**Homework 3 – CS60 Linnell**

**Arjun Kohli**

**W1579330**

**Problem 1**

~~53~~ 35

n1

p1

~~72~~ 21

n2

p2

6.3

n4

p4

p3

p1

n1

~~53~~ 35

p2

n2

72

p3

p4

n4

6.3

main

Stack

n4 = n1; // Compiles because type double can hold int value (35 🡪 35.0)

n1 = n4; // Compiles because type int can hold rounded down double value (6.3 🡪 6)

p1 = p4; // Does not compile because we cannot assign an int pointer to a double pointer

p2 = &p1; // Does not compile because we cannot assign an int pointer to an int pointer pointer

p4 = &n1; // Does not compile because we cannot use a double pointer for an int value

&n1 = &n2; // Does not compile because we cannot assign an address to something else (&n1 is a value)

\*p1 = \*p2; // Compiles because we are changing the value of p1 to the value of p2

**Problem 2**

**biguint.cpp (class file):**

#include "biguint.h"

//1

biguint::biguint() {

for (int i = 0; i < CAPACITY; i++) {

data\_[i] = 0;

}

}

//2-3

biguint::biguint(const std::string & s) {

for (size\_t i = 0; i < CAPACITY-1; i++) {

data\_[i] = 0;

}

int j = 0;

for (int i = s.size()-1; i >= 0; i--) {

char c = s[i];

data\_[j] = c - '0';

j++;

}

}

//4

unsigned short biguint::operator [](std::size\_t pos) const {

if (pos > CAPACITY) {

return 0;

}

return data\_[pos];

}

//5

std::ostream& operator <<(std::ostream& out, const biguint& b) {

for (size\_t i = b.CAPACITY-1; i > 0; i--) {

out << b[i-1];

}

return out;

}

//6-7

void biguint::operator += (const biguint & b) {

for (size\_t i = 0; i < CAPACITY; i++) {

data\_[i] += b[i];

if (data\_[i] > 9) {

data\_[i] -= 10;

data\_[i+1]++;

}

}

}

**//Homework 3**

//Problem 2

//8

biguint operator + (const biguint & b1, const biguint & b2) {

biguint bsum;

bsum += b1;

bsum += b2;

return bsum;

}

//9

int biguint::compare(const biguint & b) const {

for (size\_t i = CAPACITY-1; i >= 0; i--) {

if(data\_[i] > b[i]) {

return 1;

} else if (data\_[i] < b[i]) {

return -1;

}

}

return 0;

}

bool operator < (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == -1) {

return true;

}

return false;

}

bool operator <= (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == -1 || b1.compare(b2) == 0) {

return true;

}

return false;

}

bool operator != (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == 1 || b1.compare(b2) == -1) {

return true;

}

return false;

}

bool operator == (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == 0) {

return true;

}

return false;

}

bool operator >= (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == 1 || b1.compare(b2) == 0) {

return true;

}

return false;

}

bool operator > (const biguint & b1, const biguint & b2) {

if (b1.compare(b2) == 1) {

return true;

}

return false;

}

//Problem 3

//10

void biguint::operator -= (const biguint & b) {

for (size\_t i = 0; i < CAPACITY; i++) {

data\_[i] -= b[i];

if (data\_[i] < 0) {

data\_[i] += 10;

data\_[i+1]--;

}

}

}

biguint operator - (const biguint & b1, const biguint & b2) {

biguint bsub;

bsub = b1;

bsub -= b2;

return bsub;

}

//Problem 4

//11

std::string biguint::toString() {

std::string result = "";

char c = ' ';

int j = CAPACITY-1;

while (data\_[j] == 0) {

j--;

}

for (size\_t i = j+1; i > 0; i--) {

c = data\_[i-1] + '0';

result += c;

}

return result;

}

**main.cpp (main file):**

#include "biguint.h"

#include <iostream>

using namespace std;

int main() {

biguint b1("50");

biguint b2("10");

biguint b3 = b1+b2;

cout << b3 << endl;

cout << (b1.compare(b2)) << endl;

cout << (b1<b2) << endl;

cout << (b1<=b2) << endl;

cout << (b1!=b2) << endl;

cout << (b1==b2) << endl;

cout << (b1>=b2) << endl;

cout << (b1>b2) << endl;

b1-=b2;

cout << b1 << endl;

biguint b4 = b1-b2;

cout << b4 << endl;

string s = b1.toString();

cout << s << endl;

}

**Output:**

0000000000000000060

1

0

0

1

0

1

1

0000000000000000040

0000000000000000030

40