**Lab 08**

**Object Oriented Programming Lab**

**Common Solution**

**14 Marks**

**Challenge-1:***Big Number*

**BigNumber.h**

#ifndef BIG\_NUMBER\_H

#define BIG\_NUMBER\_H

#include<iostream>

using namespace std;

classBigNumber

{

char\* number;

intnumberLength;

intgetStrLength(constchar\* str);

voidcopyStr(constchar\* source, char\* destination);

intcompare(constBigNumber&ref)const;

public:

BigNumber(constchar\*);

BigNumber(constBigNumber&);

~BigNumber();

voidsetNumber(constchar\* num);

voidprint() const;

BigNumberoperator = (constBigNumber&);

BigNumberoperator + (constBigNumber&) const;

BigNumberoperator - (constBigNumber&other) const;

BigNumberoperator \* (constBigNumber&other) const;

booloperator == (constBigNumber&other) const;

booloperator > (constBigNumber&other) const;

booloperator < (constBigNumber&other) const;

booloperator >= (constBigNumber&other) const;

booloperator <= (constBigNumber&other) const;

booloperator != (constBigNumber&other) const;

};

#endif // !BIG\_NUMBER\_H

**BigNumber.cpp**

#include "BigNumber.h"

//Private Functions:

intBigNumber::getStrLength(constchar\* str)

{

int length = 0;

while (str[length] != '\0')

{

length++;

}

return length;

}

voidBigNumber::copyStr(constchar\* source, char\* destination)

{

if (source == nullptr)

{

return;

}

inti = 0;

while (source[i] != '\0')

{

destination[i] = source[i];

i++;

}

destination[i] = '\0';

}

intBigNumber::compare(constBigNumber&other) const

{

if (numberLength<other.numberLength)

return 3;

elseif (numberLength>other.numberLength)

return 2;

else

{

for (inti = 0; i<numberLength; ++i)

{

if (number[i] <other.number[i])

return 3;

elseif (number[i] >other.number[i])

return 2;

}

return 1;

}

}

// Public Functions:

BigNumber::BigNumber(constchar\* input)(-0.5 if not created)

{

if (input == nullptr||input=="")

{

number = newchar[1];

number[0] = '\0';

numberLength = 0;

return;

}

numberLength = getStrLength(input);

number = newchar[numberLength + 1];

copyStr(input, number);

}

BigNumber::BigNumber(constBigNumber&ref)(-0.5 if not created)

{

if (ref.number == nullptr || ref.number == "")

{

number = newchar;

\*number = '\0';

numberLength = 0;

return;

}

numberLength = ref.numberLength;

number = newchar[numberLength + 1];

copyStr(ref.number, number);

}

BigNumber::~BigNumber()(-0.5 if not created)

{

if(number)

delete[] number;

number = nullptr;

numberLength = 0;

}

voidBigNumber::setNumber(constchar\* num)(-0.5 if not created)

{

delete[] number;

numberLength = getStrLength(num);

number = newchar[numberLength + 1];

copyStr(num, number);

}

voidBigNumber::print() const(-0.5 if not created)

{

if (numberLength == 0)

cout<<"0";

else

cout<< number;

}

BigNumberBigNumber::operator=(constBigNumber&other)----(2.5)

{

if (this == &other)

{

return \*this;

}

this->~BigNumber();

if (other.number == nullptr)

return \*this;

numberLength = other.numberLength;

number = newchar[numberLength + 1];

copyStr(other.number, number);

return \*this;

}

Self-assignment check ---- (0.5)

Calling object destructor ---- (0.5)

Nullptr check ---- (0.5)

Copy string correctly ---- (1)

Atomicity (copyStr) -> (-0.5)

BigNumberBigNumber::operator+(constBigNumber&other) const---- (0.5)

{

intmaxLength = (numberLength>other.numberLength) ? numberLength : other.numberLength;

char\* result = newchar[maxLength + 1];

result[maxLength] = '\0';

int carry = 0;

inti = numberLength - 1;

int j = other.numberLength - 1;

int k = maxLength - 1;

while (i>= 0 || j >= 0)

{

int digit1 = (i>= 0) ? (number[i] - '0') : 0;

int digit2 = (j >= 0) ? (other.number[j] - '0') : 0;

int sum = digit1 + digit2 + carry;

carry = sum / 10;

result[k--] = (sum % 10) + '0';

i--;

j--;

}

if (carry > 0)

result[k] = carry + '0';

BigNumbersumNumber(result);

delete[] result;

returnsumNumber;

}

Sample Run:

1. 123456 + 7890 = 131,346 ---- (0.5)

Not deleting array on heap -> (-0.25)

Atomicity -> (-0.25)

BigNumberBigNumber::operator-(constBigNumber&other) const---- (1.5)

{

if (compare(other)==1||compare(other)==3)

returnBigNumber("0");

int\* temp = newint[numberLength + 1];

for (inti = 0; i<numberLength; i++)

{

temp[i] = number[i] - '0';

}

char\* result = newchar[numberLength+1];

inti = numberLength-1, j = other.numberLength-1;

while (j >= 0) //till small no length

{

if (temp[i] <other.number[j] - '0')

{

int k = i - 1;

while (temp[k] == 0)

{

k--;

}

while (k <i) //borrowing 1 uptoi

{

temp[k]--;

temp[k + 1] += 10;

k++;

}

}

result[i] = temp[i] - (other.number[j] - '0') + '0';

j--;

i--;

}

while (i>= 0) //till large no

{

result[i] = temp[i] + '0';

i--;

}

result[numberLength] = '\0';

int z= 0;

while (result[z] == '0') //to exclude starting zeroes

{

z++;

}

BigNumbera{ result +z};

delete[]result;

return a;

}

Sample Runs:

1. 8566 - 929 = 7637 ---- (0.5)
2. 1234567890 - 987654321 = 246913569 ---- (1)

Atomicity ->(-0.25)

Not Deleting array on heap -> (-0.25)

BigNumberBigNumber::operator\*(constBigNumber&other) const---- (2.5)

{

if (other.numberLength == 0)

return \*this;

if (numberLength == 0)

returnother;

intresultLength = numberLength + other.numberLength;

char\* result = newchar[resultLength + 1];

result[resultLength] = '\0';

for (inti = 0; i<resultLength; i++)

{

result[i] = '0';

}

for (inti = numberLength - 1; i>= 0; i--)

{

int carry = 0;

for (int j = other.numberLength - 1; j >= 0; j--)

{

int digit1 = number[i] - '0';

int digit2 = other.number[j] - '0';

int product = digit1 \* digit2 + carry + (result[i + j + 1] - '0');

result[i + j + 1] = (product % 10) + '0';

carry = product / 10;

}

result[i] += carry;

}

intstartPos = 0;

while (result[startPos] == '0'&&startPos<resultLength - 1)

{

startPos++;

}

BigNumberresultBigNum(result + startPos);

delete[] result;

returnresultBigNum;

}

Sample Runs:

1. 450 \* 3 = 1350 ---- (1)
2. 987654 \* 123456 = 121931812224 ---- (1)
3. 12345 \* 0 = 0 ---- (0.5)

Not deleting array on heap -> (-0.25)

Atomicity -> (-0.5)

boolBigNumber::operator==(constBigNumber&other) const---- (1.25)

{

return compare(other) == 1;

}

Sample Runs:

1. cout<< 99==99 //prints 1 ----(0.5)
2. cout<< 99==90 //prints 0 ----(0.75)

Atomicity -> (-0.5)

boolBigNumber::operator>(constBigNumber&other) const---- (1)

{

return compare(other) == 2;

}

Sample Runs:

1. cout<<99>9//prints 1 ----(0.5)
2. cout<<99>100 //prints 0 ----(0.5)

Atomicity -> (-0.5)

boolBigNumber::operator<(constBigNumber&other) const---- (1)

{

return compare(other) == 3;

}

Sample Runs:

1. cout<< 99<9 //prints 0 ----(0.5)
2. cout<< 99<100 //prints 1 ----(0.5)

Atomicity -> (-0.5)

boolBigNumber::operator>=(constBigNumber&other) const---- (1.5)

{

int result = compare(other);

return result == 1 || result == 2;

}

Sample Runs:

1. cout<< 99>= 9 //prints 1 ----(0.5)
2. cout<< 99>= 100 //prints 0 ----(0.5)
3. cout<< 99 >= 99 //prints 1 ----(0.5)

Atomicity -> (-0.5)

boolBigNumber::operator<=(constBigNumber&other) const---- (1.5)

{

int result = compare(other);

return result == 1 || result == 3;

}

Sample Runs:

1. cout<< 99 <=9 //prints 0 ----(0.5)
2. cout<< 99 <= 100 //prints 1 ----(0.5)
3. cout<< 99 <= 99 //prints 1 ----(0.5)

Atomicity -> (-0.5)

boolBigNumber::operator!=(constBigNumber&other) const---- (0.75)

{

return compare(other) != 1;

}

Sample Runs:

1. cout<<99 !=99 //prints 0 ----(0.5)
2. cout<<99 !=90 //prints 1 ----(0.25)

Atomicity -> (-0.5)

**Quick Revision:**

BigNumberoperator=(constBigNumber&other)----(2.5)

Self-assignment check ---- (0.5)

Calling object’s destructor ---- (0.5)

Nullptr check ---- (0.5)

Copy string correctly ---- (1)

Atomicity (copyStr) -> (-0.5)

BigNumberoperator+(constBigNumber&other) const---- (0.5)

Sample Run:

1. 123456 + 7890 = 131,346 ---- (0.5)

Not deleting array on heap -> (-0.25)

Atomicity -> (-0.25)

BigNumberoperator-(constBigNumber&other) const---- (1.5)

Sample Runs:

1. 8566 - 929 = 7637 ---- (0.5)
2. 1234567890 - 987654321 = 246913569 ---- (1)

Atomicity -> (-0.25)

Not Deleting array on heap -> (-0.25)

BigNumberoperator\*(constBigNumber&other) const---- (2.5)

Sample Runs:

1. 450 \* 3 = 1350 ---- (1)
2. 987654 \* 123456 = 121931812224 ---- (1)
3. 12345 \* 0 = 0 ---- (0.5)

Not deleting array on heap -> (-0.25)

Atomicity -> (-0.25)

booloperator==(constBigNumber&other) const---- (1.25)

Sample Runs:

1. cout<< 99==99 //prints 1 ----(0.5)
2. cout<< 99==90 //prints 0 ----(0.75)

Atomicity -> (-0.5)

booloperator>(constBigNumber&other) const---- (1)

Sample Runs:

1. cout<< 99 > 9 //prints 1 ----(0.5)
2. cout<< 99 > 100 //prints 0 ----(0.5)

Atomicity -> (-0.5)

booloperator<(constBigNumber&other) const---- (1)

Sample Runs:

1. cout<< 99 < 9 //prints 0 ----(0.5)
2. cout<< 99 < 100 //prints 1 ----(0.5)

Atomicity -> (-0.5)

booloperator>=(constBigNumber&other) const---- (1.5)

Sample Runs:

1. cout<< 99 >= 9 //prints 1 ----(0.5)
2. cout<< 99 >= 100 //prints 0 ----(0.5)
3. cout<< 99 >= 99 //prints 1 ----(0.5)

Atomicity -> (-0.5)

booloperator<=(constBigNumber&other) const---- (1.5)

Sample Runs:

1. cout<< 99 <=9 //prints 0 ----(0.5)
2. cout<< 99 <= 100 //prints 1 ----(0.5)
3. cout<< 99 <= 99 //prints 1 ----(0.5)

Atomicity -> (-0.5)

booloperator!=(constBigNumber&other) const---- (0.75)

Sample Runs:

1. cout<<99 != 99 //prints 0 ----(0.5)
2. cout<<99 != 90 //prints 1 ----(0.25)

Atomicity -> (-0.5)

Penalties: If functions of previous lab are not made then

---- (-0.5) for each function.

**Penalty Matrix:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Penalty List | Labs | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 3 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| Indentation putting { Infront of loop header, in do while, putting while with closing } | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Meaningful Variable Names |  | -2 | -2 | -2 | -2 | -2 | -2 |  |  |  |  |  |  |  |  |  |
| Camel Case Notation | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Atomicity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Syntax error | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Linker error | 0 | 0 | 0 | 0 | 0 | 0 | - |  |  |  |  |  |  |  |  |  |
| Wrong function prototypes | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Class interface or additional members |  |  | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Use of library function/class without permission | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Continue statement | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| cin/cout where it isn’t needed | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Multi-filing |  |  | 0 | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |
| Wrong #ifndef or name of header file |  |  | -2 | -2 | -2 | -2 | -2 |  |  |  |  |  |  |  |  |  |
| Global functions |  |  | -3 | -3 | -3 | -3 | -3 |  |  |  |  |  |  |  |  |  |
| Multiple classes in one header file |  |  | -3 | -3 | -3 | -3 | -50% |  |  |  |  |  |  |  |  |  |