CS2 Notes

1. **All variables have a lifetime,** i.e., they are created at some point in time during the program's execution, and deleted/destroyed at a later point in time during the program's execution.

   - How to tell a variable's lifetime?

2. For any **object variable,** a constructor is called at its creation, the destructor is called at its deletion.

- *Which constructor of the class?*
  - *Depends on the parameters (or the lack of) provided:*

```
Rational a;                 //no-parameter constructor
Rational b(1,3);            //constructor with two int parameter
Rational c(3);
Rational d(a);              //copy constructor
```

3. If a class does not define a constructor, then **C++ provides a default no-parameter constructor**…

in which

1. *base/parent class's no-parameter constructor is called first (which in turn calls its parent class's no-paramter constructor, …)*
2. *all member variables's no-parameter constructors is called*

4. If a class does not define a destructor, then **C++ provides a default destructor**

in which

1. *all member variables's destructor are called one by one*
2. *call the base class's destructor (which in turns call the its base class's destructor).*

5. If a class does not provide them, **C++ provides default copy constructor, default assignment operator overload**

which makes a byte-by-byte copy

```
Rational d(a);    // copy a's memory byte-by-byte to d
```

a=b; //assignment operator is called

6. **Reference variables: create a second name (alias) for a variable.** (safer, more user-friendly than pointer!)

* a reference parameter in function: is refering to the actual argument

```
void swap (int& x, int &y)
{
    int tmp = y;
    y = x;
    x = tmp;
}

int main()
{
       int a[2] = {0, 100};
       swap (a[0], a[1]);    // on this call, x is referring to a[0], y is referring to a[1]
 }
```

* int x = 10, y=20;
   int& a = x; //a reference variable has to be initialized at the declaration, and canot be reassigned. a is referencing x, alias for variable x
   // There is no way to change a to refere some other variable!!!

```
  a = 100;
  cout <<x<<endl;
  a = y;    //What's being done here?
  cout <<x <<endl;
  a++;
  cout <<x<<"; " << y<<endl;
```

* return reference from a function

```
int& func(int i)
{
     static int a[4]={0,0,0,0};
     return a[i];
}

func(0)=10;       //as the func return a reference to a variable, we can assign a value to the variable being returend
cout <<func(0) <<endl;
```

7. Most operators can be overloaded as non-member functions and as member functions, except
- — except <<, >> can only be overloaded as non-member
- — [ ], = can only be overlaoded as member function

```
class Rational{
public:
     // overload + as member
     Rational operator+ (const Rational & secOp){
          Rational sum;
          sum.num = num*secOp.den+secOp.num*den;
          sum.den = den*secOp.den;
           return sum;
     }

};

Rational operator- (const Rational & first, const Rational & sec)
{
    …
}

int main()
{

   Rational a(1/3), b(1/2);
   Rational c = a+b;   //is same as    a.operator+(b);

   Rational d = a - b;   // is same as    operator- (a, b)
}
```