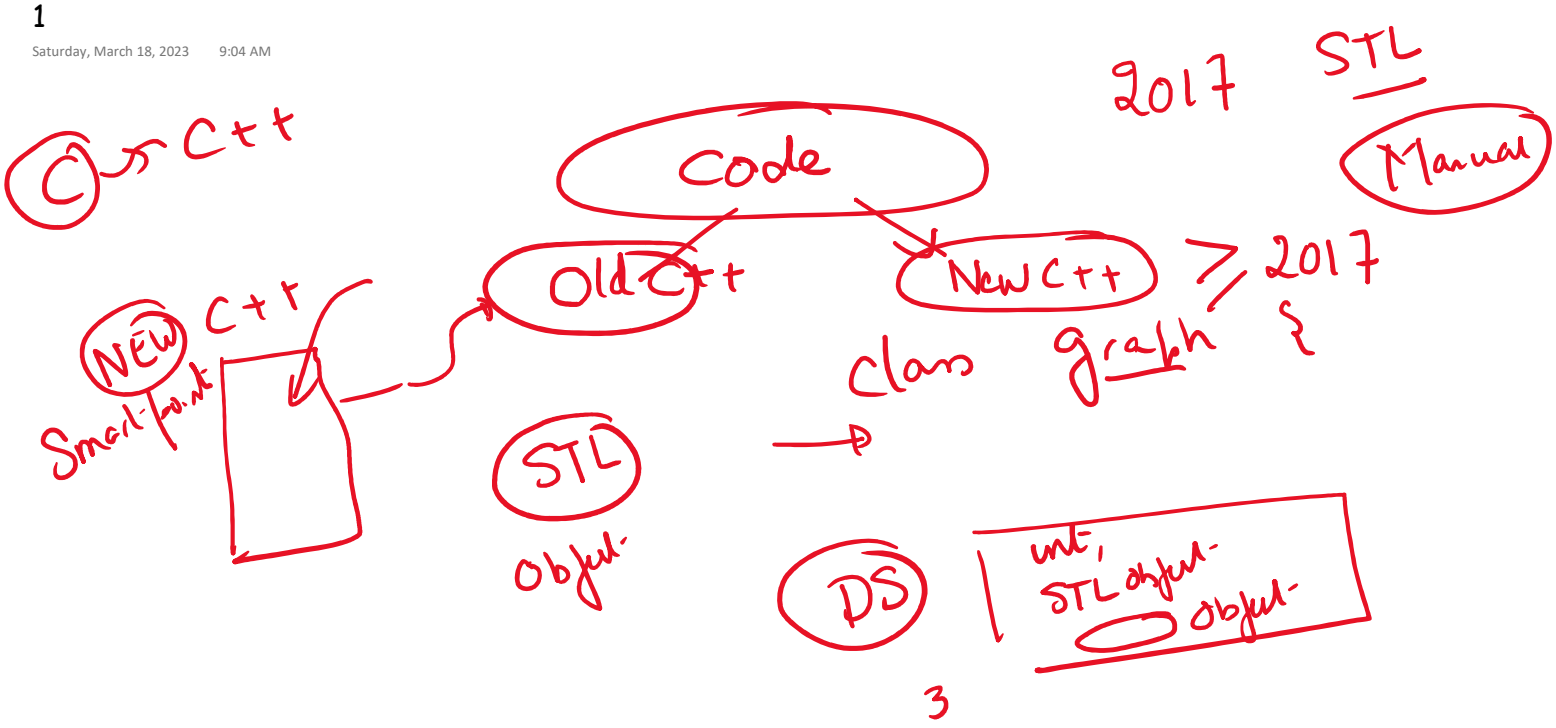
C++ 10

Saturday, March 18, 2023 8:23 AM

1

Saturday, March 18, 2023 9:04 AM



Saturday, March 18, 2023 9:08 AM

l-value & r-value

*(y+3)

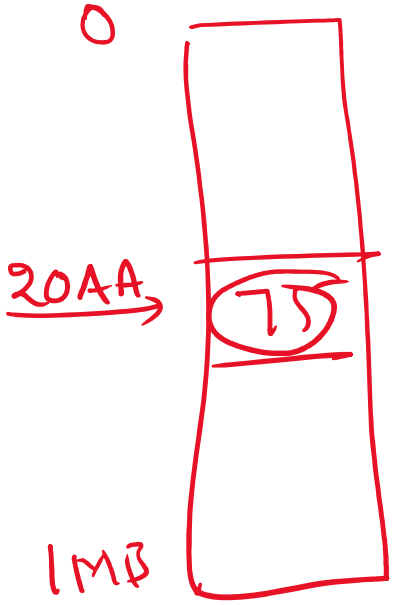
int x = 75

&

int *p = &x;

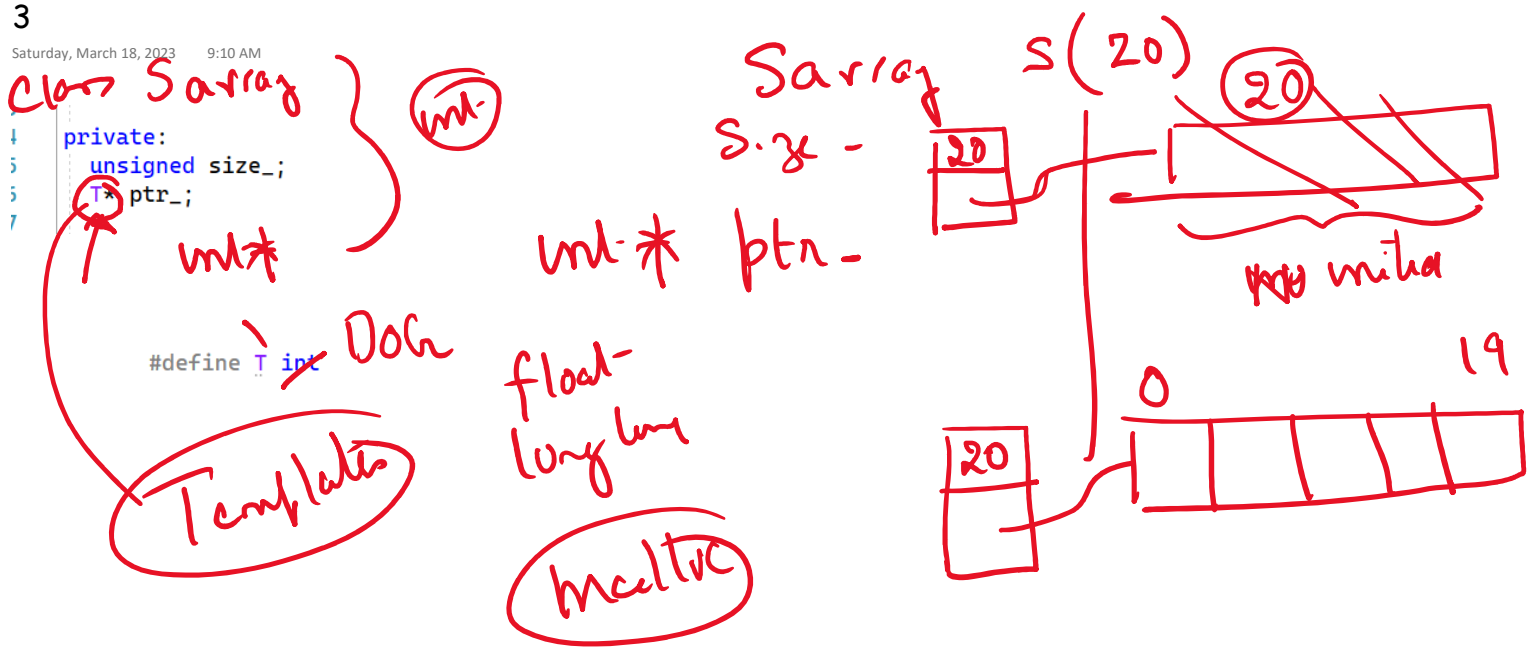
int y = 75

int x = y + 3



Saturday, March 18, 2023 9:10 AM

```
4   private:
5     unsigned size_;
6     T* ptr_;
7
```

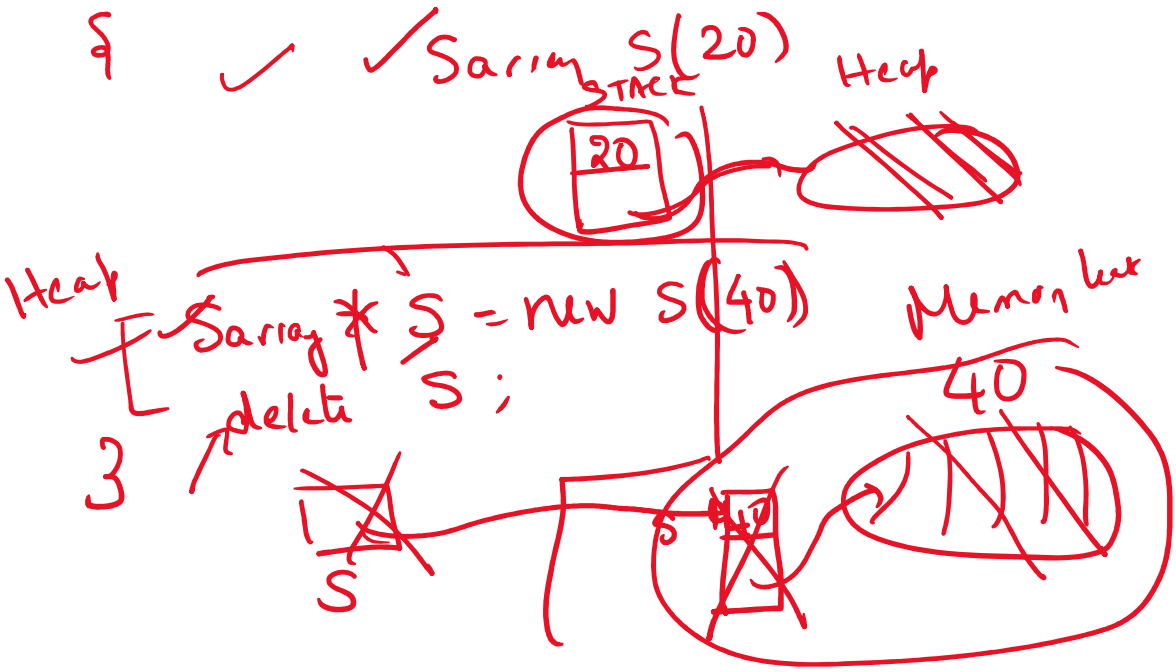


4

Saturday, March 18, 2023 9:16 AM

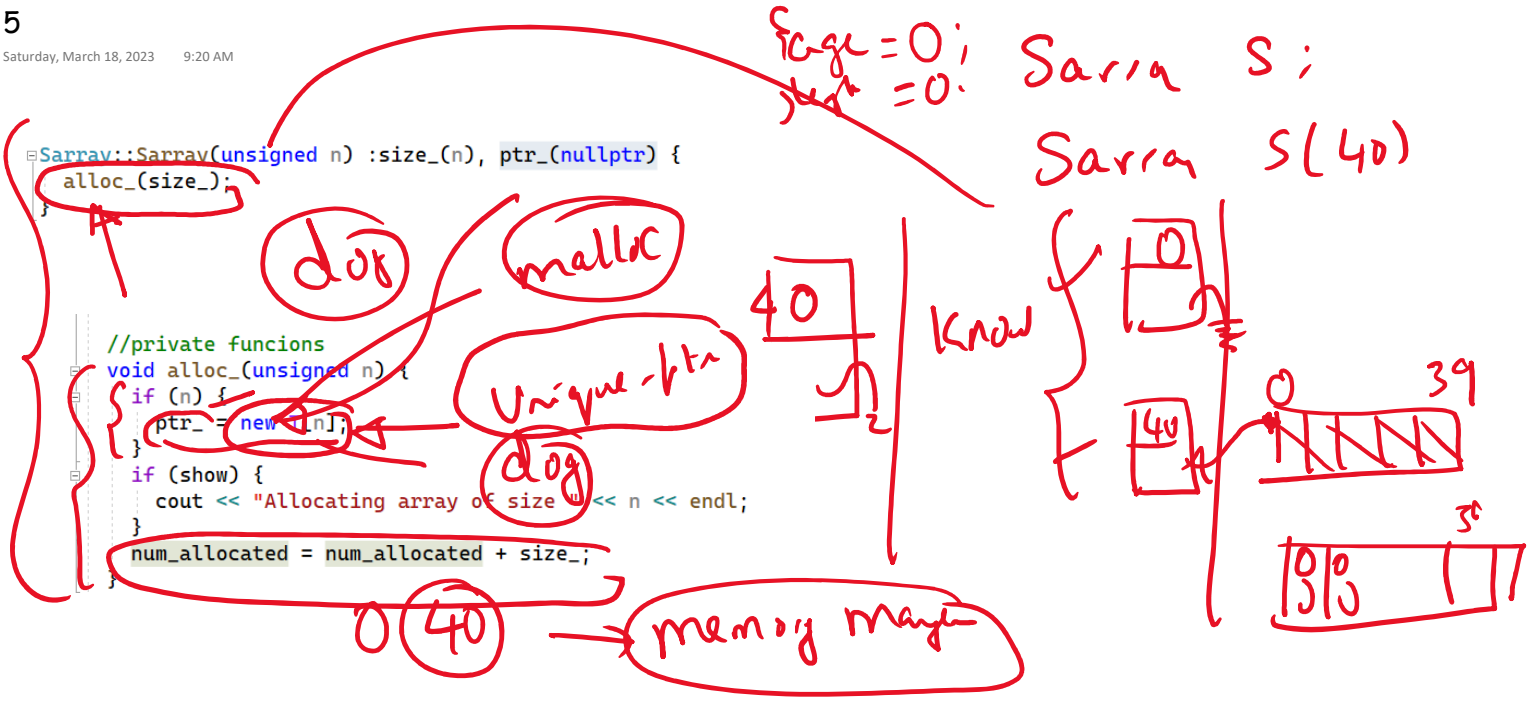
```
#define T int

private:
  unsigned size_;
  T* ptr_;
```

new



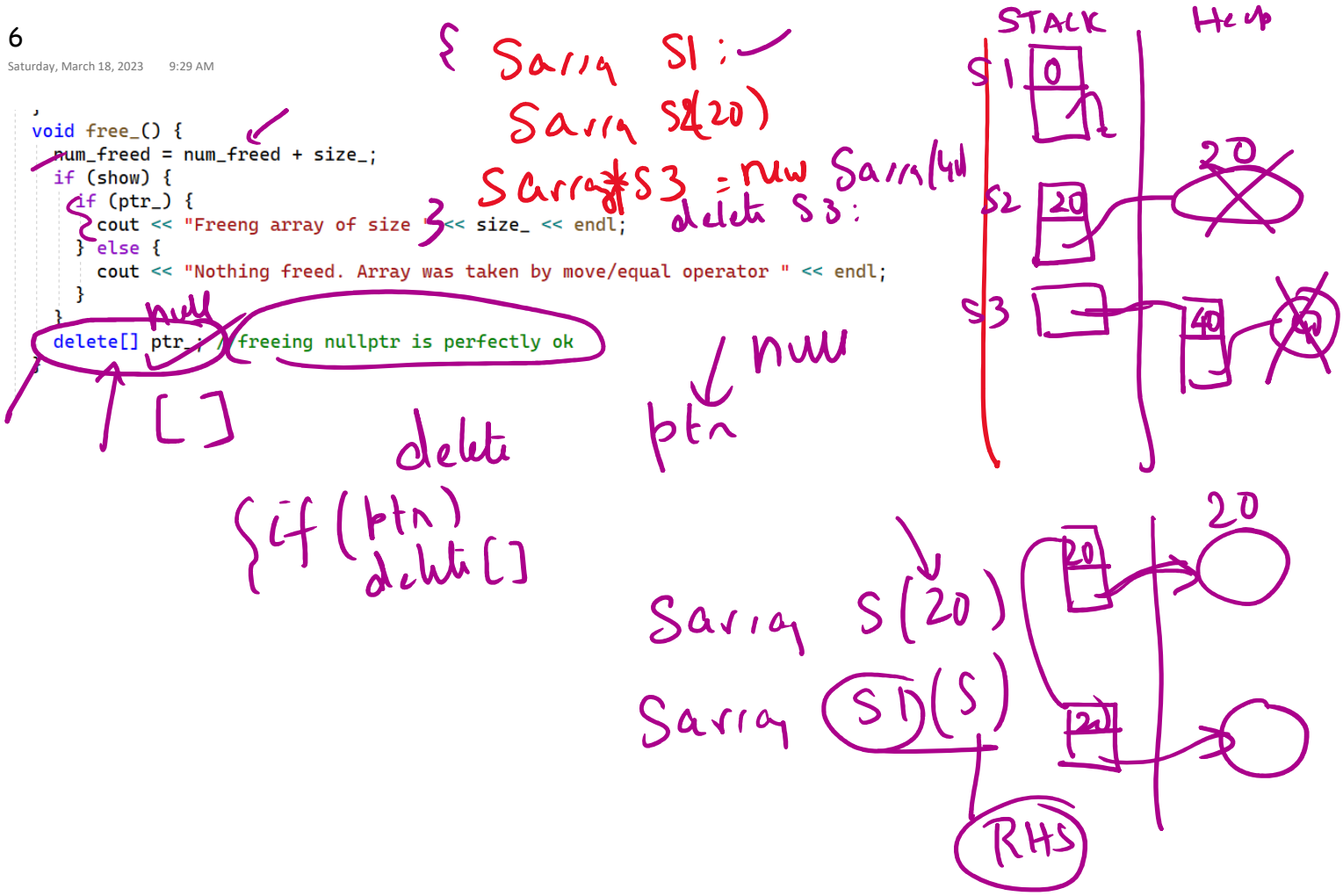
Saturday, March 18, 2023 9:20 AM

```
Sarray::Sarray(unsigned n) :size_(n), ptr_(nullptr) {
  alloc_(size_);
}

  //private funcions
  void alloc_(unsigned n) {
    if (n) {
      ptr_ = new int[n];
    }
    if (show) {
      cout << "Allocating array of size " << n << endl;
    }
    num_allocated = num_allocated + size_;
  }
```



Saturday, March 18, 2023 9:29 AM

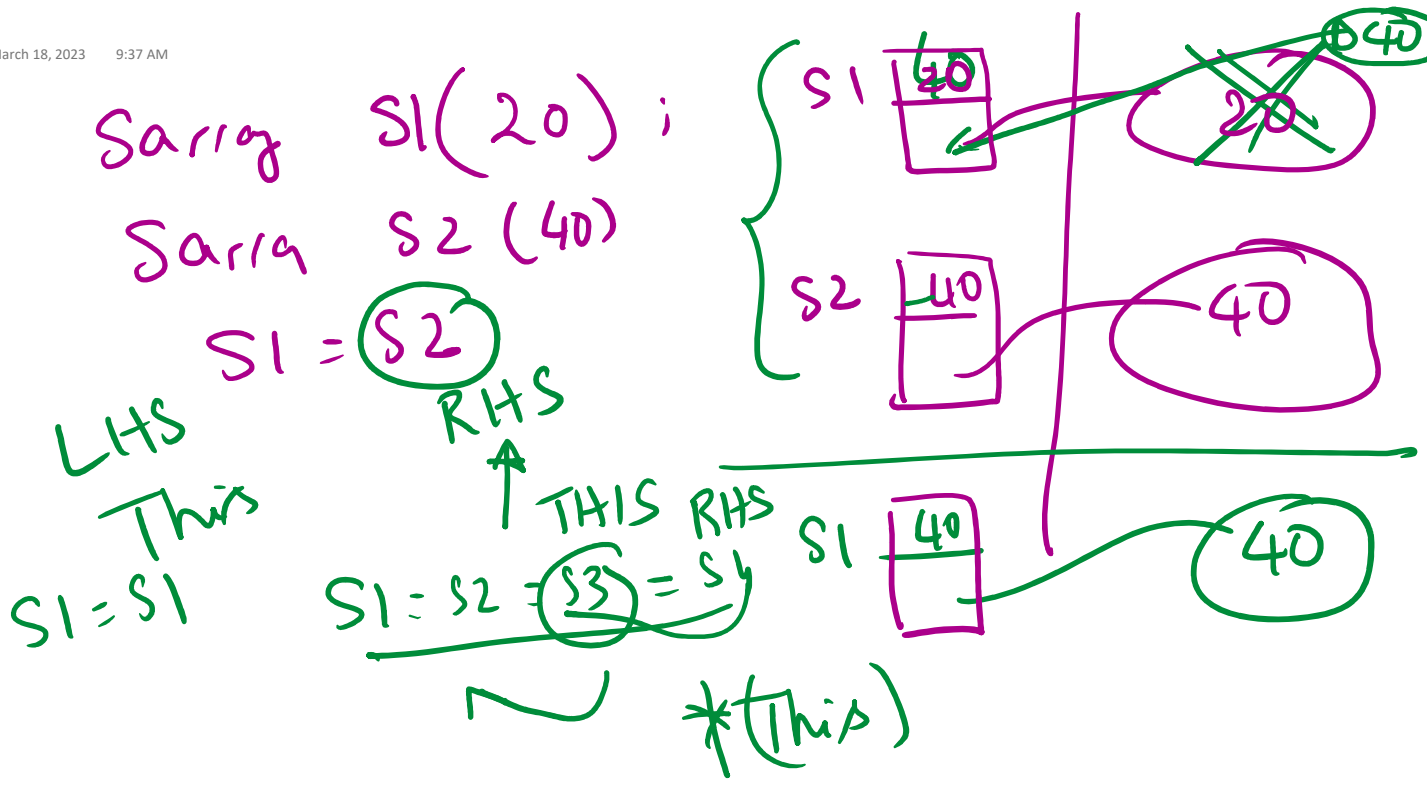
```
}
void free_() {
  num_freed = num_freed + size_;
  if (show) {
    if (ptr_) {
      cout << "Freeng array of size " << size_ << endl;
    } else {
      cout << "Nothing freed. Array was taken by move/equal operator " << endl;
    }
  }
  delete[] ptr_; // freeing nullptr is perfectly ok
}
```



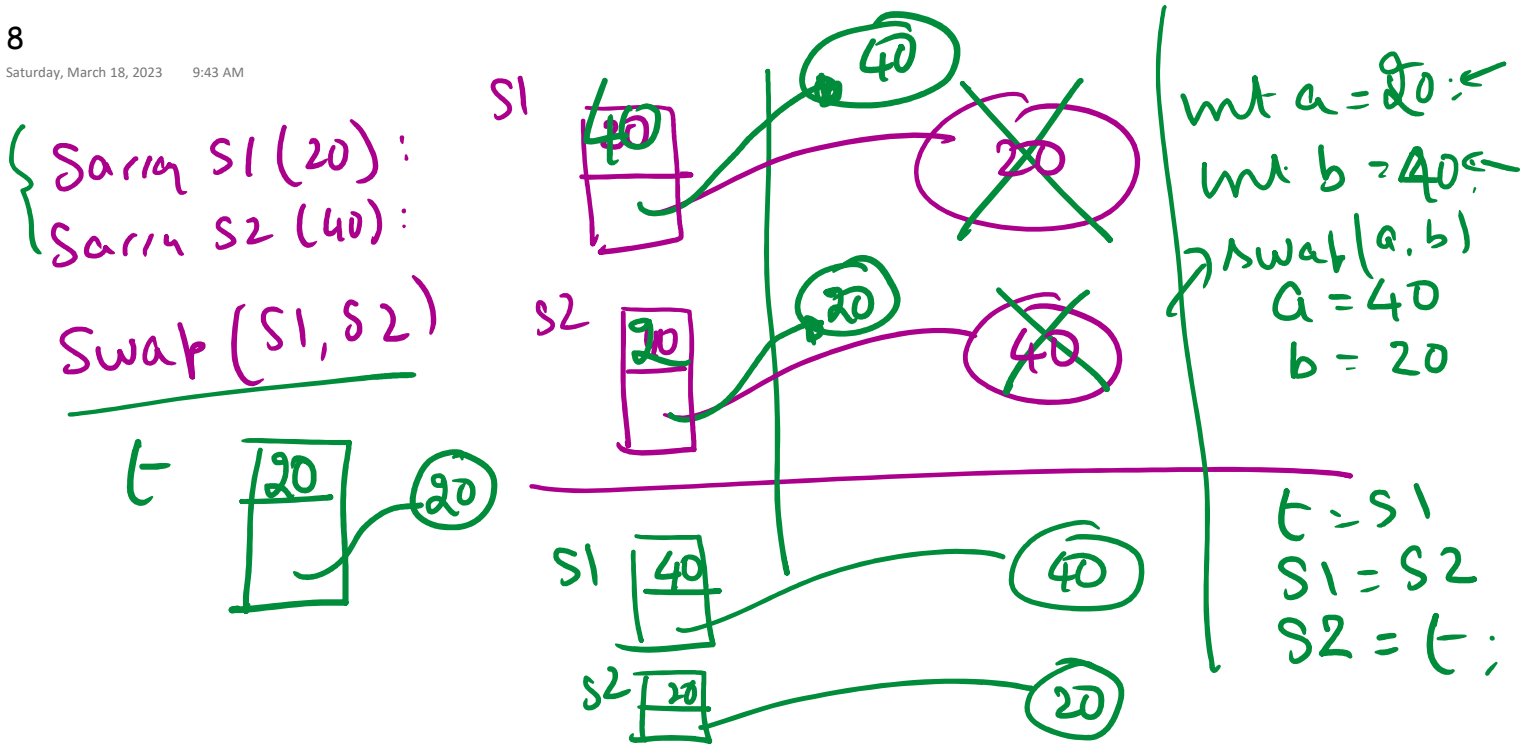
7

Saturday, March 18, 2023 9:37 AM

7



Saturday, March 18, 2023 9:43 AM

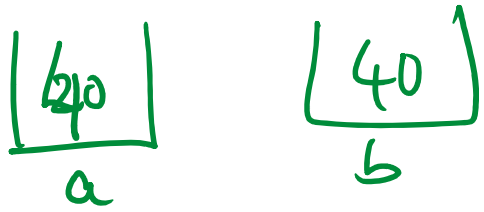


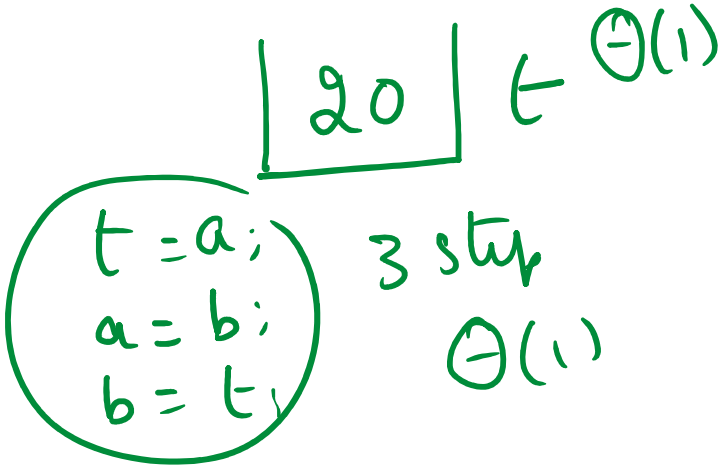
9

Saturday, March 18, 2023 9:45 AM

int a = 20
int b = 40

swap(a, b)
a = 40, b = 20





9

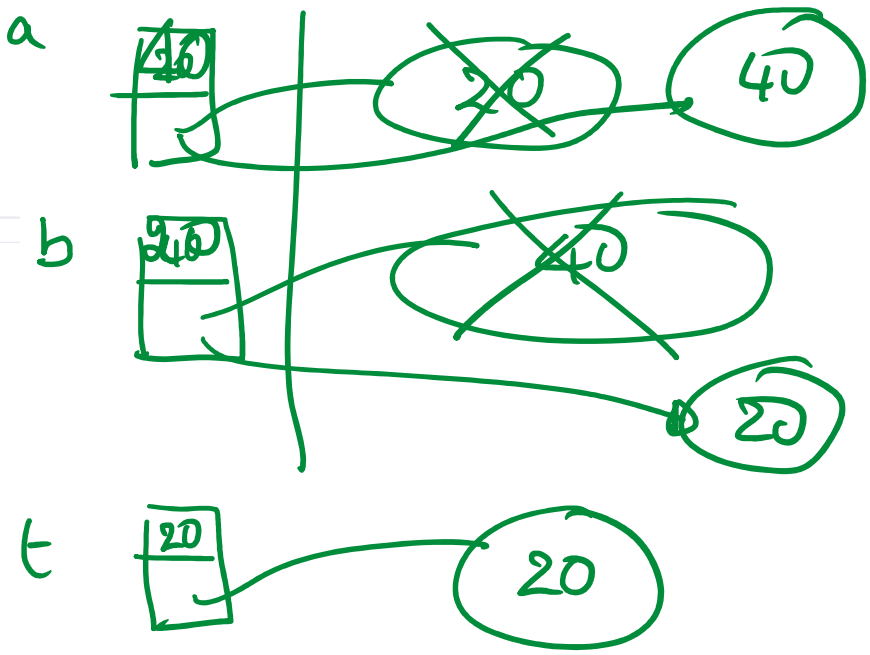
Saturday, March 18, 2023 9:52 AM

```
25      Sarray::num_freed = 0;
26      {
27          Sarray a(20);
28          Sarray b(40);
29          Swap(a, b);
30      }
```

t = a;

a = b

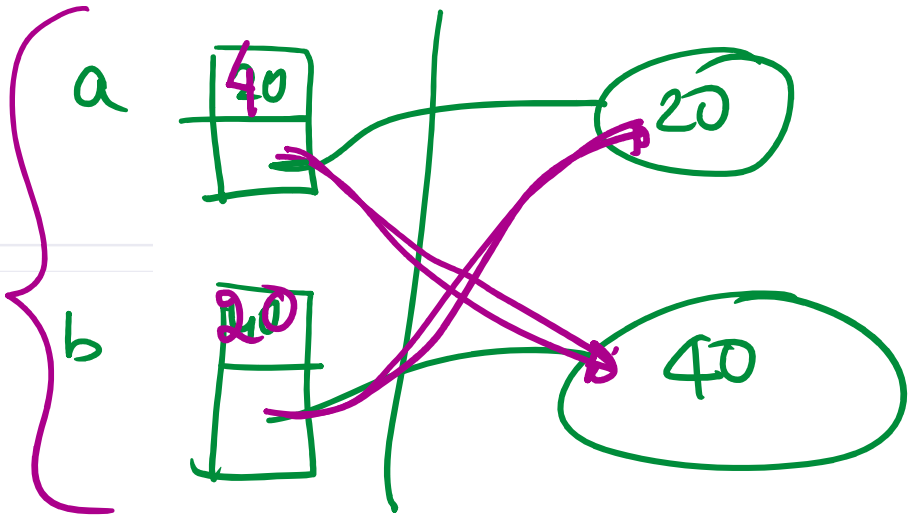
b = t;



10

Saturday, March 18, 2023 9:54 AM

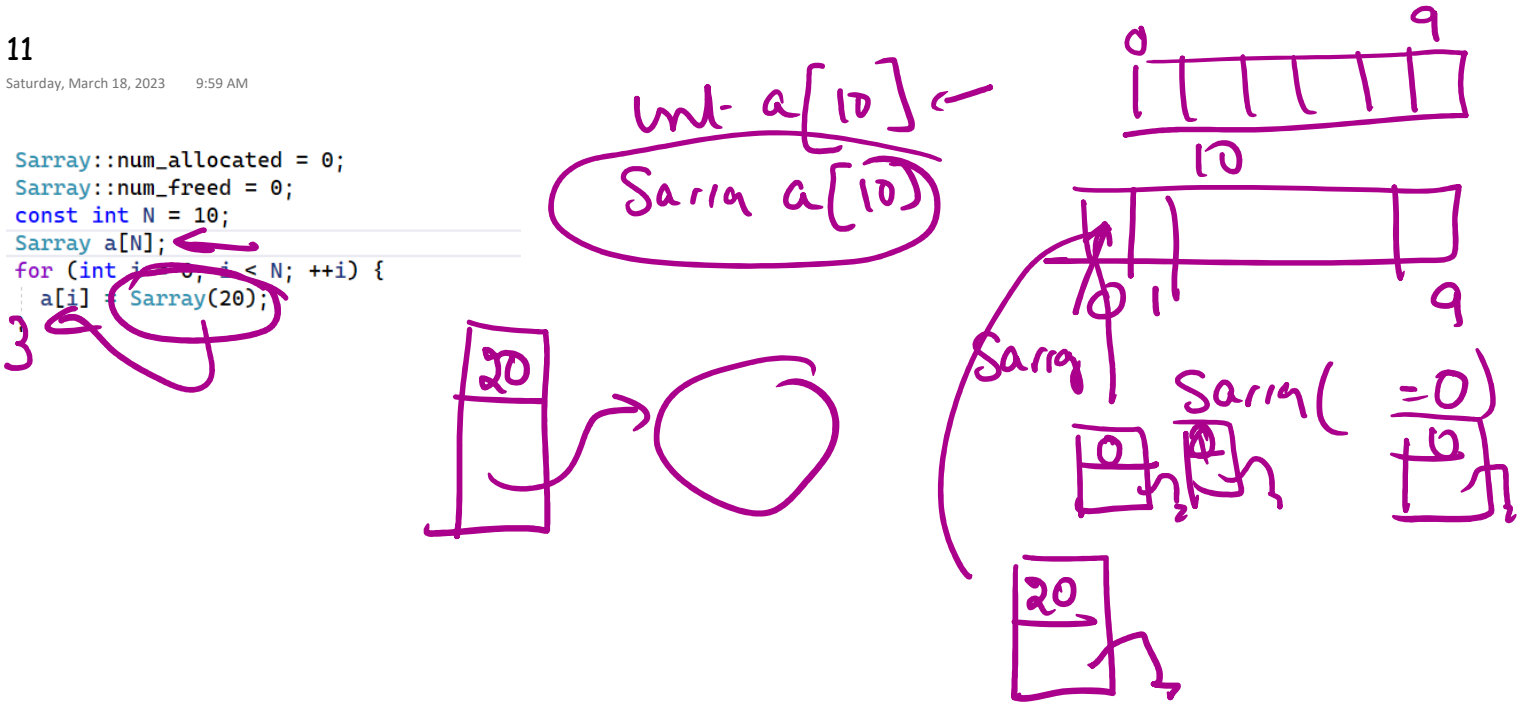
```
25      Sarray::num_freed = 0;
26      {
27          Sarray a(20);
28          Sarray b(40);
29          Swap(a, b);
30      }
```



11

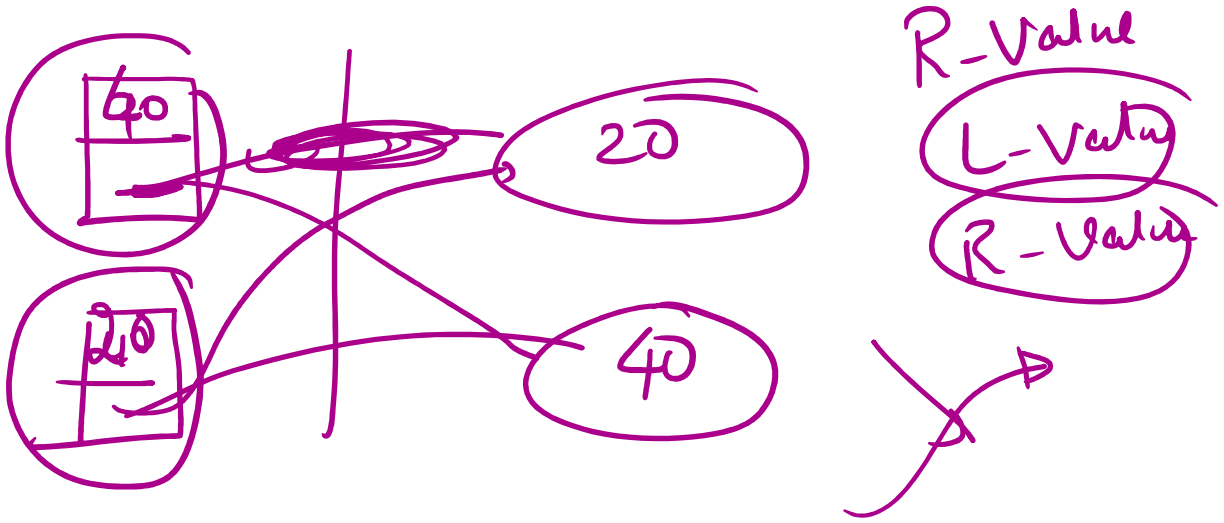
Saturday, March 18, 2023 9:59 AM

```
Sarray::num_allocated = 0;
Sarray::num_freed = 0;
const int N = 10;
Sarray a[N];
for (int i = 0; i < N; ++i) {
  a[i] = Sarray(20);
}
```



12

Saturday, March 18, 2023 10:11 AM



```cpp
Sarray::Sarray(const Sarray& rhs) {
  if (rhs.show) {
    cout << "calling copy constructor of size " << rhs.size_ << endl;
  }
  copy_(rhs);
}
```

```cpp
void Swap(Sarray& a, Sarray& b) {
  if (0) {
    //OLD WAY
    Sarray t(a);
    a = b;
    b = t;
  } else {
    Sarray t = std::move(a); //If you don't write move constructor, this calls copy constructor
    a = std::move(b); //If you don't write move equal operator, this calls equal operator
    b = std::move(t);
  }
}
```

```cpp
#ifdef MOVE
Sarray::Sarray(Sarray&& rhs) {
  if (rhs.show) {
    cout << "calling move constructor of size " << rhs.size_ << endl;
  }
  move_(rhs);
}
```

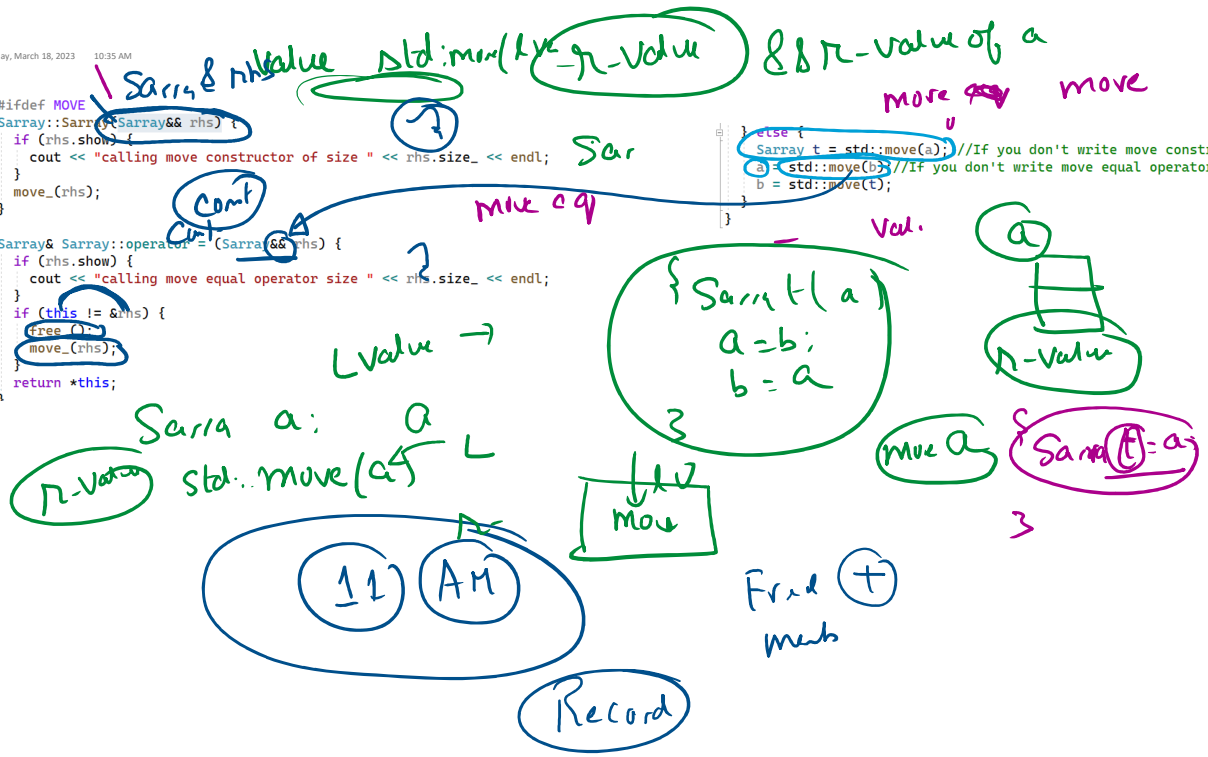
```cpp
{
  Sarray a(20);
  Sarray b(40);
  Swap(a, b);
}
```

```cpp
void copy_(const Sarray& rhs) {
  size_ = rhs.size_;
  alloc_(rhs.size_);
}

void move_(Sarray& rhs) {
  size_ = rhs.size_;
  ptr_ = rhs.ptr_; //steal the pointer
  rhs.ptr_ = nullptr;
  rhs.size_ = 0;
}
};
```

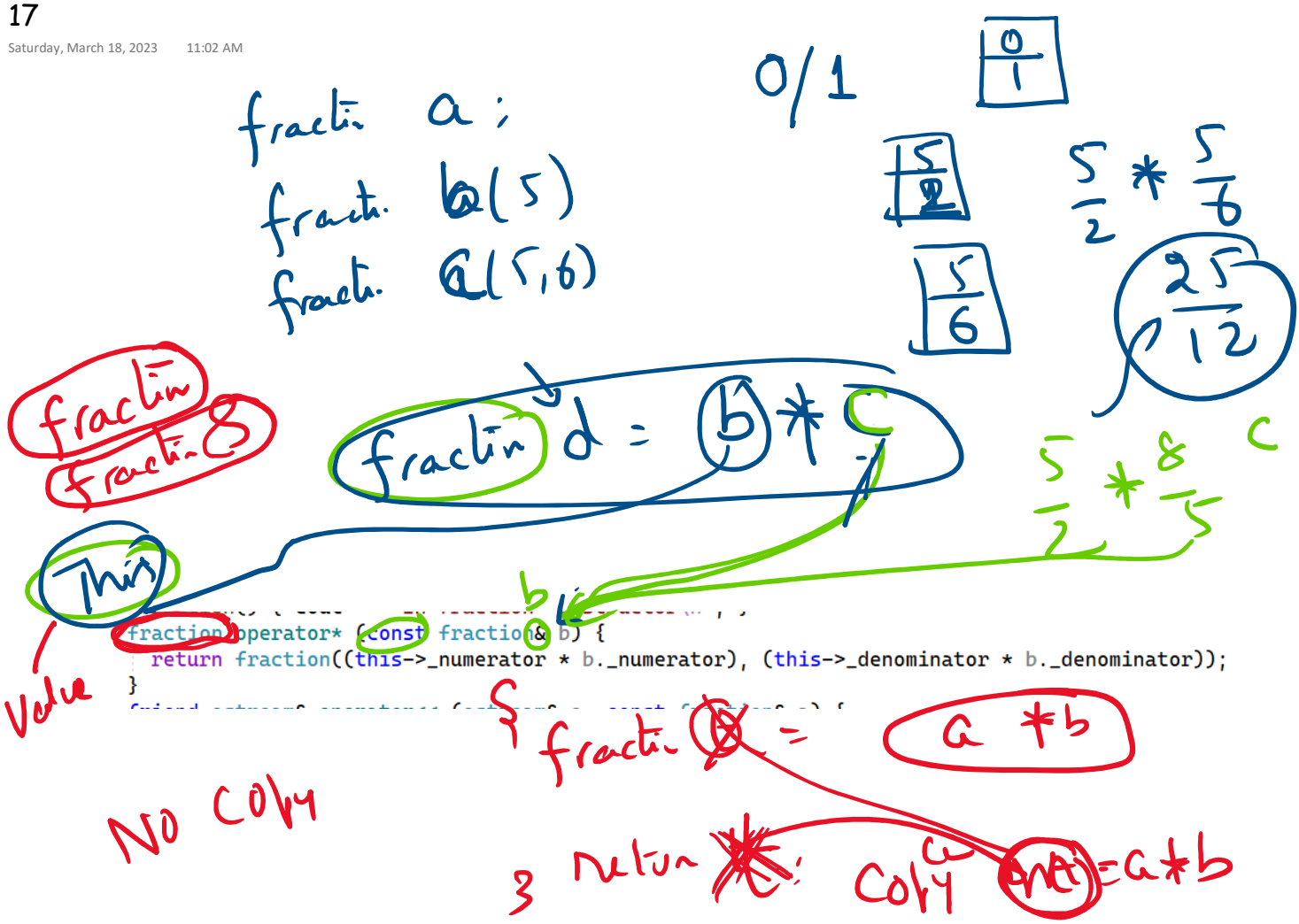
Saturday, March 18, 2023 10:35 AM

```cpp
#ifdef MOVE
Sarray::Sarray(Sarray&& rhs) {
  if (rhs.show) {
    cout << "calling move constructor of size " << rhs.size_ << endl;
  }
  move_(rhs);
}

Sarray& Sarray::operator = (Sarray&& rhs) {
  if (rhs.show) {
    cout << "calling move equal operator size " << rhs.size_ << endl;
  }
  if (this != &rhs) {
    free();
    move_(rhs);
  }
  return *this;
}
```

```cpp
  } else {
    Sarray t = std::move(a);//If you don't write move constructor, this calls copy constructor
    a = std::move(b);//If you don't write move equal operator, this calls equal operator
    b = std::move(t);
  }
}
```



Saturday, March 18, 2023 11:02 AM



18

Saturday, March 18, 2023 11:32 AM

```
friend fraction operator* (const fraction& a, const fraction& b) {
  if (1) {
    return fraction((a._numerator * b._numerator),(a._denominator * b._denominator)) ;
  }else {
    fraction t((a._numerator * b._numerator), (a._denominator * b._denominator));
    return t;
  }
}
```

this

1,

fract. a(1,3)

b(2,8)

c = a*b

1/3 * 2/8 = 2/24

Member fun

1 1

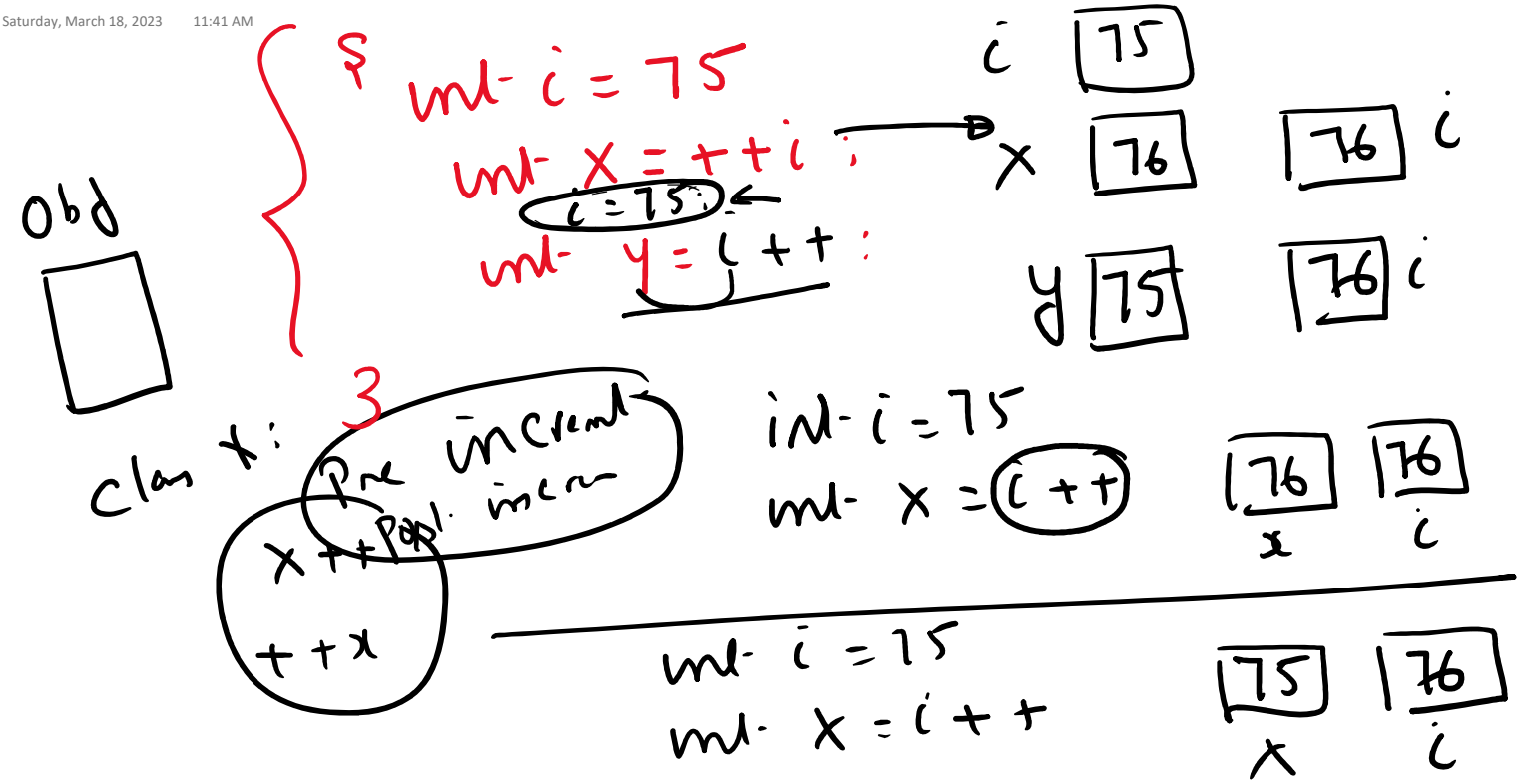
1 2

2 3

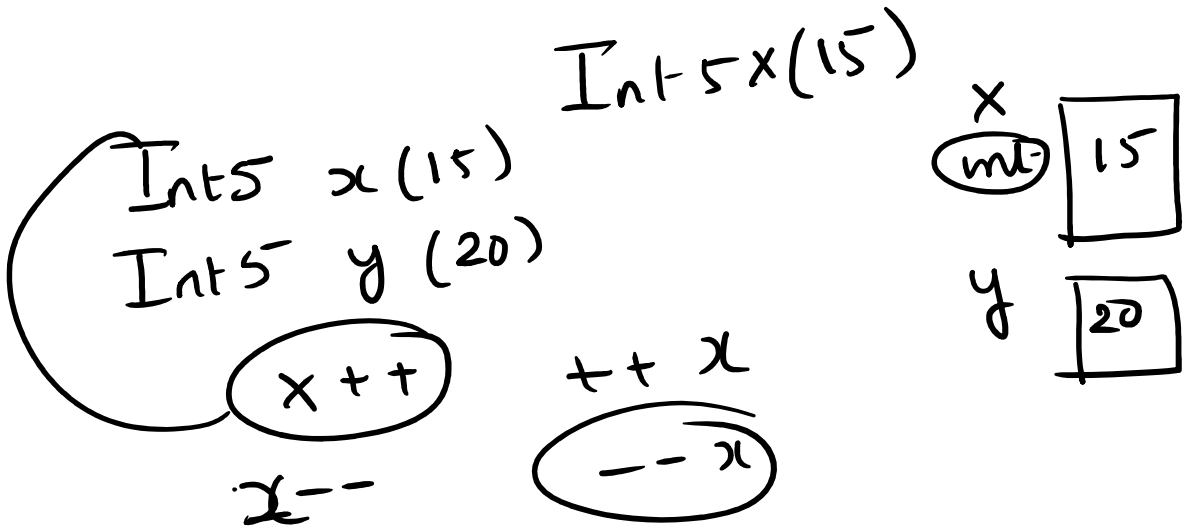
this a (1)

```
fraction operator* (const fraction& b) {
  return fraction((this->_numerator * b._numerator), (this->_denominator * b._denominator));
}
```

Saturday, March 18, 2023 11:41 AM



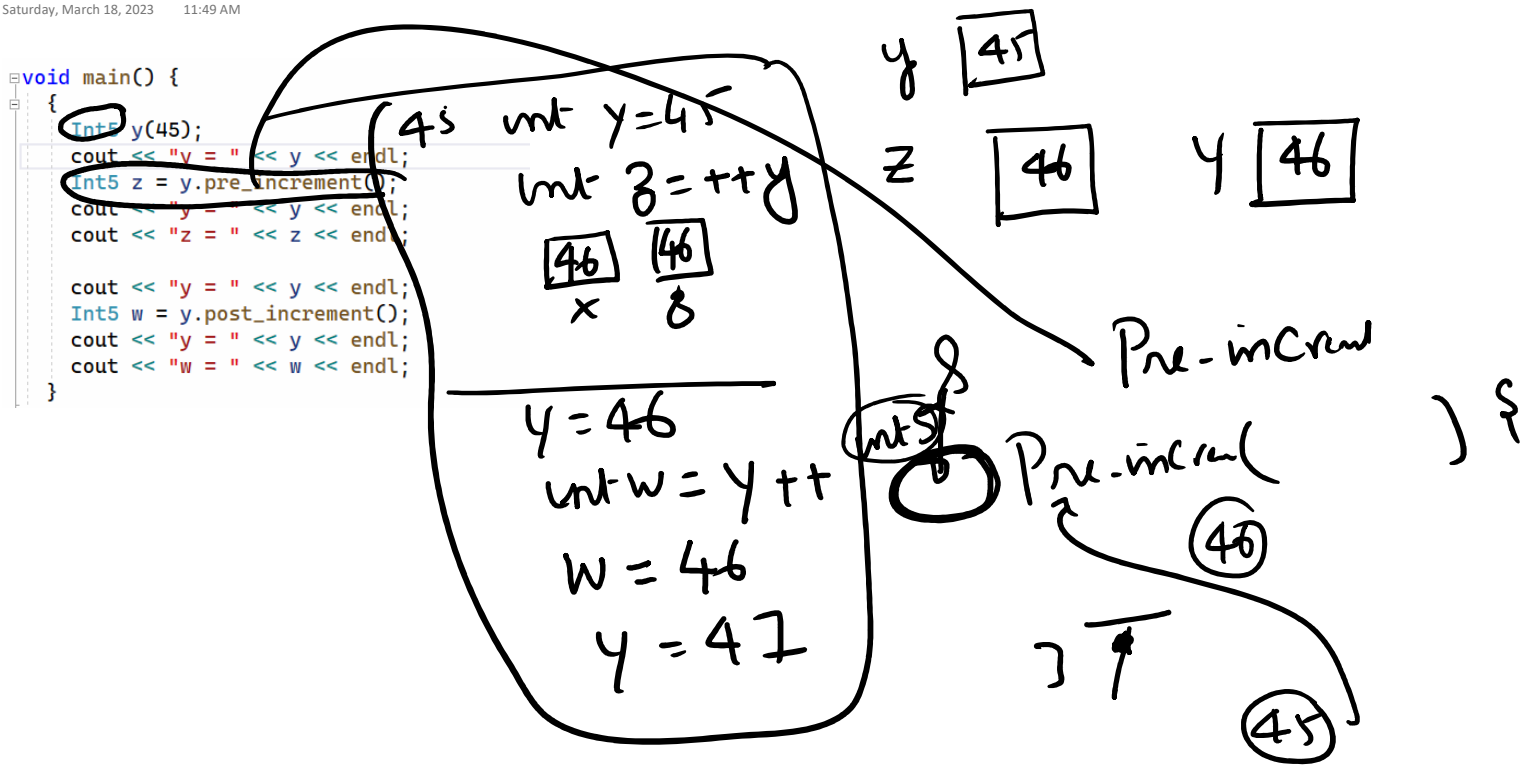
Saturday, March 18, 2023 11:46 AM



21

Saturday, March 18, 2023 11:49 AM

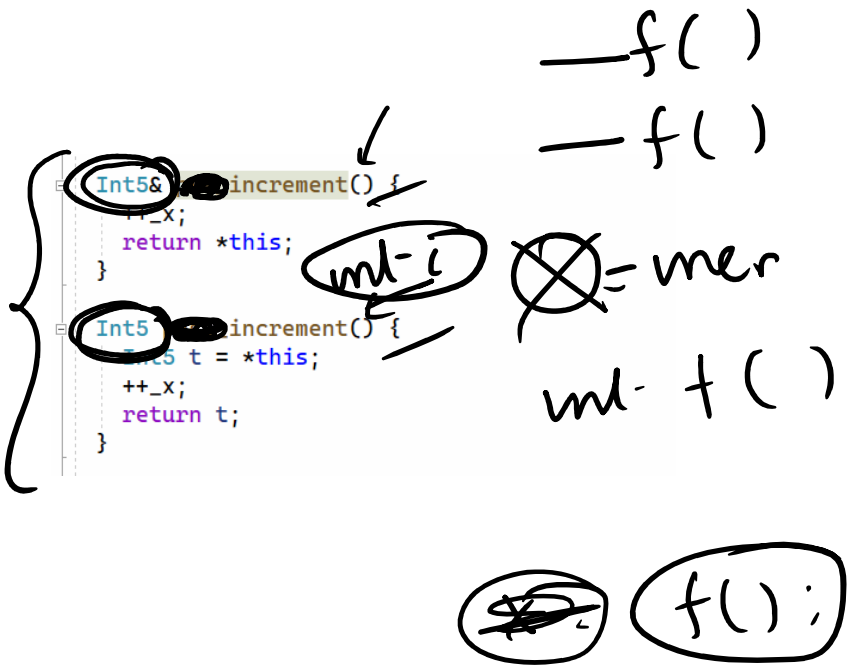
```
void main() {
    {
    Int5 y(45);
    cout << "y = " << y << endl;
    Int5 z = y.pre_increment();
    cout << "y = " << y << endl;
    cout << "z = " << z << endl;

    cout << "y = " << y << endl;
    Int5 w = y.post_increment();
    cout << "y = " << y << endl;
    cout << "w = " << w << endl;
    }
```



22

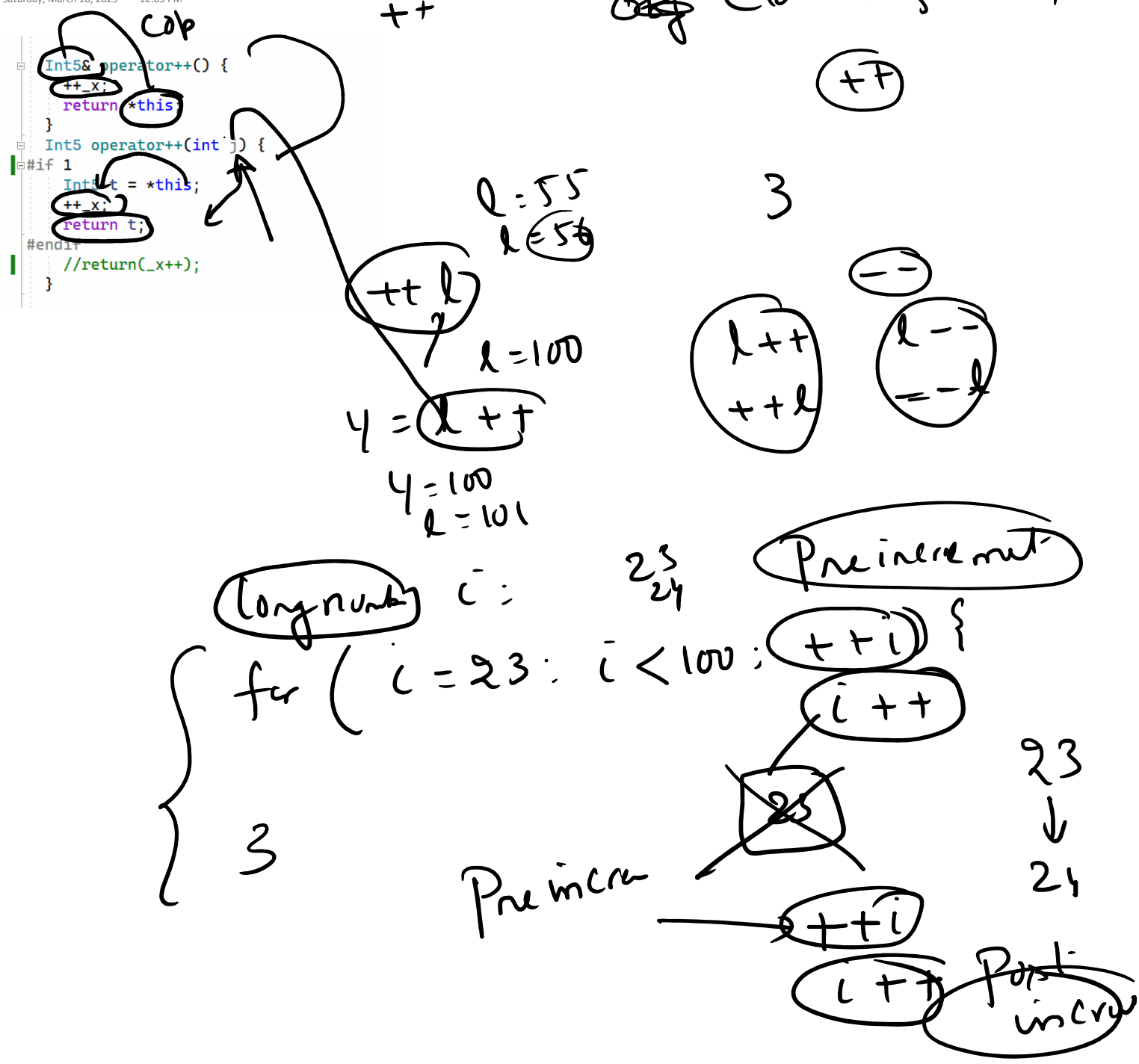
Saturday, March 18, 2023 11:58 AM

```
Int5& pre_increment() {
  ++_x;
  return *this;
}

Int5 post_increment() {
  Int5 t = *this;
  ++_x;
  return t;
}
```



Saturday, March 18, 2023 12:05 PM

```
Int5& operator++() {
  ++_x;
  return *this;
}
Int5 operator++(int j) {
#if 1
  Int5 t = *this;
  ++_x;
  return t;
#endif
  //return(_x++);
}
```



Class Obj { ++ -- }

Private:
int -x;

3

Obj a;
Obj b(5);
Obj *c = new Obj(8)
b = 5

x=5 b=6 int x = b++;
y=6 b=5 int y = b--;
z=6 b=6 int z = ++b;
k=5 b=5 int k = --b

int x = c++
c--
++c
--c

{
int a = 5;
Obj b(5);
int x = a++;
Obj.y = b++;
3 → if(x != y) {
→ Assert(F
3

bool(Obj, int. ++ --
→ 70-1.
TB: 30-1.