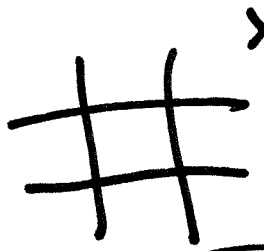
Create a class to represent a game - tic tac toe




```
class tictactoe
{   public:
      tictactoe();
      ~tictactoe();
    void displaybrd();
    void selectpos();
    private:
      board game;
      player plyrs[2];
};
```

```
struct row
{   char col[3];
};
row brd[3];
```

```
struct board
{   char row1[3];
    char row2[3];
    char row3[3];
};
```

```
struct board
{   char pieces[3][3];
};
```

create a class for a library

- books
- CDs
- Movies

ISBN
Title
Author(s)
Search key
Abstracts

```
class library
{
  public:
    library();
    ~library();
    library(int);
  private:
    book * allbooks;
    group * all;
    int num_groups;
};
```

```
struct group
{
  book * allbooks;
  char cat[41];
  int num;
  int size_array;
};
```

other functions →

```
struct Author
{
  char name[41];
};
```

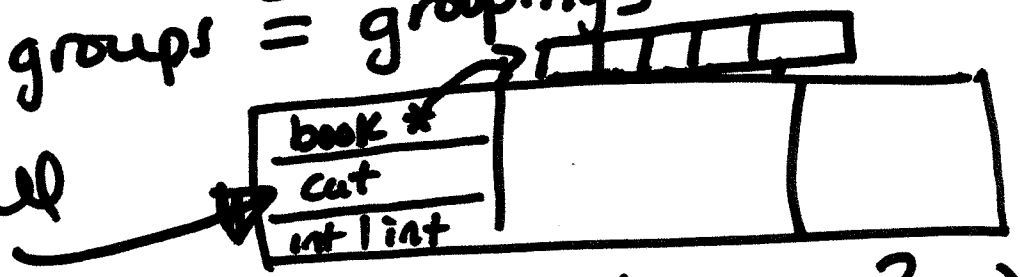
```
class book
{
  public:
    book();
    ~book();
    void set(book&);
    void input();
    void display();
    bool compare(char[]);
  private:
    char ISBN[81];
    char * title;
    Author *fullname;
};
```

Extending the book

void set(char [], char[], Author []);

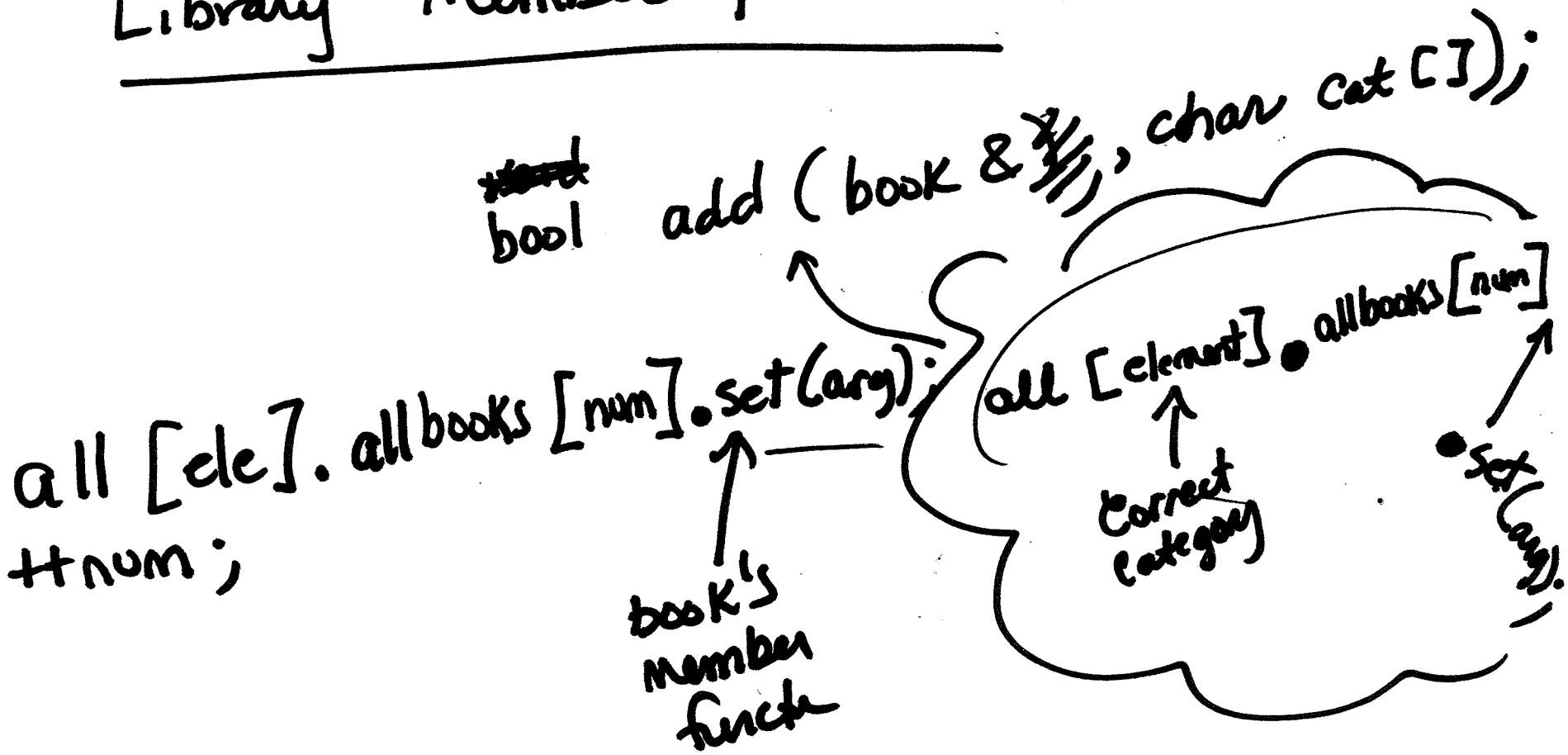
→ ~~void edit(char~~
void editisbn(char []);

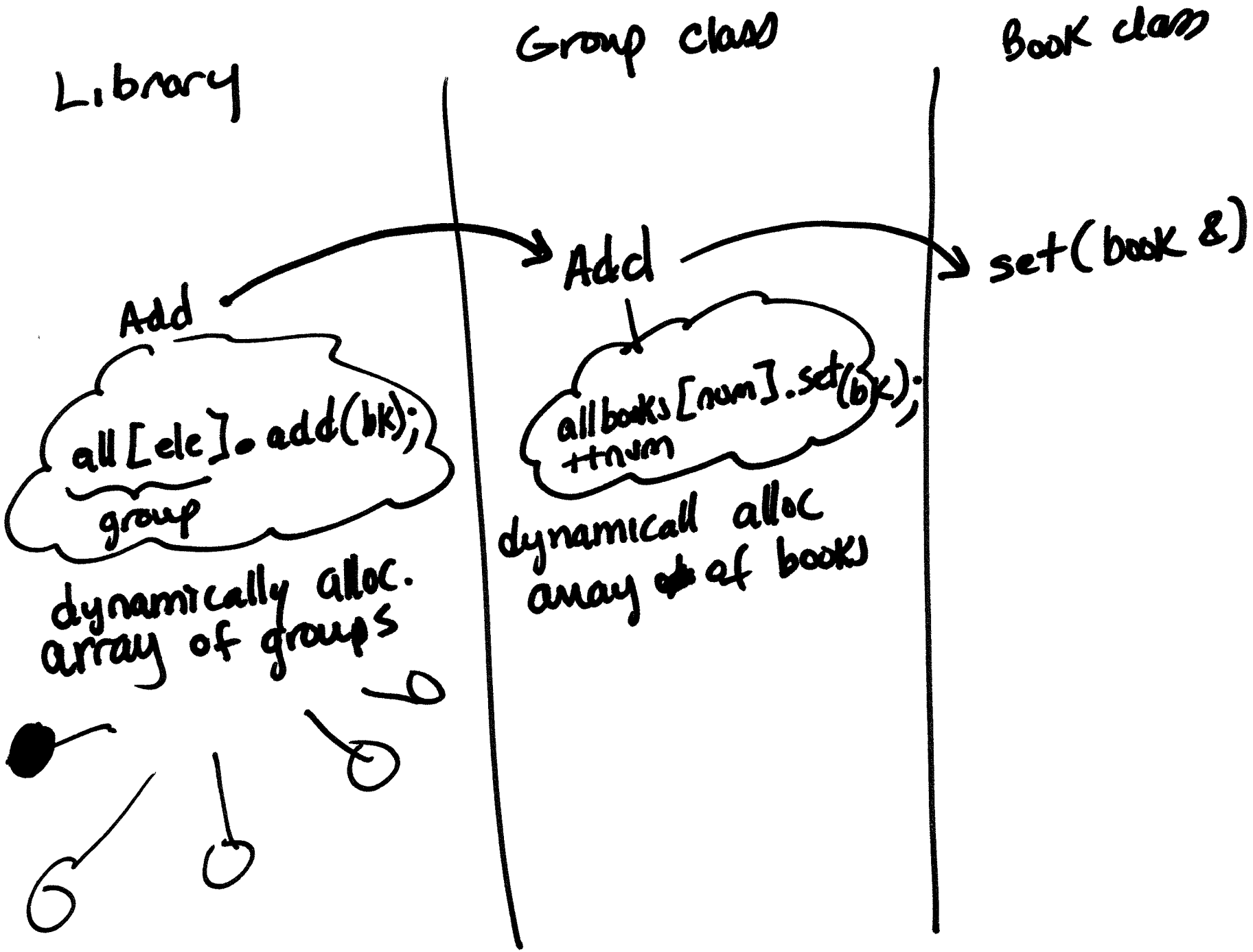
```
library::library(int groupings)
{
    all = new group[groupings];
    num_groups = groupings
```



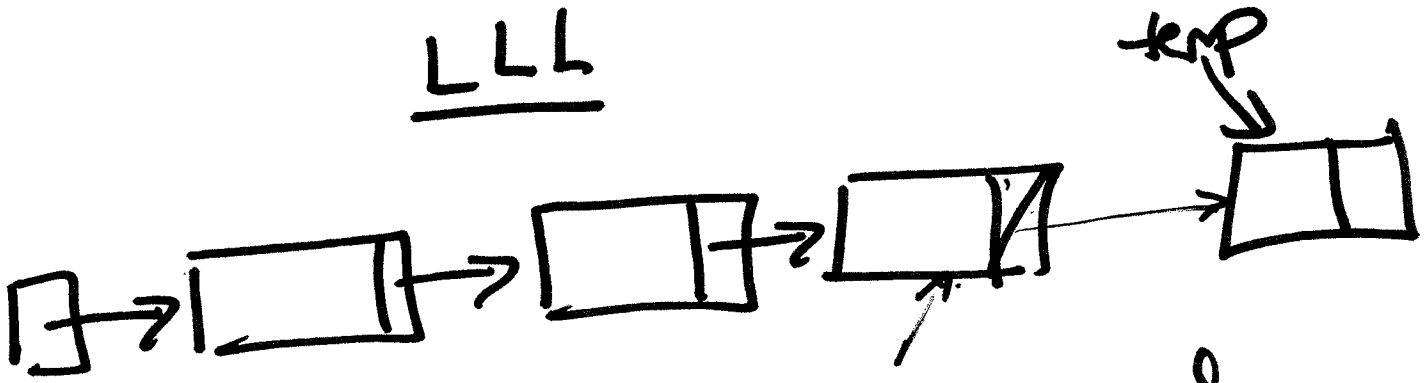

```
for(int i = 0; i < num_groups; ++i)
{
    ~~allbooks~~[
    all[i]. allbooks = new book[MAX];
    size_array = MAX;
    cat[0] = '\0';
    num = 0;
}
```

Library Member functions




Library
Group class
Book class
Add
Add
set(book &)
all[ele].add(bk);
group
dynamically alloc.
array of groups
allbooks[num].set(bk);
++num
dynamicall alloc
array of books

LLL



Basics

- allocating
- deallocating

Examining code

```
current = head;
while ( current)
{     previous = current;
      current = current->next;

}
previous->next = head;
```

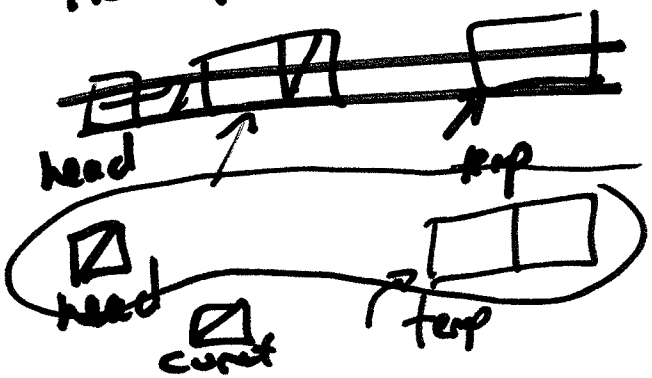
① What does this do? → it made the last node pt to the first

② previous->next = temp;

What is Wrong?

head is Never connect ☺ First

Traversal

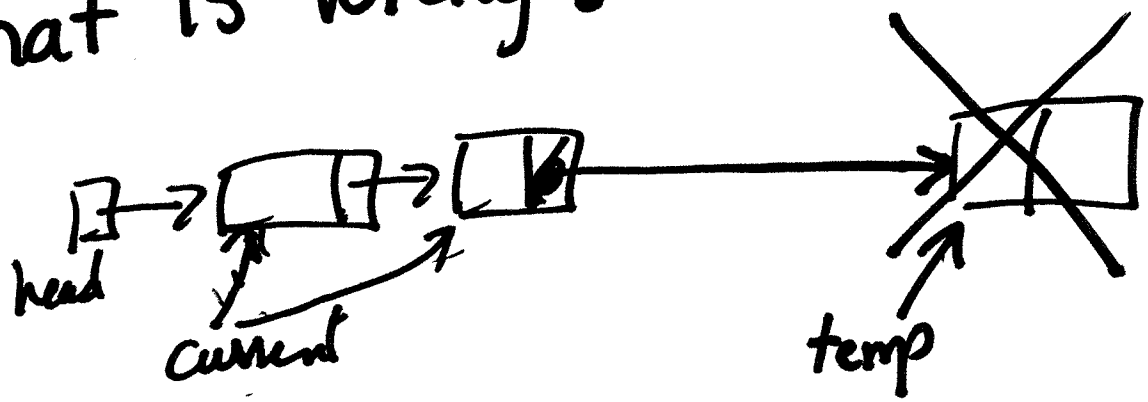


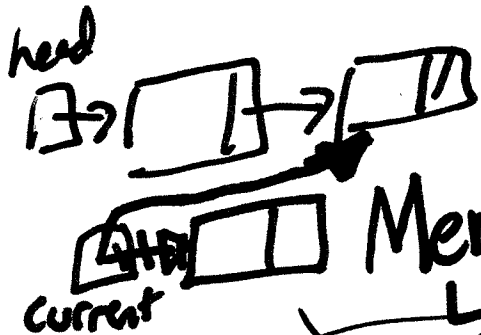
```
if(head)
{   temp = new Node;
    current = head;
    while (current->next)
    {     current= current->next;
    }
    current->next = temp;
```

NO!

```
delete temp;
```

What is Wrong?

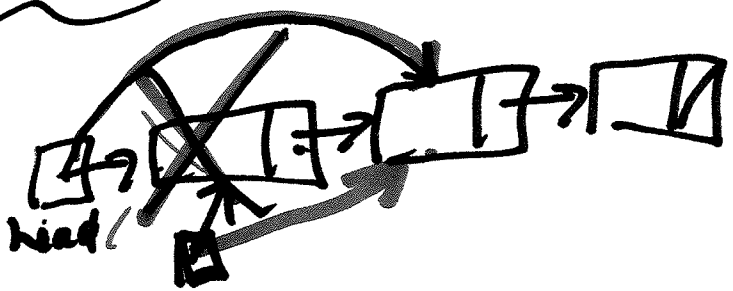





```
list::~list()
{
    node * current = new node;
    while (head)
    {
        current = head->next;
        delete head;
        head = current;
    }
    head = NULL;
}
```

① only use new to allocate memory!

② Redundant



hw 3

1. `int * ptr;`
2. `ptr = NULL;`
3. Statically allocated array vs dynamically

BOTH FIXED SIZE

The size is known at Compile time

The size is specified as a constant

Wait until runtime to allocate the memory

Same

"Using" is the same

`array[index]`

`*(array+index)`

4.

```
char temp[101];
cin.get(temp, 101, '\n');
cin.ignore(100, '\n');

char *name;
name = new char[strlen(temp)+1];
strcpy(name, temp);
```

---

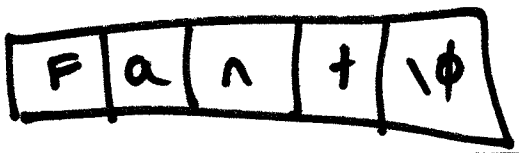
4 →

```
length = strlen(temp);
```

"Fart"

```
name = new char[length+1];
```

5



```
char *name = new char[5];
```

5. delete [] name;

deallocate the memory name is pointing to.... if it had been an array of class objects it will implicitly call each object's destructor.

6. int * ptr;

ptr = new int [50];

int num = 0

← how many integers are actually in this array

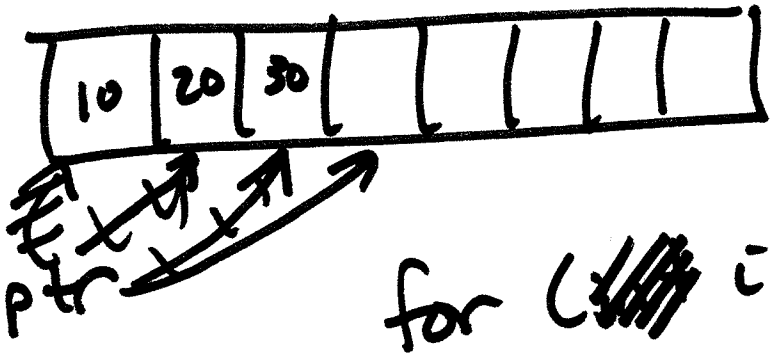
6a. (wrong)

```
for (int i = 0; i < num; ++i)
    cout << ptr[i] << endl;
```

up to 50

ptr[i] ≡ *(ptr + i)

i * sizeof(int) → offset address



temp
sum

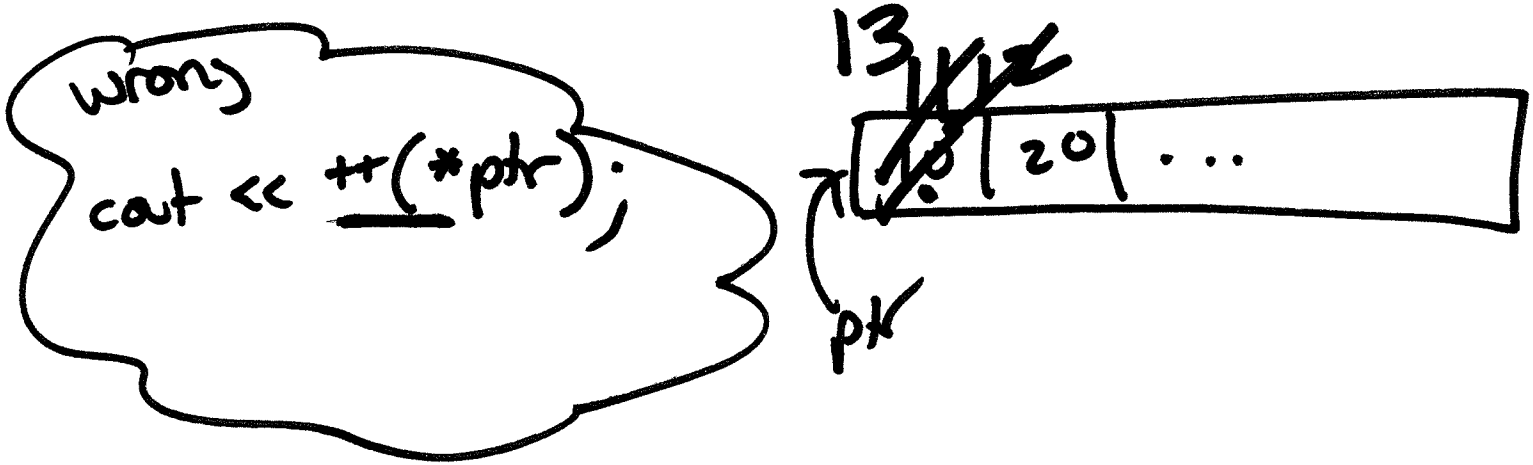
```
for ( i = 0; i < num; ++i)
{
    cout << *ptr << endl;
    ++ptr;
}
```

ptr -= i;

```
for (i = 0; i < num; ++i)
    cout << *ptr++ << endl;
```

`*ptr++`

```
--ptr;
for (. . . . .)
    cout << *(++ptr) << endl;
```



a[i] == *(a+i)

```
for (int i=0; i < num; ++i)
    cout << [redacted] *(a+i);
```