(pop)**

for more information see: https://docs.microsoft.com/en-us/cpp/preprocessor/warning?view=vs-2019

warning(push) basically adds a warning/error block. Warning disable removes error 4700.

warning(pop) will remove the top most error block which puts it back to where it was at before the push.

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: No problem**

**G++ Run: Crashed**

**DR Memory: UNADDRESSABLE ACCESS beyond top of stack,** **UNINITIALIZED READ**

Delete the pragmas after this

6) create a static array of 10 integers, free it

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: warning: attempt to free a non-heap object**

**G++ Run: No problem**

**DR Memory: UNADDRESSABLE ACCESS,** **INVALID HEAP ARGUMENT,** **POSSIBLE LEAK 23 direct bytes**

7) create a static array of 10 integers, delete it (standard delete, not array delete)

**VS Compile: w**arning C4154: deletion of an array expression

7b) And place the following after your function:

**#pragma warning(push)**

**#pragma warning(disable : 4154 4156)**

And place the following after your function:

**#pragma warning(pop)**

**VS Compile: No problem**

**VS Run: Exception thrown**

Delete the pragmas after this

7c) Change to an array delete

**VS Compile:** warning C4154: deletion of an array expression

7d) And place the following after your function:

**#pragma warning(push)**

**#pragma warning(disable : 4154)**

And place the following after your function:

**#pragma warning(pop)**

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: warning: deleting array**

**G++ Run: No problem**

**DR Memory: UNADDRESSABLE ACCESS,** **INVALID HEAP ARGUMENT to free,** **POSSIBLE LEAK 23 direct bytes**

8) dynamically allocate new an array of 10 MemLabClass, free it (not delete)

**VS Compile: No problem**

**VS Run: Exception thrown**

**G++ Compile: No problem**

**G++ Run: Crashed**

**DR Memory: UNADDRESSABLE ACCESS, INVALID HEAP ARGUMENT,** **POSSIBLE LEAK 23 direct bytes,** **LEAK 88 direct bytes**

**From the output, number of constructor calls? 10**

**From the output, number of destructor calls? 0**

8b) change the free to a non array delete

**DR Memory: UNADDRESSABLE ACCESS beyond top of stack,** **INVALID HEAP ARGUMENT,** **POSSIBLE LEAK 23 direct bytes, LEAK 88 direct bytes**

**From the output, number of constructor calls? 10**

**From the output, number of destructor calls? 1**

8c) change to an array delete

**DR Memory: UNADDRESSABLE ACCESS beyond top of stack,** **POSSIBLE LEAK 23 direct bytes**

**From the output, number of constructor calls? 10**

**From the output, number of destructor calls? 10**

9) Create a static array of 10 integers, don't initialize them, for(int i = 0; i <= 9; i++)   
output each element along with an endl

**VS Compile: No problem**

**VS Run: Print garbage value 10 times**

**G++ Compile: No problem**

**G++ Run: Print garbage values**

**DR Memory: UNADDRESSABLE ACCESS,** **UNINITIALIZED READ, POSSIBLE LEAK 23 direct bytes**

9b) initialize the array by using = { 0 };

**DR Memory: UNADDRESSABLE ACCESS, POSSIBLE LEAK 23 direct bytes**

**How did this change the output? Print 0 ten times**

9c) initialize the array by using = { 1 };

**DR Memory: UNADDRESSABLE ACCESS, POSSIBLE LEAK 23 direct bytes**

**How did this change the output? 1 0 0 0 0 0 0 0 0 0**

9d) change the for loop to go i <= 10 (now accessing 11 elements)

**VS Compile: No problem**

**VS Run: Print out garbage value when access to 11 element.**

**G++ Compile: No problem**

**G++ Run: Print out value 16 for the 11 element.**

9e) run "make memory" to run dr memory (this result may surprise you)

**DR Memory: UNADDRESSABLE ACCESS,** **UNINITIALIZED READ,** **UNINITIALIZED READ,** **POSSIBLE LEAK 23 direct bytes**

10) Create a dynamic array of 10 integers, don't initialize them, for(int i = 0; i <= 9; i++)   
output each element along with an endl, delete the array properly;

**VS Compile: No problem**

**VS Run: Print 10 times of garbage value**

**G++ Compile: No problem**

**G++ Run: Print out garbage value on first and third elements.**

**DR Memory: UNADDRESSABLE ACCESS, UNINITIALIZED READ,** P**OSSIBLE LEAK 23 direct bytes,** **LEAK 40 direct bytes**

10b) initialize the array by using new int[10]{ 0 };

**VS Compile: No problem**

**VS Run: Print out 0 ten times**

**G++ Compile: No problem**

**G++ Run: Print out 0 ten times**

**DR Memory: UNADDRESSABLE ACCESS,** **POSSIBLE LEAK 23,** **LEAK 40 direct bytes**

**How did this change the output? Originally the compiler printed the garbage value but now print 0 uniformly.**

10c) initialize the array by using new int[10]{ 1 };

**How did this change the output?**

**It also printed garbage value before but now it’s initialized first element to 1 and left elements to 0**

10d) change the for loop to go i <= 10 (now accessing 11 elements)

**VS Compile: No problem**

**VS Run: Print garbage values**

**G++ Compile: No problem**

**G++ Run: Print garbage values on first, third and last elements**

**DR Memory: UNADDRESSABLE ACCESS,** **UNINITIALIZED READ,** **UNADDRESSABLE ACCESS beyond heap bounds,** **POSSIBLE LEAK 23 direct bytes,** **LEAK 40 direct bytes**