20/25

NATIONAL UNIVERSITY
OF COMPUTER & EMERGING SCIENCES
LAHORE

FAST

Course Name & Section Object Oriented Programming Date 25-3-24

Student's Name Faheem Sarwar Roll No. 23L-3010 Signature

Quiz B

// Methods defined in this class :)

```
class ComplexNumberArray {
Private:
    double* real;
    double* imaginary;
    int size;
    static int totalCount;

Public:
    ComplexNumberArray() {
        real = nullptr;
        imaginary = nullptr;
        size = Null;
        totalCount ++;
    }
    ComplexNumberArray(double* r = nullptr, double* i = nullptr,
    int s = 0){
        real = r
        imaginary = i
        size = s;
        totalcount ++;
    }
    double* getreal(){
        return this->real;
    }
    double* getimginary(){
        return this->imginary;
    }
    int getsize(){
        return this->size;
    }
    static int gettotalcount(){
        return totalcount;
    }
```

Q / Part No.

Rough Work

(5)

```
~ComplexNumberArray(){
    delete[] real;
    delete[] imaginary;
    totalcount--;
}
```

(6)

```
ComplexNumberArray(const ComplexNumber& that){
    this->real = new double[that.size];
    this->imaginary = new double[that.size];
    for(int i=0; i<that.size; i++){
        this->real[i] = that.real[i];
        this->imaginary[i] = that.imaginary[i]; }
    this->size = that.size;
    totalcount++;
}
```

(7)

```
ComplexNumberArray operator=(const ComplexNumberArray& that){
    this->real = that.real;
    this->imaginary = that.imaginary;
    this->size = that.size;
    totalCount++;
}
```

memory leakage

(8)

```
void ComplexNumberArray operator+=(const ComplexNumberArray& a){
    if(this->size != a.size){
        cout<<"error, cannot add, size not same"<<endl;
        ComplexNumberArray A;
        return A; }
    else{
        ComplexNumberArray A;
        A.size = a.size;
        A.real = new double[a.size];
        A.imaginary = new double[a.size]
        for(int i=0; i<size; i++){
            this->A.real[i] = this->real[i] + a.real[i];
            this->A.imaginary[i] = this->imaginary[i] + a.imaginary[i];
        }
        total count++;
        return A;
    }
```

Q / Part No.

Rough Work

(9) ✓

```
~~Complex~~ void operator += (const ComplexNumberArray & c1){
    ~~this~~
    if (this->size != c1.size){
        cout << "error, size not equal" << endl;
    }
    else{
        for (int i=0; i<this->size; ~~i~~ j++){
            this->real[i] += c1.real[i];
            this->imaginary[i] += c1.imaginary[i];
        }
    }
```

(10) ✓

```
double* operator [] (int index){
    double *A = new double [2];
    A[0] = this->real[index];
    A[1] = this->imaginary[index];
    //returns individual complex Number in Array)
    return A;
}
```

(11) ✓

```
ComplexNumberArray & conjugate (){
    ComplexNumberArray A;
    A.size = this->size;
    ~~for (int i=0~~
    A.real = new double [this->size];
    A.imaginary = new double [this->size];
    for (int i=0; i<size; i++){
        ~~this~~ A.real[i] = this.real[i];
        A.imaginary[i] = this.imaginary[i] * -1;
    }
    ~~re total count++;~~
    Return A;
```

(12) ✓

```
Void print () ~~con~~{
    ~~cout <<~~
    for (int i=0; i<size; i++){
        cout << this->real[i] << " " << this.imaginary[i]
             << "i" << ;
    }
```