UNIVERSITY OF RUHUNA

Faculty of Engineering

End-Semester 2, Examination in Engineering, February 2023

Module No: EE2201 Module Name: Computer Programming II

[2 Hours]

[Answer all questions. All questions carry equal marks]

Part II

Q1. Declaration of the class cList as it appears in the include file is given in Listing 1, and described in Table Q1. The class cList is designed to store a list of 10 numbers, indexed from 1 to 10. Definition (implementation) of constructors as they appear in the file cList.cpp is given in Listing 2 in page 3.

Table Q1: Description of cList members

Method	Description
cList()	Constructor. Creates Empty List
cList(**double** argx[ ])	Constructor. Creates List initialized with argx[ ]
bool InsertAt(**int** index, **double** argx)	insert number argx at given index
double FindAt(**int** index)	Returns the number at given index
bool Find(**double** argx)	Returns true if given number argx exists in the list
int FindIndex(**double** argx)	Gets the index of a given number argx
cList Reverse()	Reverses the list and returns it
double x[N]	Store data of the list, where N=10

Implement the following member methods in C++.

a) **bool** InsertAt(**int** index, **double** argx) [2 Marks]

b) **bool** Find(**double** argx) [2 Marks]

c) **int** FindIndex(**double** argx) [2 Marks]

d) cList Reverse() [2 Marks]

Q2. Declaration of the class cComplex as it appears in the include file cComplex.h is given in Listing 3, and described in Table Q2. The class cComplex is designed to represent a complex number of standard form $Z = Re + iIm$, where Re and Im are real numbers. Definition) of constructors as they appear in the file cComplex.cpp is given in Listing 4.

a) Consider the definition of overloaded assignment operator of the class cComplex given in Listing 5. Explain the purpose of statement **return** ***this**; by using an appropriate example. [2 Marks]

Table Q2: Description of cComplex members

Method	Description
double Re	Real part of the imaginary number
double Im	Imaginary part of the imaginary number
cComplex()	Overridden Default Constructor
cComplex(**double** argRe, **double** argIm)	Overloaded Constructor. Sets Re=argRe and Im=argIm
cComplex(&argZ)	Copy Constructor
void setRe(**double** argRe)	Sets Re = argRe
void setIm(**double** argIm)	Sets Im = argIm
void setZ(**double** argRe, **double** argIm)	Sets Re = argRe and Im = argIm
double getRe()	Returns Re
double getIm()	Returns Im
cComplex & **operator** = (cComplex &argZ)	Overloaded Assignment Operator
cComplex & **operator** + (cComplex &argZ)	Overloaded + Operator
bool isTemporary	True if the object is *temporary*
void killWhenTmp(cComplex *m);	Deletes m

b) Modify the definition of the assignment operator of the class cComplex (in Listing 5) in order to avoid self assignment. [2 Marks]

c) Explain the purpose of method killWhenTemp() used in the overloaded + operator given in Listing 6, by using an appropriate example of application. [2 Marks]'

d) Multiplication of complex numbers z_1 and z_2 where $z_1 = a + ib$ and $z_2 = c + id$, results in $z = z_1 z_2 = (ac - db) + i(ad + bc)$. Give the definition of overloaded operator $*$ for the said multiplication $z_1 z_2$. [2 Marks]

Q3. a) Why inheritance is required in computer programming?. Explain by using an example. [4 Marks]

b) Assume that the variable x is declared in the base class. Explain the accessibility of x in the derived class, if the applied access specifier on tox is **private**, **public** or **protected**. [2 Marks]

c) What is polymorphism? Explain by using an example. [2 Marks]

Listing 1: cList class declaration as it appears in the file cList.h

```
#ifndef CLIST_H
#define CLIST_H

#define N 10

class cList
{
    public:
        cList(); // Creates Empty List
        cList(double argx[]); // Creates List initialized with argx[]
        bool   InsertAt(int index, double argx );// insert value at given index
        double FindAt(int index);// find the value at given index
        bool Find(double argx);// check if given value exists
        int FindIndex(double argx);//get the index of a given value
        cList Reverse();

        virtual ~cList();

    private:
        double x[N];
};

#endif // CLIST_H
```

Listing 2: Constructors of cList as it appears in the file cList.cpp

```
cList::cList()
{    int i=0; while(i<N) { x[i]=0; ++i;} }

cList::cList(double argx[])
{     int i=0; while(i<N) { x[i]=argx[i];++i;} }
```

Listing 3: cComplex class declaration as it appears in the file cComplex.h

```
#ifndef CCOMPLEX_H
#define CCOMPLEX_H

class cComplex
{
    public:
        cComplex();
        cComplex(double argRe, double argIm);
        cComplex(cComplex &argZ);
        virtual ~cComplex();
        void setRe(double argRe);
        void setIm(double argIm);
        void setZ(double argRe, double argIm);
        double getRe();
        double getIm();

        cComplex & operator = (cComplex &argZ);
```

```
        cComplex & operator + (cComplex &argZ);

    private:
        double Re;
        double Im;
        bool isTemporary;
        void killWhenTmp(cComplex *m);
};
#endif // CCOMPLEX_H
```

Listing 4: Constructors of cComplex as it appears in the file cComplex.cpp

```
cComplex::cComplex()
{    Re = 1; Im = 1; isTemporary = false;}

cComplex::cComplex(double argRe, double argIm)
{    Re = argRe;Im = argIm;isTemporary = false;}

cComplex::cComplex(cComplex &argZ)
{    Re = argZ.Re;Im = argZ.Im;killWhenTmp(&argZ);}
```

Listing 5: overloaded assignment operator of cComplex

```
cComplex & cComplex::operator = (cComplex &argZ)
{    Re = argZ.Re; Im = argZ.Im;
     killWhenTmp(&argZ);
     return *this;
}
```

Listing 6: overloaded + operator of cComplex

```
cComplex & cComplex::operator + (cComple &argZ)
{    cComplex *ptr = new cComplex();
     ptr->isTemporary = true;
     ptr->Re = Re + argZ.Re;
     ptr->Im = Im + argZ.Im;

     killWhenTmp(&argZ);
     killWhenTmp(this);
     return *ptr;
}
```