**Homework Four Project Report**

In this homework project, an observation I made was the scope of variables, classes, and methods throughout the helper file and cpp file. This was crucial when defining methods and operators for overloading in the Mystring.cpp files (Mystring::). An example of this scope resolution operator being used is for the non-global method append (Mystring& **Mystring::**append). Something that I understood better from this lab was Class definitions and the uses of constructors in testing.

An extension of scope observations was the use of other methods in implementation. This made it much more efficient to write the code for certain functions like replace which can use the append method in its implementation.

**4.2 default assignment operator testing:**

We get a compilation error as many of the Mystring functions in the header files are not implemented. Because there is no implementation of the operator or methods, the compiler does not know what to do when those methods and operators are in the main.

The main difference between outputs using Mystring and the given c++ string class is the capacity(). The standard string class is based on memory allocation versus Mystring which is written to be len+1. This is also the case for max\_size() which is dependent on memory while our code has it defined in the files.

Test cases are extremely important; thus this homework assignment contained many test cases to cover a wide range of possible outputs from calling constructors, methods, and operators.

**My Test Cases:**

void check (const string s, const string name)

{

cout << "checking " << name << endl;

cout << name << " contains " << s << endl;

cout << name << " capacity() is " << s.capacity() << endl;

cout << name << " length() is " << s.length() << endl;

cout << name << " size() is " << s.size() << endl;

cout << name << " max\_size() is " << s.max\_size() << endl << endl;

}

int main()

{

cout << "This is Homework 4\n";

string s1("Hello, World!");

string s1name("s1");

check (s1, s1name);

cout << "---Testing assignment operator---\n\n";

string s2;

s2=s1; //sets them equal to each other

string s2name("s2");

check(s2,s2name); //compares to s1

check(s1,s1name); //compares to s2

cout << "---Testing append operator---\n\n";

s2+=s1; //appends s1 to s2

check(s2,s2name);

s2+=":-)"; //appends :-) to s2

check(s2,s1name);

cout << "---Testing push\_back---\n\n";

s2.push\_back('H'); //adds 'H' to the end

s2.push\_back('i'); //adds 'i' to the end

s2.push\_back('!'); //adds '!' to the end

check(s2,s2name);

cout << "---Testing append---\n\n";

string s3("ABC");

string s3name("s3");

s1.append(s3); //append s3 to s1

check(s1,s1name);

s1.append("DEF"); //append DEF to s1

check(s1,s1name);

cout << "---Testing insert---\n\n";

s1.insert(5,s3); //inserts s3 at index 5

check(s1,s1name);

s1.insert(5,"DEF"); //inserts DEF at index 4

check(s1,s1name);

cout << "---Testing replace---\n\n";

s1.replace(5,3,s2); //replace 3 characters of s1 at index 5 with s2

check(s1,s1name);

s1.replace(8,3,"lo!l"); //replace 3 characters of s1 at index 8 with lo!l

check(s1,s1name);

cout << "---Testing find\_first\_of---\n\n";

cout << "The first comma is at pos " << s1.find\_first\_of(",",0,25) << "\n"; //finds the first comma

check(s1,s1name);

cout << "The first e is at pos " << s1.find\_first\_of("e",0,25) << "\n"; //finds the first e

check(s1,s1name);

cout << "---Testing last\_not\_of and [] operator---\n\n";

cout << "The last thing not a comma is at pos " << s1.find\_last\_not\_of(",") << "\n";//find the last not ,

cout << s1[s1.find\_last\_not\_of(",")] << endl; //output the value of that last character

check(s1,s1name);

cout << "The last thing not l is at pos " << s1.find\_last\_not\_of("l") << "\n"; //find the last not l

cout << s1[s1.find\_last\_not\_of("l")] << endl; //output the value of that last character

check(s1,s1name);

cout << "---Testing equivalence == ---\n\n";

string s4(s3);

string s4name("s4");

if (s3 == s4)

cout << "s3 == s4 \n";

check(s3,s3name);

check(s4,s4name);

if (s4 == "ABC")

cout << "s4 == ABC\n";

if ("ABC" == s4)

cout << "ABC == s4\n\n";

cout << "---Testing clear and empty---\n\n";

s4.clear();

if (s4.empty())

cout << "s4 is empty\n";

else

cout << "s4 is not empty\n";

check(s4,s4name);

cout << "---Testing nonequivalence != ---\n\n";

if (s3 != s4)

cout << "s3 != s4 \n";

if (s4 != "ABC")

cout << "s4 != ABC\n";

if ("ABC" != s4)

cout << "ABC != s4\n";

check(s3,s3name);

check(s4,s4name);

cout << "---Testing + operator---\n\n";

cout<<"before:\n";

check(s2,s2name);

check(s3,s3name);

check(s4,s4name);

s4 = s2 + s3; //add s2 to s3

cout<<"after:\n";

check(s2,s2name);

check(s3,s3name);

check(s4,s4name);

cout << "---Testing replace special cases---\n\n";

//s5 should be "Hddo"

string s5("Hello");

s5.replace(1, 3, "dd"); //replaces 3 characters of Hello starting at index 1 with string "dd"

check(s5,"s5");

//s6 should be the same as s7 "Dips"

string s6("Dog");

s6.replace(1, 2, "ips"); //replaces 2 characters of Dog starting at index 1 with string "ips"

check(s6,"s6");

string s7("Dog");

s7.replace(1, 3, "ips"); //replaces 3 characters of Dog starting at index 1 with string "ips"

check(s7,"s7");

//s8 should be "Ahobama"

string s8("Ahmed");

s8.replace(2, 5, "obama"); //replaces 5 characters of Ahmed starting at index 2 with string "obama"

check(s8,"s8");

//s9 should be "poapples"

string s9("potato");

s9.replace(2, 5, "apples"); //replaces 5 characters of potato starting at index 2 with string "apples"

check(s9,"s9");

cout << "HW4 end\n";

return 0;

}

**Output for My Test Cases:**

This is Homework 4

checking s1

s1 contains Hello, World!

s1 capacity() is 14

s1 length() is 13

s1 size() is 13

s1 max\_size() is 1073741820

---Testing assignment operator---

checking s2

s2 contains Hello, World!

s2 capacity() is 14

s2 length() is 13

s2 size() is 13

s2 max\_size() is 1073741820

checking s1

s1 contains Hello, World!

s1 capacity() is 14

s1 length() is 13

s1 size() is 13

s1 max\_size() is 1073741820

---Testing append operator---

checking s2

s2 contains Hello, World!Hello, World!

s2 capacity() is 27

s2 length() is 26

s2 size() is 26

s2 max\_size() is 1073741820

checking s1

s1 contains Hello, World!Hello, World!:-)

s1 capacity() is 30

s1 length() is 29

s1 size() is 29

s1 max\_size() is 1073741820

---Testing push\_back---

checking s2

s2 contains Hello, World!Hello, World!:-)Hi!

s2 capacity() is 33

s2 length() is 32

s2 size() is 32

s2 max\_size() is 1073741820

---Testing append---

checking s1

s1 contains Hello, World!ABC

s1 capacity() is 17

s1 length() is 16

s1 size() is 16

s1 max\_size() is 1073741820

checking s1

s1 contains Hello, World!ABCDEF

s1 capacity() is 20

s1 length() is 19

s1 size() is 19

s1 max\_size() is 1073741820

---Testing insert---

checking s1

s1 contains HelloABC, World!ABCDEF

s1 capacity() is 23

s1 length() is 22

s1 size() is 22

s1 max\_size() is 1073741820

checking s1

s1 contains HelloDEFABC, World!ABCDEF

s1 capacity() is 26

s1 length() is 25

s1 size() is 25

s1 max\_size() is 1073741820

---Testing replace---

checking s1

s1 contains HelloHello, World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 55

s1 length() is 54

s1 size() is 54

s1 max\_size() is 1073741820

checking s1

s1 contains HelloHello!l World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 56

s1 length() is 55

s1 size() is 55

s1 max\_size() is 1073741820

---Testing find\_first\_of---

The first comma is at pos 24

checking s1

s1 contains HelloHello!l World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 56

s1 length() is 55

s1 size() is 55

s1 max\_size() is 1073741820

The first e is at pos 1

checking s1

s1 contains HelloHello!l World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 56

s1 length() is 55

s1 size() is 55

s1 max\_size() is 1073741820

---Testing last\_not\_of and [] operator---

The last thing not a comma is at pos 23

o

checking s1

s1 contains HelloHello!l World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 56

s1 length() is 55

s1 size() is 55

s1 max\_size() is 1073741820

The last thing not l is at pos 1

e

checking s1

s1 contains HelloHello!l World!Hello, World!:-)Hi!ABC, World!ABCDEF

s1 capacity() is 56

s1 length() is 55

s1 size() is 55

s1 max\_size() is 1073741820

---Testing equivalence == ---

s3 == s4

checking s3

s3 contains ABC

s3 capacity() is 4

s3 length() is 3

s3 size() is 3

s3 max\_size() is 1073741820

checking s4

s4 contains ABC

s4 capacity() is 4

s4 length() is 3

s4 size() is 3

s4 max\_size() is 1073741820

s4 == ABC

ABC == s4

---Testing clear and empty---

s4 is empty

checking s4

s4 contains

s4 capacity() is 1

s4 length() is 0

s4 size() is 0

s4 max\_size() is 1073741820

---Testing nonequivalence != ---

s3 != s4

s4 != ABC

ABC != s4

checking s3

s3 contains ABC

s3 capacity() is 4

s3 length() is 3

s3 size() is 3

s3 max\_size() is 1073741820

checking s4

s4 contains

s4 capacity() is 1

s4 length() is 0

s4 size() is 0

s4 max\_size() is 1073741820

---Testing + operator---

before:

checking s2

s2 contains Hello, World!Hello, World!:-)Hi!

s2 capacity() is 33

s2 length() is 32

s2 size() is 32

s2 max\_size() is 1073741820

checking s3

s3 contains ABC

s3 capacity() is 4

s3 length() is 3

s3 size() is 3

s3 max\_size() is 1073741820

checking s4

s4 contains

s4 capacity() is 1

s4 length() is 0

s4 size() is 0

s4 max\_size() is 1073741820

after:

checking s2

s2 contains Hello, World!Hello, World!:-)Hi!

s2 capacity() is 33

s2 length() is 32

s2 size() is 32

s2 max\_size() is 1073741820

checking s3

s3 contains ABC

s3 capacity() is 4

s3 length() is 3

s3 size() is 3

s3 max\_size() is 1073741820

checking s4

s4 contains Hello, World!Hello, World!:-)Hi!ABC

s4 capacity() is 36

s4 length() is 35

s4 size() is 35

s4 max\_size() is 1073741820

---Testing replace special cases---

checking s5

s5 contains Hddo

s5 capacity() is 5

s5 length() is 4

s5 size() is 4

s5 max\_size() is 1073741820

checking s6

s6 contains Dips

s6 capacity() is 5

s6 length() is 4

s6 size() is 4

s6 max\_size() is 1073741820

checking s7

s7 contains Dips

s7 capacity() is 5

s7 length() is 4

s7 size() is 4

s7 max\_size() is 1073741820

checking s8

s8 contains Ahobama

s8 capacity() is 8

s8 length() is 7

s8 size() is 7

s8 max\_size() is 1073741820

checking s9

s9 contains poapples

s9 capacity() is 9

s9 length() is 8

s9 size() is 8

s9 max\_size() is 1073741820

HW4 end

**Output for Given Test Cases:**

checking s1

s1 contains Hello, World!

s1 capacity() is 14

s1 length() is 13

s1 size() is 13

s1 max\_size() is 1073741820

---Testing assignment operator and == operator---

checking s2

s2 contains Hello, World!

s2 capacity() is 14

s2 length() is 13

s2 size() is 13

s2 max\_size() is 1073741820

s1 and s2 are same string

checking s3

s3 contains Hi

s3 capacity() is 3

s3 length() is 2

s3 size() is 2

s3 max\_size() is 1073741820

checking s3

s3 contains Hey

s3 capacity() is 4

s3 length() is 3

s3 size() is 3

s3 max\_size() is 1073741820

Hey

assignment operator is correct

--------------------------------------

---Testing array notation---

second char of s1: e

--------------------------------------

---Testing += operator ---

s2 after += s1: Hello, World!Hello, World!

Hey you

--------------------------------------

---Testing append ---

s2 after append s1: Hello, World!Hello, World!??

checking s2

s2 contains Hello, World!Hello, World!??

s2 capacity() is 29

s2 length() is 28

s2 size() is 28

s2 max\_size() is 1073741820

--------------------------------------

---Testing insert ---

s2 after insert: Hello,c++ World!Hello, World!??

--------------------------------------

---Testing replace ---

s2 after replace: Hello,codeld!Hello, World!??

--------------------------------------

---Testing find\_first\_of ---

s2 find first of aeiou after index 2: 4--------------------------------------

---Testing find\_last\_not\_of ---

s1: 13

s1 find find\_last\_not\_of !? after index 2: 11--------------------------------------

---Testing + ---

Hello, World!Hello,codeld!Hello, World!??

--------------------------------------