# Read these Instructions before start the assignment

## Write all your codes within the main.cpp provided with the assignment. Marks for Question 1, 7 and 8 will be given based on your code implementation.

## [Main function body of your code will be automatically generated and tested with random values for complex numbers for Question 1, 7 and 8. Please make sure program is compiling before submission]

## Specify your index number correctly in index\_number hash define in main.cpp

#define index\_number 160299D

## Rename main.cpp file to main\_<Index Number>.cpp where <Index Number> is your index number.

## Submit your code (main\_<Index Number>.cpp) for Question 1, 7 and 8.

## For other questions, submit this document with answers.

## Be honest. Please do not copy. Whole purpose of this assignment is to kickoff your interest about C++

## Duration – 2 hours

## Compiler conforming C++14 will be required

1. Implement member functions correctly for following ComplexNumber class **(20 marks)**

class ComplexNumber

{

public:

ComplexNumber(const float& real, const float& complex);//Constructor

~ComplexNumber();//Destructor

//Getters

float GetReal() const; //Get method for Real part

float GetComplex() const; //Get method for Complex part

//Setters

void SetReal(const float& real);//Set Real part

void SetComplex(const float& complex);//Set Complex part

//Operations

void Add(const ComplexNumber& number);//Add cn to this

bool Equal(const ComplexNumber& number) const;//Check for equality

private:

float m\_real;

float m\_complex;

};

1. What is the purpose of having ‘private:’ keyword in class declaration? **(5 marks)**

**The private members cannot be accessed outside the class and they can be accessed only through methods of the class. We use to hide internal data and encapsulation**

1. Both Constructor parameters are specified with ‘&’ qualifier. Select correct statements **(5 marks)**
   1. Const variables always has to specified with ‘&’ qualifier
   2. Parameters are passed by reference
   3. Parameters are memory addresses of float variables
   4. Parameters are passed by value unless ‘&’ qualifier is specified.

**Answer- b,d**

1. Both Constructor parameters are specified with ‘const’ qualifier. Select correct statements **(5 marks)**
   1. Only constant variables accepted by constructor
   2. Above constructor doesn’t change parameter values
   3. Constructor parameters should always specified with ‘const’ qualifier
   4. Above constructor only accept hard coded values

**Answer- b**

1. Specify benefits of passing a parameter by reference, compared to passing parameter by value in the context of C++ performance oriented coding? **(5 marks)**

**If we pass the reference of an argument in the calling function to the corresponding formal parameter of the called function, it can modify the value of the argument by using its reference passed in.**

**Directly accessing memory address using reference is faster than passing variables and searching for address.**

1. Have a closer look at below code

int main()

{

auto num1 = ComplexNumber(2.0f,3.0f);

auto num2 = new ComplexNumber(2.0f, 3.0f);

auto num3 = &num2;

auto num4 = std::make\_shared<ComplexNumber>(2.0f, 3.0f);

auto num5 = std::make\_unique<ComplexNumber>(2.0f, 3.0f);

return 0;

}

* 1. Specify data type of each variable **(10 marks)**
  2. num1 is **Complex number**
  3. num2 is **Complex number**
  4. num3 is **pointer**
  5. num4 is **pointer**
  6. num5 is **pointer**
  7. Specify the region of memory (stack or heap) used to allocate memory for each variable **(10 marks)**

1. num1 is allocated in **stack**
2. num2 is allocated in **heap**
3. num3 is allocated in **stack**
4. num4 is allocated in **stack**
5. num5 is allocated in **heap**
   1. Which variables cause memory leaks? **(10 marks)**

**Pointer variables(num3 and num4)**

1. Refer below code segment. Add the declaration of member function into ComplexNumber class which makes ‘ComplexNumber e = a + b’ statement compliable. Implement it properly **(10 marks)**

int main()

{

ComplexNumber a (3.0f, 4.0f);

ComplexNumber b (2.0f, 2.0f);

ComplexNumber e = a + b;

return 0;

}

1. Refer below code segment. Add the declaration of member function into ComplexNumber class which makes ‘a == b’ statement compliable. Implement it properly to compare two given complex numbers. **(10 marks)**

int main()

{

ComplexNumber a (2.0f, 3.0f);

ComplexNumber b (2.0f, 3.0f);

if (a == b)

{

std::cout<< “a and b are equal”<< std::endl;

}

return 0;

}

1. Once question 7 is implemented, have a closer look at below code which is not compiling. Specify fixes need to be done. (**Note:** you are only allowed to change the body of the main function) **(10 marks)**

int main()

{

auto c = std::make\_shared<ComplexNumber>(2.0f, 3.0f);

auto d = std::make\_unique<ComplexNumber>(2.0f, 3.0f);

auto e = c + d;

return 0;

}

int main()

{

auto c = std::make\_shared<ComplexNumber>(2.0f, 3.0f);

auto d = std::make\_unique<ComplexNumber>(2.0f, 3.0f);

auto e = \*c + \*d;

return 0;

}

#token number 63,36