CSCI60 Lab 3

Benjamin Xiong

Mike Dong

10/12

1.

```
biguint::biguint(){
	for(int i=0; i<CAPACITY; i++){
		data_[i]=0;
	}
}
```

2. We let the string number -'0', this will convert the string to an int number.

3.

```
biguint::biguint(string s){
	for(int i=0; i<CAPACITY; i++){
		if(i<s.length()){
			data_[i]=s[s.length()-1-i]-'0';
		}
		else{
			data_[i]=0;
		}
	}
}
```

4.

```
unsigned short biguint::operator [](int pos){
        if(pos>CAPACITY){
                return 0;
        }
        return data_[pos];
}
```

Calling a[900] will return 0, since it is out of our CAPACITY.

5.

Ben's method:

We add each corresponding digit together. If the outcome is greater than 10, we % it by 10 to leave only the first digit, and add the next digit by 1. We use an if statement if(i+1<CAPACITY) to make sure the outcome stays in the CAPACITY.

Mike's method:

I overload the += operator as a member function. If the command big1 += big2; it should add big2 to big1. My plan is to add each digit, starting from the least significant digit (the rightmost digit), and carrying over any value greater than 9 to the next digit. Specifically, I first initialized an int variable called carry=0, it is used to determine whether it is greater and equal than 10 or not, 0 means <10, 1 means >=10. Then, I used a for loop from size_t i = 0 to i < CAPACITY, which is used to add every corresponding digit together and over 10 into 1. In the for loop, I defined another int variable: int sum = data_[i] + b.data_[i] + carry; By doing this, I can add the two numbers together and solve the problem of over 10 into 1 each time. To further achieve this, I

made an if else statement in the for loop, if sum is greater than or equal to 10, set carry to 1(over 10 into 1), if sum is less than 10, set carry to 0.

6.

Ben's +=:

```
void biguint::operator +=(biguint b){
	for(int i=0; i<CAPACITY; i++){
		data_[i]+=b[i];
		if(data_[i]>=10){
			data_[i]%=10;
			if(i+1<CAPACITY){
				data_[i+1]++;
			}
		}
	}
}
```

Mike's +=:

```
//Overload the += operator as a member function. The command big1 += big2; should add big2 to big1.
void biguint::operator += (biguint b) {
	//add each digit, starting from the least significant digit (the rightmost digit), and carrying over any value greater than 9 to the next digit.
	int carry = 0;//used to determine whether it is greater equal than 10 or not, 0 means <10, 1 means >=10
```

```
        for (size_t i = 0; i < CAPACITY; i++) {//add every corresponding digits together,
over 10 into 1
                int sum = data_[i] + b.data_[i] + carry;//add the digits and carry together
                if (sum >= 10) {
                        data_[i] = sum - 10;//if sum is greater than or equal to 10
                        carry = 1;//set carry to 1
                }
                else {//if sum is less than 10
                        data_[i] = sum;
                        carry = 0;//set carry to 0
                }
        }
}
```

Full program:

```
#include <iostream>
#include "biguint.h"
using namespace std;

biguint::biguint(){
        for(int i=0; i<CAPACITY; i++){
                data_[i]=0;
        }
```

```
}
biguint::biguint(string s){
	for(int i=0; i<CAPACITY; i++){
		if(i<s.length()){
			data_[i]=s[s.length()-1-i]-'0';
		}
		else{
			data_[i]=0;
		}
	}
}
unsigned short biguint::operator [](int pos){
	if(pos>CAPACITY){
		return 0;
	}
	return data_[pos];
}
//this is Ben's method, Mike's method also works
void biguint::operator +=(biguint b){
	for(int i=0; i<CAPACITY; i++){
		data_[i]+=b[i];
		if(data_[i]>=10){
			data_[i]%=10;
```

```
                        if(i+1<CAPACITY){
                                data_[i+1]++;
                        }
                }
        }
}
int main(){
        biguint test;
        for(int i=0; i<test.CAPACITY; i++){
                cout<<test[i];//20 zeros
        }
        cout<<endl;
        biguint test1("1472");
        for(int j=0; j<test1.CAPACITY; j++){
                cout<<test1[j];//2741 and 16 zeros
        }
        cout<<endl;
        biguint test2("9463");
        test1+=test2;
        for(int x=0; x<test1.CAPACITY; x++){
                cout<<test1[x];//53901 and 15 zeros
        }
        return 0;
```

```
}
```