Air University
(Final Examination: Spring 2024)

Subject:	Object Oriented Programming	Total Marks:	100
Course Code:	CS112	Date:	
Class:	BS-CYS	Time:	
Semester:	II	Duration:	3 Hours
Section:	A & B	FM Name:	Dr. Kashif Kifayat

HoD Signatures:_____ FM Signatures:_____

Note:
- **All questions must be attempted.**
- **Understanding question is part of the examination.**

There are three sections in this paper. Please attempt all sections.

Q1: Please write down the output of the programs given below. [50 Marks] **[CLO2]**

Q1-a.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

#define PI 3.1416
#define AREA(r) r * r * PI

class Distance {
public:
    int meter, feet, inch;

    Distance()
    {
       this->meter =0;
       this->feet = 0;
        this->inch = 0;
    }

    Distance(int m, int f, int i)
    {
       this->meter = m;
        this->feet = f;
        this->inch = i;
    }
```

```
    Distance operator+(Distance& d2)
    {
        // Create an object to return
        Distance d3;
        d3.meter = this->meter + d2.meter;
        d3.feet = this->feet + d2.feet;
        d3.inch = this->inch + d2.inch;
        return d3;
    }
};



int _tmain(int argc, _TCHAR* argv[])
{

    Distance d1(15, 8, 9);
    Distance d2(10, 10, 2);
    Distance d3;

    d3 = d1 + d2;

    cout << "\nTotal Meter, Feet & Inches: " <<d3.meter<<"-"<<
            d3.feet << "-" << d3.inch;

}
```

Q1-b.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

#define PI 3.1416
#define AREA(r) r * r * PI

class Distance {
public:
    int meter, feet, inch;

    Distance()
    {
           this->meter =0;
           this->feet = 0;
        this->inch = 0;
    }

    Distance(int m, int f, int i)
    {
```

```
            this->meter = m;
        this->feet = f;
        this->inch = i;
    }

    Distance operator+(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter + d2.meter;
        d3.feet = this->feet + d2.feet;
        d3.inch = this->inch + d2.inch;
        return d3;
    }

     Distance operator-(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter - d2.meter;
        d3.feet = this->feet - d2.feet;
        d3.inch = this->inch - d2.inch;
        return d3;
    }
};



int _tmain(int argc, _TCHAR* argv[])
{

    Distance d1(15, 8, 9);
    Distance d2(10, 10, 2);
    Distance d3;

    // Use overloaded operator
    d3 = d1 + d2 - d2;

    cout << "\nTotal Meter, Feet & Inches: " <<d3.meter<<"-"<<
             d3.feet << "-" << d3.inch;
}
```

Q1-c.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

#define PI 3.1416
```

```
#define AREA(r) r * r * PI

class Distance {
public:
    int meter, feet, inch;

    Distance()
    {
           this->meter =0;
           this->feet = 0;
        this->inch = 0;
    }

    Distance(int m, int f, int i)
    {
         this->meter = m;
        this->feet = f;
        this->inch = i;
    }

    Distance operator+(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter + d2.meter;
        d3.feet = this->feet + d2.feet;
        d3.inch = this->inch + d2.inch;
        return d3;
    }

     Distance operator-(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter - d2.meter;
        d3.feet = this->feet - d2.feet;
        d3.inch = this->inch - d2.inch;
        return d3;
    }

     Distance operator*(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter * d2.meter;
        d3.feet = this->feet * d2.feet;
        d3.inch = this->inch * d2.inch;
        return d3;
    }

     Distance operator/(Distance& d2)
    {
```

```
        // Create an object to return
        Distance d3;
           d3.meter = this->meter / d2.meter;
        d3.feet = this->feet / d2.feet;
        d3.inch = this->inch / d2.inch;
        return d3;
    }

     Distance operator%(Distance& d2)
    {
        // Create an object to return
        Distance d3;
           d3.meter = this->meter % d2.meter;
        d3.feet = this->feet % d2.feet;
        d3.inch = this->inch % d2.inch;
        return d3;
    }
};


int _tmain(int argc, _TCHAR* argv[])
{

     Distance d1(5, 10, 15);
     Distance d2(5, 5, 5);
     Distance d3(10, 10, 10);
     Distance d4;

    // Use overloaded operator
    d4 = d1 + d2 * d3 / d2 % d1;

    cout << "\nTotal Meter, Feet & Inches: " <<d4.meter<<"-"<<
             d4.feet << "-" << d4.inch;
}
```

Q1-d.

```
     #include "stdafx.h"
     #include <iostream>
     #include <conio.h>

     using namespace std;

     class A
     {
                  protected:
                  int a;

                  public:
                      void set_A()
```

```
                {
                        a=10;
                }
                void disp_A()
                {
                      cout<<endl<<"Value of A="<<a;
                }
};

class B: public A
{
          protected:
               int b;

          public:
                 void set_B()
               {
               b=a+2;

               }


               void disp_B()
                 {
                    cout<<endl<<"Value of B="<<b;
                 }
};

class C: public B
{
      int c,p;

      public:
          void set_C()
          {
                 c=a+b;
           }

           void disp_C()
           {
                 cout<<endl<<"Value of C="<<c;
           }

            void cal_product()
             {
                  p=a*b*c;
                 cout<<endl<<"Product of "<<a<<" * "<<b<<" *
"<<c<<" = "<<p;
              }
```

```
};


int _tmain(int argc, _TCHAR* argv[])
{
   C _c;
    _c.set_A();
    _c.set_B();
    _c.set_C();
    _c.disp_A();
    _c.disp_B();
    _c.disp_C();
    _c.cal_product();

  _getch();
   return 0;
}
```

Q1-e.

```
#include "stdafx.h"

using namespace std;
#include <iostream>
#include <conio.h>

class A
{
    protected:
    int a;
    public:
    void get_a()
    {
       a=5;
    }
};

class B : public A
{
    protected:
    int b;
    public:
    void get_b()
    {
       b=4;
         get_a();
    }
};
class C
{
    protected:
```

```
    int c;
    public:
    void get_c()
    {
        c=8;

    }
};

class D : public B, public C
{
    protected:
    int d;
    public:
    void mul()
    {
         get_b();
         get_c();
         cout << "Multiplication of a,b,c is : " <<a*b*c;
    }
};
```

Q1-f. Please find error (s) in the code below.

```
#include "stdafx.h"

using namespace std;
#include <iostream>
#include <conio.h>

class A
{
    protected:
    int a;
    public:
    void get_a()
    {
       a=5;
    }
};

class B : public A
{
    protected:
    int b;
    public:
    void get_b()
    {
       b=4;
```

```
    }
};
class C
{
    private:
    int c;
    public:
    void get_c()
    {
        c=8;
           get_a();
    }
};

class D : public B, public C
{
    protected:
    int d;
    public:
    void mul()
    {
         get_b();
         get_c();
         cout << "Multiplication of a,b,c is : " <<a*b*c;
    }
};



int _tmain(int argc, _TCHAR* argv[])
{
    D d;
    d.mul();
      _getch();
  return 0;

}
```

Q1-g. Please find error (s) in the code below.

```
#include <iostream>
#include <conio.h>
using namespace std;

class BankAccount {
private:
    std::string accountHolder;
public:
    // Constructor
    BankAccount(std::string holder, double initialBalance) {
        accountHolder = holder;
```

```
        balance = initialBalance;
    }
    // Deposit method
    void deposit(double amount) {
        balance += amount;
    }
    // Withdraw method
    void withdraw(double amount) {
        if (amount > balance) {
            std::cout "Error: Insufficient funds.\n";
        } else {
            balance -= amount;
        }
    }
    // Display account info
    void displayInfo() {
        std::cout << "Account Holder: " << accountHolder << '\n';
        std::cout << "Balance: $" << balance << '\n';
    }
};
int _tmain(int argc, _TCHAR* argv[])
{
    BankAccount myAccount("John Doe", 500.0);
    myAccount.deposit(-50);
    myAccount.displayInfo();
    return 0;
}
```

Q1-h. Please write the outputs.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;


class base {

protected:

        int a, b, c, d;

public:
     void fun_1() { cout << "base-1\n"; a=10; b=20;c=30;d=40;}
    virtual void fun_2() { cout << "base-2\n"; }
    virtual void fun_3() { cout << "base-3\n";
a=100;b=200;c=300;d=400;}
    virtual void fun_4() { cout << "base-4";
cout<<"="<<a+b+c+d<<endl;}
};
```

```
class derived : public base {
public:
    void fun_1() { cout << "derived-1\n"; a++;}
    void fun_2() { cout << "derived-2="; d=a+b+c;
cout<<d<<"\n";}
    void fun_4(int x) { cout << "derived-4\n"; }
};

int _tmain()
{
    base* p;
    derived obj1;
    p = &obj1;
    p->fun_1();
    p->fun_2();
    p->fun_3();
    p->fun_4();

     _getch();
    return 0;
}
```

Q1-i.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>
#include <thread>
using namespace std;

void foo(int Z)
{
    for (int i = 0; i < Z; i++) {
        cout << "Thread using function"
                " pointer as callable\n";
    }
}

class thread_obj {
public:
    void operator()(int x)
    {
        for (int i = 0; i < x; i++)
            cout << "Thread using function"
                    " object as callable\n";
    }
};

class Base {
```

```
public:

    void foo()
    {
        cout << "Thread using non-static member function "
                "as callable"
             << endl;
    }

    static void foo1()
    {
        cout << "Thread using static member function as "
                "callable"
             << endl;
    }
};


int main()
{
    cout << "Threads 1 and 2 and 3 "
            "operating independently"
         << endl;

    thread th1(foo, 3);

    thread th2(thread_obj(), 3);

    auto f = [](int x) {
        for (int i = 0; i < x; i++)
            cout << "Thread using lambda"
                    " expression as callable\n";
    };

    thread th3(f, 3);

    Base b;

    thread th4(&Base::foo, &b);
    thread th5(&Base::foo1);

    th1.join();
    th2.join();
    th3.join();
    th4.join();
    th5.join();
    _getch();
    return 0;
}
```

Q1-j.

```
#include "stdafx.h"
#include <iostream>
#include <conio.h>

using namespace std;

#define PI 3
#define AREA(r) r * r * PI
#define SQR(x) x * x
#define CUBE(x) x * x * x

int _tmain(int argc, _TCHAR* argv[])
{

  int radius = 5;
  int r;
  int c;

  r = SQR(AREA(radius));
  c = CUBE(5);
  cout << "Round 1 = "<< r <<"="<<c<<"="<<r/c<<endl<<endl;


     //Example 3
  #ifdef PI
    r+=10;c+=5;
  #else
    cout << "PI undefined\n";
  #endif

  #undef PI

  #ifdef PI
    cout << "PI defined\n";
  #else
    r+=2;c+=2;
  #endif

     cout<< "Round 2 = "<<r<<" = "<<c<<endl;

  _getch();
  return 0;

}
```

Q2: Theory **[25 Marks]**
[CLO1]

Q2-a: What is the difference between function overloading, constructor overloading and operator overloading (5). Please write a program where you use all three in one example/program (15).

15 Marks

Q2-b: What are three access specifiers, and what are the differences between them? Please describe each of them and give an example of each.

10 Marks

Q3: Coding **[25 Marks]**
[CLO2]

Q3-a: Give a programming example that overloads "==" operator with its use.

5 Marks

Q3-b: Write program using class Geometry. The program should ask the user for length and width if the length and width both are same then it should call square function to calculate its area and parameter otherwise it should call rectangle function to calculate the area and parameter of the rectangle.

10 Marks

Q3-c: Write a C++ program to make a car class with wheels, doors as private data members and cur_speed as public data member while speed and break as public member functions. Use constructor with default value of wheels=4,doors=2 and cur_speed=0 for initialization. Make two objects in the name of 14errari and hino. Ferrari is initialized through default values while hino has 10 wheels, 4 doors. Every time when you call speed function cur_speed is increased by 5 while break function decrease it by 5. Display current speed.

10 Marks
