Roll No : _____ QUIZ #1 – Date: 24-10-11

Name : Solution

Dry run and write the output of the following code. [20]

```
#include <iostream>
using namespace std;
class intclass
{
  private:
    int var;
  public:
    int getVar()
    {
      return var;
    }
    void setVar(int v)
    {
      var = v;
    }
    intclass()
    {
      cout << "created stateless" << endl;
    }
    intclass(int v)
    {
      var = v;
      cout << "created " << var << endl;
    }
    intclass(intclass &r)
    {
      var = r.var;
      cout << "copy created " << var << endl;
    }
    ~intclass()
    {
      cout << "destroyed "  << var << endl;
    }
    void display()
    {
      cout << var ;
    }
    intclass operator=(intclass rhs)
    {
      cout << "entered in assignment" << endl;
      var = rhs.var;
      cout << "leaving assignment" << endl;
      return rhs;
    }
    intclass operator+(intclass rhs)
    {
      cout << "entered in plus" << endl;
      intclass res;
      res.var = var + rhs.var;
      cout << "leaving plus" << endl;
      return res;
    }
};

int main(void)
{
  intclass r0(10);
  intclass r1 = r0;
  intclass r2 = intclass(17);
  intclass r3;
  cout << "summing" << endl;
  r3 = r1 + r2 ;
  cout << "sum is " << r3.getVar() << endl;
  return -1;
}
```

```
created 10
copy created 10
created 17
created stateless
summing
copy created 17
entered in plus
created stateless
leaving plus
copy created 27
destroyed 27
destroyed 17
entered in assignment
leaving assignment
copy created 27
destroyed 27
destroyed 27
sum is 27
destroyed 27
destroyed 17
destroyed 10
destroyed 10
```

Define a friend overloading of **!=** operator for Rational Numbers. [5]

```
bool Rational::operator!=(const Rational &lhs, const Rational &rhs)
{
	bool result = true;
	Rational lo = this->simplify(lhs);
	Rational ro = this->simplify(rhs);
	if(lo.p == ro.p && lo.q == ro.q)
	{
		result = false;
	}
	return result;
}
```

Define stream extraction operator overload for Vector3d class. [5]

```
istream & Vector3d::operator>>(istream &lhs, Vector3d &rhs)
{
	lhs >> rhs.x;
	lhs >> rhs.y;
	lhs >> rhs.z;
	return lhs;
}
```

1. Using OOP principles learned, write minimum number of statements (of main function) to start a ***PaintBrush*** program. (**2** Marks)

```
PaintBrush pb;
Pb.start();
```

2. Write definition (function's implementation) of a destructor of ***InternetExplorer*** class. Destructor just has to display the message "**IE destroyed**". (**2** Marks)

```
InternetExplorer::~InternetExplorer()
{
  cout << "IE destroyed"
}
```

3. What is meant by the STATE of an object? (**2** Marks)

Values stored in the data members of the object is collectively called STATE of that object at that instance of time.

4. Write prototypes of getters and setters of **date birthDate**, a data member of student class. (**2** Marks)

```
date student::getBirthDate() const
{
      return birthDate
}

void student::setBirthDate(date d)
{
      birthdate = d;
}
```

5. Write names of at least 10 key concepts discussed so far in the OOP course. (**2** Marks)

Object, class, private, public, function members, scope resolution, getters, setters, constructors, destructors, STATE, overloading, abstraction, encapsulation, inheritance, polymorphism

6. Considering you are provided a standard point2d class having float data members x and y is available. Using that class, you have to **describe** a class for **rectangle's** objects which are oriented in parallel to axis of coordinate system. Rectangle class must have only top-left and bottom-right points as data members. Public interface in rectangle should have getters and setters, constructors, and area, perimeter and diagonal functions. At end, define a parameterized constructor and area member function. (**10** Marks) [only create class for rectangle]

```
class Rectangle
{
  private:
        point2d tl; // topLeft
        point2d br; // bottomRight
  public:
        point2d getTopLeft() const;
        point2d getBottomRight() const;
        void setTopLeft(const point2d &p);
        void setBottomRight(const point2d &p);
        void setRectangle(const point2d &tl, const point2d &br);

        Rectangle();
        Rectangle(const point2d &tl, const point2d &br);
        Rectangle(const Rectangle &r);

        Rectangle(const point2d &tl, float len, float wid);
        Rectangle(const point2d &tl, float len);

        float area() const;
        float diagonal() const;
};

Rectangle::Rectangle(const point2d &p1, const point2d &p2)
{
  if (p1.x < p2.x && p1.y < p2.y)
  {
        tl = p1;
        br = p2;
  }
  else
  {
        throw string("Points are not properly ordered!");
  }
}

Rectangle::Rectangle(const point2d &p1, float len)
{
  point2d p2(p1.x+len, p1.y+len);
  Rectangle(p1, p2); // is it possible
}

float Rectangle::area() const
{
  return (br.x - tl.x) * (br.y - tl.y);
}
```