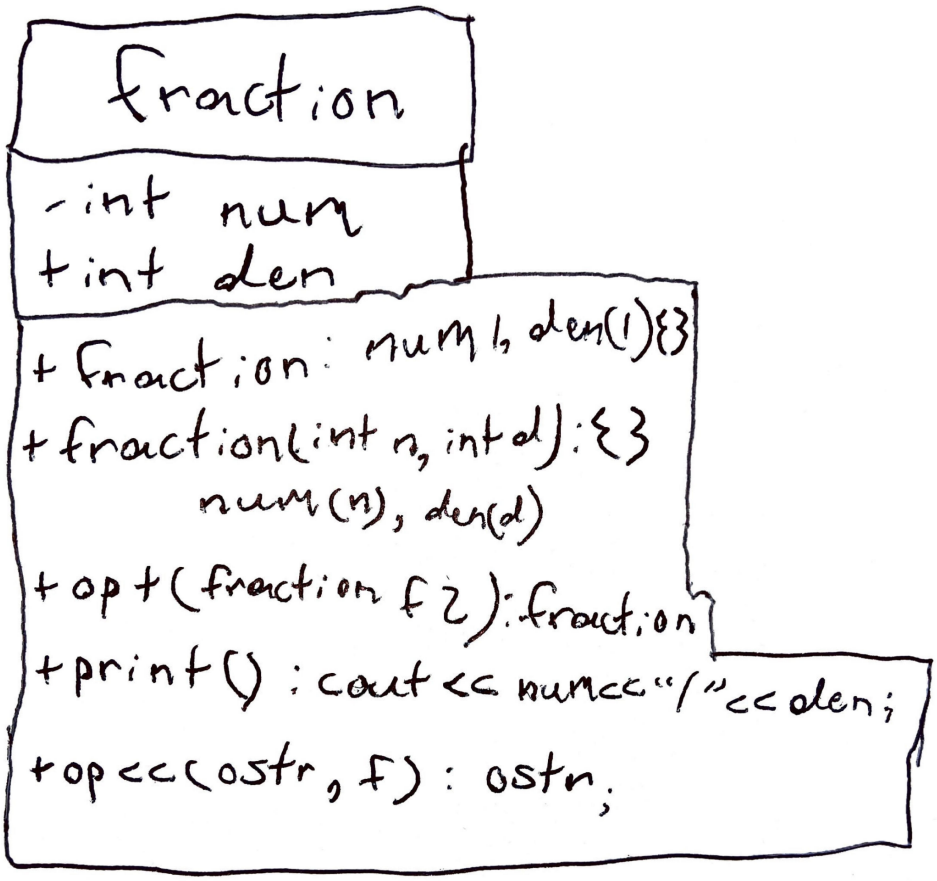
“I pledge my honor I have abided by the Stevens Honor system” -Alex Gaskins

Question 1: UML (10 points)

Create the unified modeling language (UML) for the following C++ code

```
class fraction{
private:
   int num; // numerator
public:
   int den; // denominator
   fraction() : num(1), den(1){};
   fraction(int n, int d) : num(n), den(d){};
   fraction operator+(fraction f2){
      return fraction((this->num*f2.den + this->den*f2.num),    this->den*f2.den);}
   void print(){cout << this->num << " / " << this->den << endl;}
   friend ostream& operator<<(ostream& ostr, fraction f){
      ostr << f.num << " / " << f.den << endl;
      return ostr;
   }
};
```

(Sorry for my awful handwriting)



Question 2: Conditional Statements (15 points)

Use the below C++ code to answer the following conditions

```
If (Name == "Diamond" && x > 10){
   money = 100 * x;
   bank += 1;
}
else {
   bank -= 1;
}
cout << money << " " << bank << endl;
```

- Write conditional operator short form for **if & else** condition example above

  Ternary:

  (Name == “Diamond” && x > 10) ? (money = 100*x, bank += 1) : bank -= 1

- What are the values of money and bank if Name = ”Silver” and original money = 0, bank = 50, and x = 100?

  money = 0;
  bank = 49;

- What are the values of money and bank if Name = ”Diamond” and original money = 5, bank = 40, and x = 100?

  money = 100*100 = 10000;
  bank = 41;

Question 3: Bug fixes (15 points)

Find three distinct errors in the following program and suggest appropriate fixes

```
1  #include <iostream>
2  using namespace std;
3
4  class Shape{
5
6  public:
7      int l,w;
8
9      void set_values(int l, int w){
10         this->l=l;
11         this->w=w;
12     }
13
14     void show_data(){
15         cout<<l<<endl;
16         cout<<w<<endl;
17     }
18
19     float area (){}
20 };
```

```
21 class Rect:Shape{
22 public:
23         float area (){
24                 return (l*w);
25         }
26 };
27
28 int main (){
29
30     Rect r;
31     r.set_values(2,3);
32
33     r.show_data();
34
35     Shape *s=&r;
36     s.area();
37
38     return 0;
39 }
```

1.) Line 36

Error:
s.area(); s is defined as a pointer that is equal to address r, but using s.area() does not take into account that s is always a pointer.

Solution:
Change it to either (s*.area()) or point s to area: s->area().

2.) Line 35

Error:
Shape *s points to &r of type Rect. A pointer can only point to its same type (int->int, Shape -> Shape).

Solution:
Change *s to type Rect or change r to type Shape.

3.) Line 21

Error:
Shape is inherited as a private (default) member of Rect. Any functions called for type Rect from class Shape will not work unless Shape is specified as public to be used in main(). r.set_values() and r.show_data will not run in main() as a result.

Solution:
Set inherited class Shape to public: class Rect: public Shape.

Question4: C++ Concepts - True/False (10 points)

1. We can access the static variables with the class name. [ ]
2. In C++, assigning a pointer to a pointer does not involve copying the object. [ ]
3. In C++, dynamic memory allocation is done using malloc. [ ]
4. In C++, a constructor is a public class function that get called whenever an object's destroyed. [ ]
5. The Addition operator (+) is the only operator which overloaded by default. [ ]

1.) True
2.) True
3.) True
4.) False
5.) False

Question 5 C++ Classes and functions (10 points)

Write a class Calculator that holds two integer values x,y set by the class constructor. The class also contains a member function div that return x/y. The class should handle the possible exception in case (y=0)

Write main function to test the implementation of the Calculator class

```
//Calculator class
class Calculator {
  double x, y;

public:
  double add() {
          return x + y;
          }
  double subtract() {
          return x - y;
          }
  double multiply() {
           return x * y;
          }
  double divide() {
    if (y == 0) {
      cout << "Can't divide by 0!" << endl;
      return 0;
    } else {
      return x / y;
    }
  }

  void getValues() {
    cout << "Please give me the first value: ";
    cin >> x;
    cout << "Now the second value: ";
    cin >> y;
  }
};
```

```
//Main function
int main() {

  Calculator c;
  cout << "1 for addition";
  cout << "2 for subtraction";
  cout << "3 for multiplication";
  cout << "4 for division";
  cout << "Anything else to quit" << endl;

  int choice;
  cout << "Enter Choice: ";
  cin >> choice;

  switch (choice) {
  case 1:
    c.getValues();
    cout << "Answer is " << c.add() << endl;
    break;
  case 2:
    c.getValues();
    cout << "Answer is " << c.subtract() << endl;
    break;
  case 3:
    c.getValues();
    cout << "Answer is " << c.multiply() << endl;
    break;
  case 4:
    c.getValues();
    cout << "Answer is " << c.divide() << endl;
    break;
  }
}
```

Question 6 Pointers and Access Specifiers (15 points)

Given the below code, try to answer the following questions

```
#include <iostream>
using namespace std;

class Triangle
{

public:
    float width, height;
public:
    void set_data (float a, float b){
        width = a;
        height = b;
    }
    float area (){
        return (width * height / 2);
}
};

int main (){

    Triangle tri;
    tri.set_data (2,5);

    cout << tri.area() << endl;
    return 0;
```

1.) 5
2.) Result is the same, since the functions are using the private variables inside the class only. The calculations using width and height would not work outside of class Triangle without using the friend specifier.
3.) Will not run, as private values from class Triangle are being called outside of the class.
4.) Triangle *tri_p=&tri;
5.) //Constructor
Triangle(float a, float b) {
this->width = a;
this->height = b;
}

//Area Function
float Triangle::area() { return (width * height / 2); }

Question 7 C++ Concepts - multiple choices (15 points)

a.) c
b.) c
c.) e
d.) b
e.) a

Question 8 True or false – Explain (10 points)

a.) Yes, Mucus foo is called first, so it will be the first to be constructed.

b.) True, as this does not account for null cases, so if a class without any parameters is created at first, it would not work.