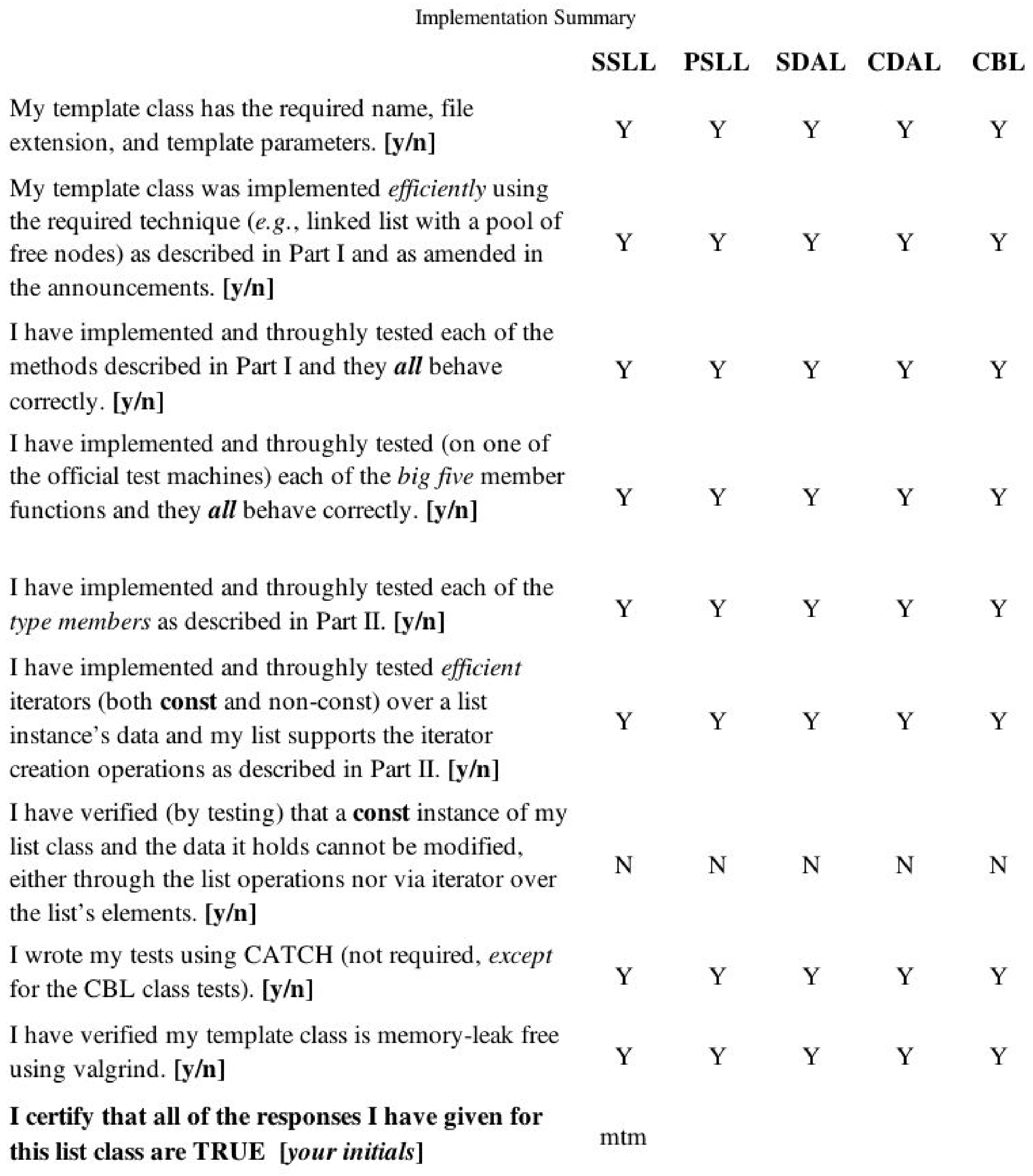
Section I: Checklist



Implementation Summary

	SSLL	PSLL	SDAL	CDAL	CBL
My template class has the required name, file extension, and template parameters. **[y/n]**	Y	Y	Y	Y	Y
My template class was implemented *efficiently* using the required technique (*e.g.*, linked list with a pool of free nodes) as described in Part I and as amended in the announcements. **[y/n]**	Y	Y	Y	Y	Y
I have implemented and throughly tested each of the methods described in Part I and they ***all*** behave correctly. **[y/n]**	Y	Y	Y	Y	Y
I have implemented and throughly tested (on one of the official test machines) each of the *big five* member functions and they ***all*** behave correctly. **[y/n]**	Y	Y	Y	Y	Y
I have implemented and throughly tested each of the *type members* as described in Part II. **[y/n]**	Y	Y	Y	Y	Y
I have implemented and throughly tested *efficient* iterators (both **const** and non-const) over a list instance's data and my list supports the iterator creation operations as described in Part II. **[y/n]**	Y	Y	Y	Y	Y
I have verified (by testing) that a **const** instance of my list class and the data it holds cannot be modified, either through the list operations nor via iterator over the list's elements. **[y/n]**	N	N	N	N	N
I wrote my tests using CATCH (not required, *except* for the CBL class tests). **[y/n]**	Y	Y	Y	Y	Y
I have verified my template class is memory-leak free using valgrind. **[y/n]**	Y	Y	Y	Y	Y
I certify that all of the responses I have given for this list class are TRUE [*your initials*]	mtm				

Section II: Learning Experience

Overall the project, while time consuming, was only moderate in difficulty and threw very little surprises. The easiest part was probably implementing the iterators, because once they were done

one could use them for parts of the internal code (like for example, `contains`). They made sense because it was the same thing as taking the code already written for iteration, and generalize it in a way that would save it's state. The most difficult part was the CDAL, mainly because it at first appeared to be straightforward, but then the challenge came when you had to shift elements after insertion and removal, and had to deal with allocating new nodes, as well as making sure that there is not more than one extra one.

I think that the project's educational objectives were first to solidify the concepts of the linked list, but also to introduce steps into making good quality code (with test cases and documentation), even for simple things. These goals were achieved for the most part, I would say that I know list implementations pretty well now, but as far as the quality code, I may have slipped a bit. Some parts of the code could have used more comments and (in my opinion) declaring methods inline in the class definition is bad practice.

Section III: Testing

The test code for all the lists is the same, the only difference is a typedef that causes the different class to be tested against.

There are nine sections of tests for each list type. The first two test the list's expected values, both with `item_at` and the peek and pop on the front and back. Random numbers are used to determine where to insert, and operations are mirrored on both the cop3530::list and a std::vector. The accuracy of the cop3530::list is determined by what the vector reports. The rand function is seeded with a constant value, so that the tests are all deterministic (as long as they are all performed on the same machine).

The rest of the test code goes through the rest of the expected behaviors, such as `contains` and `contents` and the iterators.

CBL

Compile command

```
g++ ./cbl/tests/main.cpp -std=c++11 -Wall -Wextra -Wpedantic
```

Program output

```
==============================================================================
==
All tests passed (487 assertions in 1 test case)
```

Valgrind output

```
==5754== Memcheck, a memory error detector
==5754== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5754== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright
info
==5754== Command: ./a.out
==5754==

============================================================================
==
All tests passed (487 assertions in 1 test case)

==5754==
==5754== HEAP SUMMARY:
==5754==     in use at exit: 0 bytes in 0 blocks
==5754==   total heap usage: 588 allocs, 588 frees, 122,082 bytes allocated
==5754==
==5754== All heap blocks were freed -- no leaks are possible
==5754==
==5754== For counts of detected and suppressed errors, rerun with: -v
==5754== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

CDAL

Compile command

```
g++ ./cdal/tests/main.cpp -std=c++11 -Wall -Wextra -Wpedantic
```

Program output

```

============================================================================
==
All tests passed (487 assertions in 1 test case)
```

Valgrind output

```
==5781== Memcheck, a memory error detector
==5781== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
```

```
==5781== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright
info
==5781== Command: ./a.out
==5781==

===============================================================================
==
All tests passed (487 assertions in 1 test case)

==5781==
==5781== HEAP SUMMARY:
==5781==     in use at exit: 0 bytes in 0 blocks
==5781==   total heap usage: 714 allocs, 714 frees, 163,370 bytes allocated
==5781==
==5781== All heap blocks were freed -- no leaks are possible
==5781==
==5781== For counts of detected and suppressed errors, rerun with: -v
==5781== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

PSLL

Compile command

```
g++ ./psll/tests/main.cpp -std=c++11 -Wall -Wextra -Wpedantic
```

Program output

```
===============================================================================
==
All tests passed (487 assertions in 1 test case)
```

Valgrind output

```
==5810== Memcheck, a memory error detector
==5810== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5810== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright
info
==5810== Command: ./a.out
```

```
==5810==

===============================================================================
==
All tests passed (487 assertions in 1 test case)

==5810==
==5810== HEAP SUMMARY:
==5810==     in use at exit: 0 bytes in 0 blocks
==5810==   total heap usage: 998 allocs, 998 frees, 124,546 bytes allocated
==5810==
==5810== All heap blocks were freed -- no leaks are possible
==5810==
==5810== For counts of detected and suppressed errors, rerun with: -v
==5810== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

SDAL

Compile command

```
g++ ./sdal/tests/main.cpp -std=c++11 -Wall -Wextra -Wpedantic
```

Program output

```
===============================================================================
==
All tests passed (487 assertions in 1 test case)
```

Valgrind output

```
==5826== Memcheck, a memory error detector
==5826== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5826== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright
info
==5826== Command: ./a.out
==5826==

===============================================================================
==
All tests passed (487 assertions in 1 test case)

==5826==
==5826== HEAP SUMMARY:
==5826==     in use at exit: 0 bytes in 0 blocks
==5826==   total heap usage: 593 allocs, 593 frees, 123,298 bytes allocated
==5826==
==5826== All heap blocks were freed -- no leaks are possible
==5826==
==5826== For counts of detected and suppressed errors, rerun with: -v
==5826== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

SSLL

Compile command

```
g++ ./ssll/tests/main.cpp -std=c++11 -Wall -Wextra -Wpedantic
```

Program output

```
=============================================================================
==
All tests passed (487 assertions in 1 test case)
```

Valgrind output

```
==5840== Memcheck, a memory error detector
==5840== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==5840== Using Valgrind-3.12.0.SVN and LibVEX; rerun with -h for copyright
info
==5840== Command: ./a.out
==5840==

=============================================================================
==
All tests passed (487 assertions in 1 test case)
```

```
==5840==
==5840== HEAP SUMMARY:
==5840==     in use at exit: 0 bytes in 0 blocks
==5840==   total heap usage: 998 allocs, 998 frees, 124,546 bytes allocated
==5840==
==5840== All heap blocks were freed -- no leaks are possible
==5840==
==5840== For counts of detected and suppressed errors, rerun with: -v
==5840== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```