Cairo university
Faculty of Computers and Information

CS231
Object Oriented Programming
Big real

2023

Dr. Mohamed El-Ramly

m.elramly@fci-cu.edu.eg

Team members

1 - Abdullah saeed abdullah alnami	**20220705**	bdallhsydalnmy@gmail.com
2 – Eslam Sayed Younus Snosy	**20220495**	eslam894408@gmail.com
3 – Youssef Hossam Ahmed Ismail	**20220390**	y.hossam2274@gmail.com

Task 2 (Classes , objects & Operator overloading)

Algorithms of +,-, <, > and isValidReal

isValidReal

```
bool BigReal::isValidReal(string realNumber)
{
    bool one_decimal_dot{false}; // COMMENT(1) flag to catch if more than one dot

    // COMMENT(2) handling 1st digit to equal (number or dot or a sign(+ or -))
    if (realNumber[0] == '+' || realNumber[0] == '-' || realNumber[0] == '.' ||
(realNumber[0] >= '0' && realNumber[0] <= '9')  ) {

      // COMMENT(3) checking number digits after sign bit (if exist)
        for (int i = 1; i < realNumber.size(); i++) {
            if ((realNumber[i] <='9' && realNumber[i] >= '0')) {
                continue;
            } else if (realNumber[i] == '.' && !one_decimal_dot) {
                one_decimal_dot = true;
            } else {
                return false;
            }
        }
        return true;
    }
    return false;
}
```

- COMMENT(1) One decimal dot → is a flag to make sure the no more than one dot in the number.

- COMMENT(2) handling first digit to equal a sign character (+ or -) or (.) or (a number).

- COMMENT(3) loop from second character index 1 in the number to check if the rest of the BigReal contains an ordinary number (0 - 9) or if it is a dot is it only one or more , therefore inValidReal.

Operator+

```
BigReal BigReal::operator+(BigReal &other) {

    BigReal res;
    fill_zeros(other);
    if (sign != other.sign) {
        if (other.sign == '-')
            ninesComplemet(other);
        else
            ninesComplemet(*this);

        res = this->add(other);
        if (this->SIZE() == res.SIZE()) {
            ninesComplemet(res);
            res.sign = '-';
        } else {
            res.digits_r.resize(res.digits_r.size() - 1);
            BigReal tmp;
            tmp.setNum(res.digits_d.empty() ? "+1" : "+.1");
            res = res.add(tmp);
            res.sign = '+';
        }

    } else {
        res = this->add(other);
        res.sign = sign;
    }

    res.remove_zeros();
    return res;
}
```

- COMMENT(1) make object res to store the result .

- COMMENT(2) call fill zeros function to make the two Bigreal the same size .

- COMMENT(3) if the two signs are the same so just call add function and pass the two objects.

- COMMENT(5) if not call ninesComplemnt function and pass the object with (-) sign.

```
BigReal BigReal::add(BigReal other) {

    int maxSize = max(digits_d.size(), other.digits_d.size());
    BigReal res;
    res.digits_r.pop_back();
    int carry = 0;
    for (int i = 0; i < maxSize; ++i) {
        int sum = carry;
        if (i < digits_d.size())
            sum += digits_d[i];
        if (i < other.digits_d.size())
            sum += other.digits_d[i];
        res.digits_d.push_back(sum % 10);
        carry = sum / 10;
    }
    maxSize = max(digits_r.size(), other.digits_r.size());
    for (int i = 0; i < maxSize; ++i) {
        int sum = carry;
        if (i < digits_r.size())
            sum += digits_r[i];
        if (i < other.digits_r.size())
            sum += other.digits_r[i];
        res.digits_r.push_back(sum % 10);
        carry = sum / 10;
    }
    if (carry != 0) res.digits_r.push_back(carry);
    return res;
}
```

- COMMENT(1) we start by summing the decimal digits .

- COMMENT(2) if there is carry from adding the last two digits then keep it to be added to the real digits .

Operator-

```
BigReal BigReal::operator-(BigReal &other) {

      other.sign = (other.sign == '-') ? '+' : '-';

    return *this + other;

}
```

- COMMENT(1) change the sign of the bigreal that we want to subtract it.
- COMMENT(1) return the result of adding it which well call the + operator.

Operator<

```
BigReal BigReal::compare_two_values(BigReal num1, BigReal num2)
{
    if (num1.digits_r.size() < num2.digits_r.size())
        return num1;
    else if (num1.digits_r.size() > num2.digits_r.size())
        return num2;

    if (num1.digits_d.size() < num2.digits_d.size())
        return num1;
    else if (num1.digits_d.size() > num2.digits_d.size())
        return num2;

    for (int i = num1.digits_r.size()-1 ; i >= 0 ;--i)
        if (num1.digits_r[i] > num2.digits_r[i])
            return num2;

    for (int i = num1.digits_d.size()-1 ; i >= 0 ;--i)
        if (num1.digits_d[i] > num2.digits_d[i])
            return num2;

    return num1;
}
```

```
bool BigReal::operator<(BigReal anotherReal)
{
    if (sign == '+' && anotherReal.sign == '-')
        return false;
    else if (sign == '-' && anotherReal.sign == '+')
        return true;

    if (*this == anotherReal)
        return false;

    if (compare_two_values(*this, anotherReal) == *this)   // the value of *this
is smallest
    {
        if (sign == '+')
            return true;
        else
            return false;
    }
    else   // the value of anotherReal is smallest
    {
        if (sign == '+')
            return false;
        else
            return true;
    }

    return true;
}
```

- COMMENT(1) if two sign is different, than return true if the first number has the sign(-) .

- COMMENT(2) I made a utility function(compare two values) to return a Bigreal that has a smallest value(not care about a sign).

- COMMENT(3) if two sign is (+,+), then return true if the first num has a smallest value compared with the second num.

- COMMENT(5) if two sign is (-,-), then return true if the first num has a Highest value compared with the second num.

Operator>

```
bool BigReal::operator>(BigReal anotherReal)
{
    if (*this < anotherReal || *this == anotherReal)
        return false;
    return true;
}
```

- COMMENT(1) if !(num1 < num2 || num1 == num2), then num1 > num2.

Table shows team work

Task	Eslam Sayed	Youssuf Hossam	Abdullah saeed
isValidReal function	✓		
Default constructor	✓		
Initialize from string	✓		
Copy constructor	✓		
Assignment operator		✓	
Int size() , int sign()		✓	
Void setNum		✓	
Operator+			✓
Operator-			✓
Operator<		✓	
Operator>		✓	
Operator==		✓	
Operator <<		✓	
Project documentation	✓		

Github team workflow

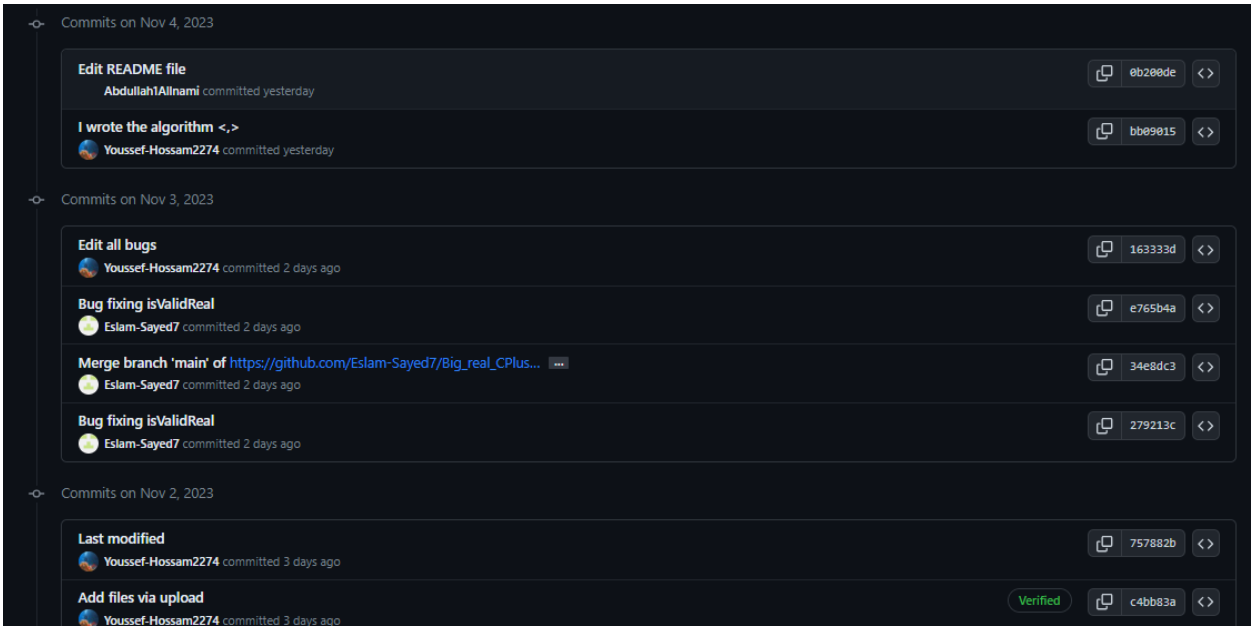

Figure 1 Github team workflow (commits)y