**Lab 04**

**Object Oriented Programming Lab**

**Common Solution**

**16 Marks**

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

**BigNumber.h**

#ifndef BIG\_NUMBER\_H

#define BIG\_NUMBER\_H

#include<iostream>

using namespace std;

enum Comparison

{

EQUAL,SMALL,LARGE

};

class BigNumber

{

char\* number;

int numberLength;

int getStartingZeros(const char \*);

int getStrLength(const char\*);

void copyStr(const char\* , char\* );

public:

BigNumber(const char \*);

BigNumber(const BigNumber &);

~BigNumber();

BigNumber add(BigNumber);

void print();

Comparison compare(BigNumber);

};

#endif // !BIG\_NUMBER\_H

**BigNumber.cpp**

#include<iostream>

//Private Functions:

int BigNumber::getStartingZeros(const char\* number)

{

int startingZeroes = 0;

while (number[startingZeroes] == '0') //for calculating zeroes at start

startingZeroes++;

return startingZeroes;

}

void BigNumber::copyStr(const char\* src, char\* dest)

{

if (src == nullptr)

{

return;

}

int i = 0;

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

{

dest[i] = src[i];

i++;

}

dest[i] = '\0';

}

int BigNumber::getLength(const char\* ch)

{

int i = 0;

if (ch == nullptr)

{

return 0;

}

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

{

i = i + 1;

}

return i;

}

//Public Functions:

BigNumber::BigNumber(const char\* input) ----- (1.5)

{

if (input == nullptr || getLength(input) == 0) ----- (0.5) for nullptr and empty

{

number = new char[2];

number[0] = '0';

number[1] = '\0';

numberLength = 1;

}

else

{

numberLength = getLength(input);

number = new char[numberLength + 1];

copyStr(input, number); ----- (1) for making deep copy

}

}

for nullptr and empty ---- (0.5)

for making deep copy ---- (1)

Atomicity for length and copyStr ----- (-1)

BigNumber::BigNumber(const BigNumber& ref) ----- (2)

{

numberLength = ref.numberLength;

number = new char[numberLength + 1];

copyStr(ref.number, number);

}

Must make deep copy.

Atomicity --- (-0.5)

BigNumber::~BigNumber() ----- (1)

{

delete[] number;

number = nullptr;

numberLength = 0;

}

void BigNumber::print() ----- (0.5)

{

cout << number;

}

BigNumber BigNumber::add(BigNumber other) ----- (7)

{

int largeLength = (numberLength >= other.numberLength) ? numberLength : other.numberLength; // finding large length

char\* result = new char[largeLength + 2]; // 1 extra for potential carry

result[largeLength + 1] = '\0';

int carry = 0;

int k = largeLength;

for (int i = numberLength - 1, j = other.numberLength - 1; i >= 0 || j >= 0; i--, j--, k--)

{

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

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

int res = digit1 + digit2 + carry;

carry = res / 10;

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

}

// handling last carry

if (carry)

{

int startingZeros = getStartingZeros(result);

result[startingZeros] = '1';

BigNumber sum(result + startingZeros);

delete[] result;

return sum;

}

else

{

//if last carry is 0, than 0 index not copied

int startingZeros = getStartingZeros(result);

BigNumber sum(result + startingZeros + 1);

delete[] result; --- (1)

return sum;

}

}

Sample Runs:

999 + 999 = 1998 ---- (2)

41634 + “” = 41634 ---- (2)

23 + 23 = 46 ---- (1)

990 + 10 = 1000 ---- (1)

1000 + 1 = 1001 ---- (1)

Logic:

The array on heap must be deleted --- (-1)

Comparison BigNumber::compare(BigNumber ref) ----- (4)

{

// handling length if zeroes are at start

int startingZeroes = getStartingZeros(number);

int startingZeroes2 = getStartingZeros(ref.number);

if (ref.numberLength - startingZeroes2 < numberLength - startingZeroes)

return LARGE;

else if (ref.numberLength - startingZeroes2 > numberLength - startingZeroes)

return SMALL;

while (startingZeroes < numberLength && startingZeroes2 < ref.numberLength)

{

if (ref.number[startingZeroes2] > number[startingZeroes])

return SMALL;

else if (ref.number[startingZeroes2] < number[startingZeroes])

return LARGE;

startingZeroes++;

startingZeroes2++;

}

return EQUAL;

}

Sample Runs:

12345 and 12354 answer: 12354 ---- (1.5)

23 and 48 answer: 48 ---- (0.5)

90 and 68 answer: 90 ---- (1.5)

150 and 1505 answer 1505 is ---- (0.5)

**Quick Revision:**

BigNumber(const char \*); ---- (1.5)

for nullptr and empty ---- (0.5)

for making deep copy ---- (1)

Atomicity for length and copyStr ----- (-1)

BigNumber(const BigNumber &); ---- (2)

Must make deep copy.

Atomicity --- (-0.5)

~BigNumber(); ---- (1)

void print(); ---- (0.5)

BigNumber add(BigNumber); ---- (7)

Sample Runs:

999 + 999 = 1998 ---- (2)

41634 + “” = 41634 ---- (2)

23 + 23 = 46 ---- (1)

990 + 10 = 1000 ---- (1)

1000 + 1 = 1001 ---- (1)

Logic:

The array on heap must be deleted --- (-1)

Atomicity --- (-1)

Comparison compare(BigNumber); ---- (4)

Sample Runs:

12345 and 12354 answer: 12354 ---- (1.5)

23 and 48 answer: 48 ---- (0.5)

90 and 68 answer: 90 ---- (1.5)

150 and 1505 answer 1505 is ---- (0.5)

Atomicity --- (-1)

**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 |  |  |  |  |  |  |  |  |  |  |  |  |
| Meaningful Variable Names |  | -2 | -2 | -2 |  |  |  |  |  |  |  |  |  |  |  |  |
| Camel Case Notation | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Atomicity |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Syntax error | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Linker error | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong function prototypes | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Class interface or additional members |  |  | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Use of library function/class without permission | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Continue statement | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| cin/cout where it isn’t needed | 0 | 0 | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Multi-filing |  |  | 0 | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Wrong #ifndef or name of header file |  |  | -2 | -2 |  |  |  |  |  |  |  |  |  |  |  |  |
| Global functions |  |  | -3 | -3 |  |  |  |  |  |  |  |  |  |  |  |  |
| Multiple classes in one header file |  |  | -3 | -3 |  |  |  |  |  |  |  |  |  |  |  |  |