

# POINTER ARITHMETIC AND MIDTERM REVIEW

---

Problem Solving with Computers-I

<https://ucsb-cs16-wi17.github.io/>

C++

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

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



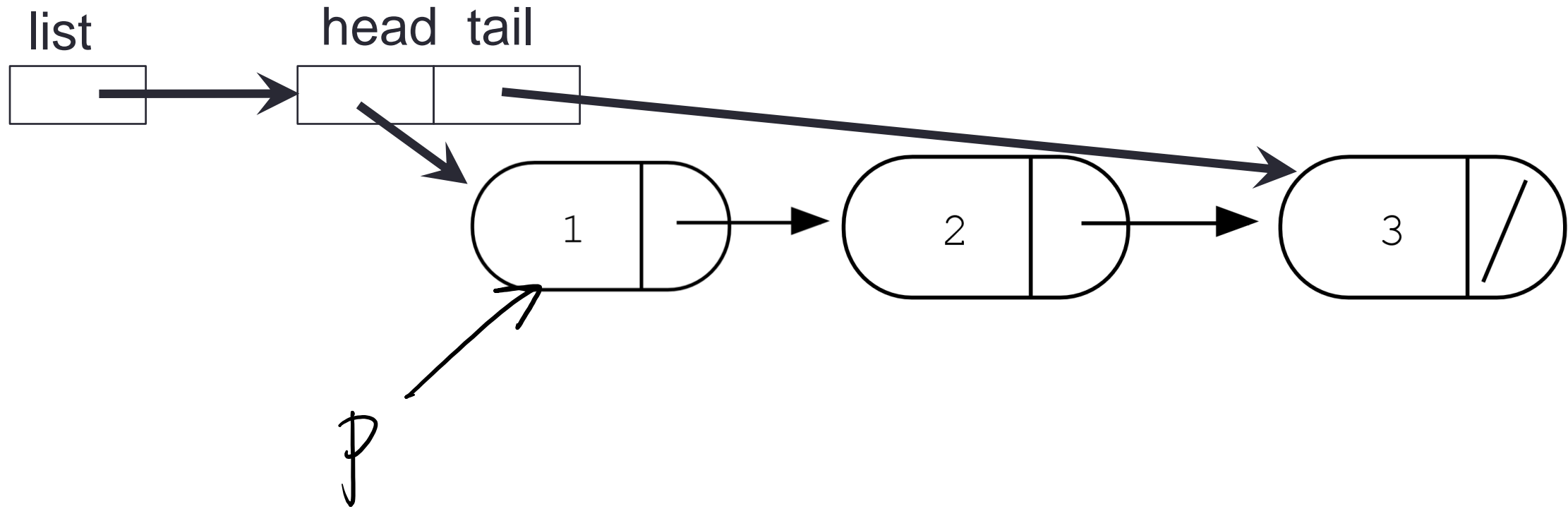
# How to study for the midterm?

- Go to the site for midterm 2: <https://ucsb-cs16-wi17.github.io/exam/e02/>
- Review hws, labs and lectures
- Create your own notes and contribute to the collaborative notes document: <https://docs.google.com/document/d/1ctpQAIAiTz5L8m8m9ibGIQG9Jcir4xmIUfiknvRR0wk/edit?usp=sharing>
- Solve the problems in our concept inventory: <https://drive.google.com/drive/folders/0B1z9k2M7uTvJaE9rR0F0Ovv5ZWws?usp=sharing>
- Ask questions!
- Midterm review sessions: Today 6pm to 8pm HFH 1152, Wednesday (during your section), Thursday morning 9am to 11am (during Diba's office hours)

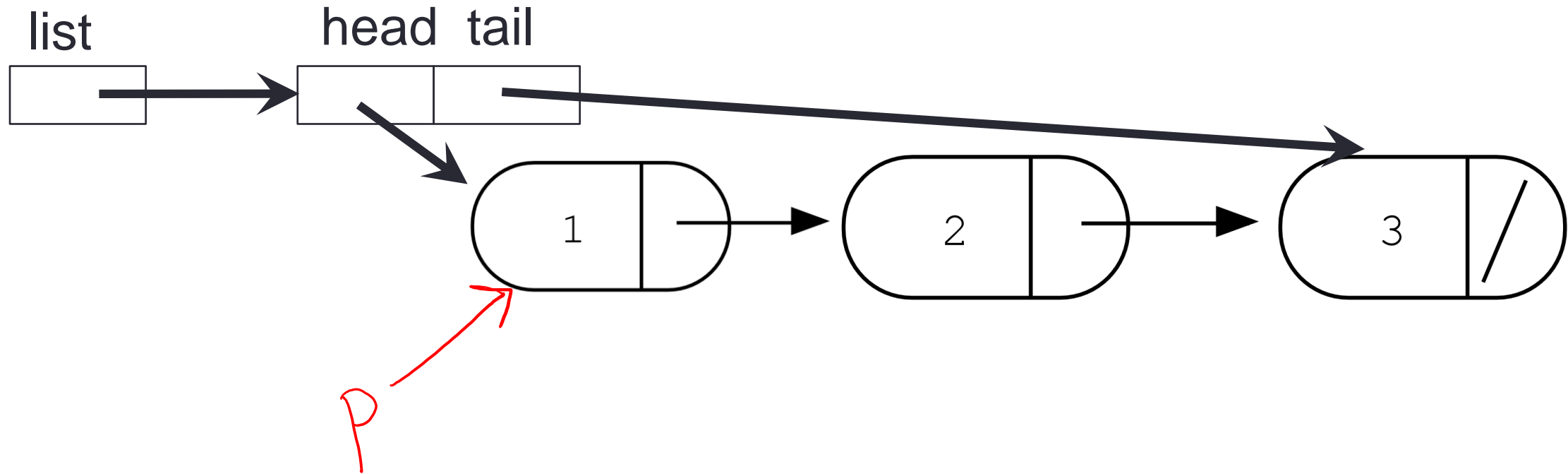
## Pointer diagram;

```
int a=6, b=7, *p1=&b, *p2=&a;  
p1 = p2;  
*p1 = 8;  
p2 = &b;
```

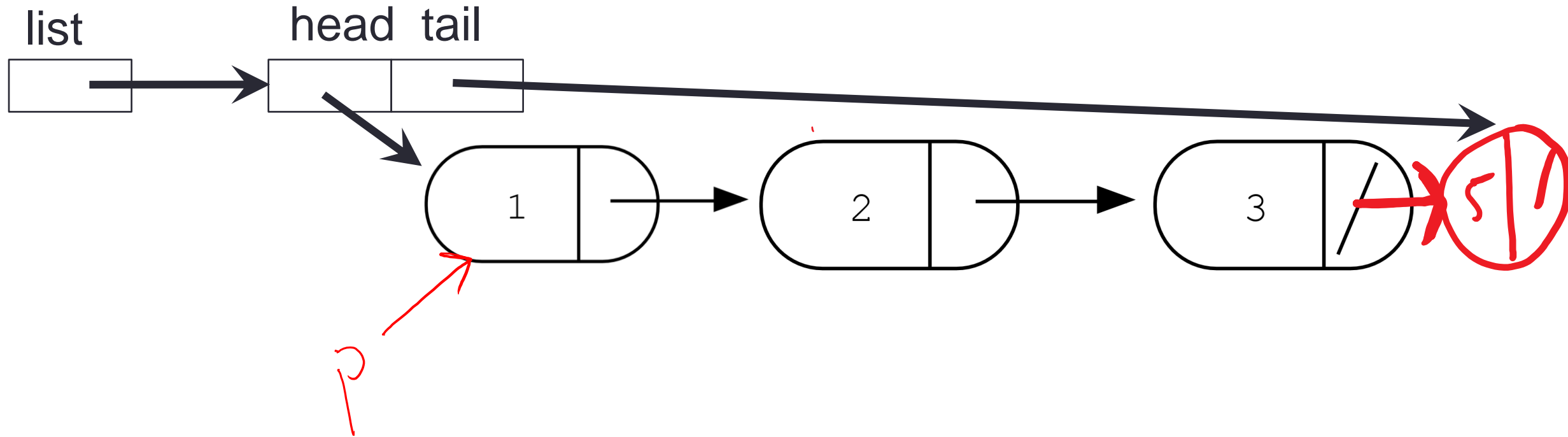
# Deleting node 2 in the list



# Deleting node 2 in the list



# Deleting a node in the list



# Pointer Arithmetic

```
int arr[]={50, 60, 70};
```

```
int *p;
```

```
p = arr;
```

```
p = p + 1;
```

```
*p = *p + 1;
```

```
UndergradStudents records[2];
```

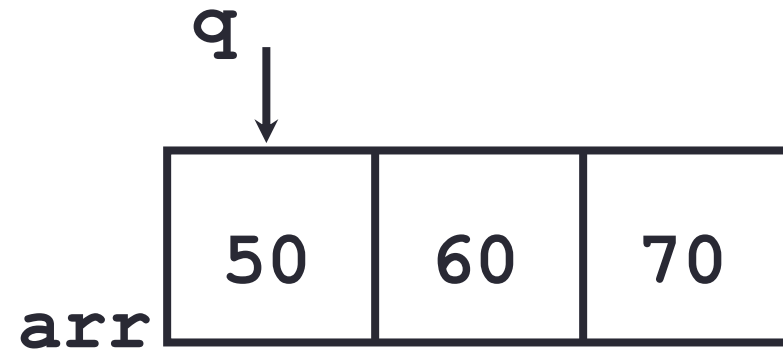
```
UndergradStudents *pRec;
```

```
pRec = records;
```

```
pRec = pRec + 1;
```

```
void IncrementPtr(int *p){  
    p = p + 1;  
}
```

```
int arr[3] = {50, 60, 70};  
int *q = arr;  
IncrementPtr(q);
```



Which of the following is true after **IncrementPtr (q)** is called in the above code:

- A. 'q' points to the next element in the array with value 60
- B. 'q' points to the first element in the array with value 50

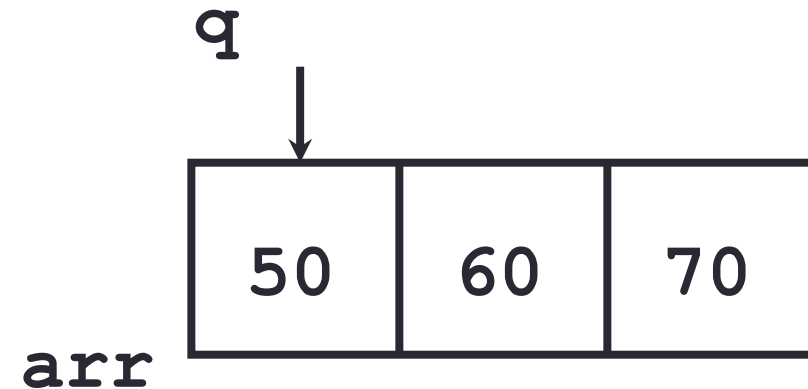


How should we implement `IncrementPtr()`, so that 'q' points to 60 when the following code executes?

```
void IncrementPtr(int **p){  
    p = p + 1; }  
}
```

```
int arr[3] = {50, 60, 70};  
int *q = arr;  
IncrementPtr(&q);
```

- A. `p = p + 1;`
- B. `&p = &p + 1;`
- C. `*p = *p + 1;`
- D. `p = &p + 1;`



# Pointer Arithmetic

- What if we have an array of large structs (objects)?
  - C++ takes care of it: In reality, `ptr+1` doesn't add 1 to the memory address, but rather adds the size of the array element.
  - C++ knows the size of the thing a pointer points to – every addition or subtraction moves that many bytes: 1 byte for a char, 4 bytes for an int, etc.

# Complex declarations in C/C++

How do we decipher declarations of this sort?

```
int **arr[];
```

Read

- \* as “pointer to” (always on the left of identifier)
- [] as “array of” (always to the right of identifier)
- ( ) as “function returning” (always to the right ...)

For more info see:

[http://ieng9.ucsd.edu/~cs30x/rt\\_lt.rule.html](http://ieng9.ucsd.edu/~cs30x/rt_lt.rule.html)

# Complex declarations in C/C++

## Right-Left Rule

```
int **arr [];
```

Step 1: Find the identifier

Step 2: Look at the symbols to the right of the identifier. Continue right until you run out of symbols \*OR\* hit a \*right\* parenthesis ")"

Step 3: Look at the symbol to the left of the identifier. If it is not one of the symbols '\*', '(', '[' just say it. Otherwise, translate it into English using the table in the previous slide. Keep going left until you run out of symbols \*OR\* hit a \*left\* parenthesis "(".

Repeat steps 2 and 3 until you've formed your declaration.

Illegal combinations include:

[]() - cannot have an array of functions

()() - cannot have a function that returns a function

()[] - cannot have a function that returns an array

# Complex declarations in C/C++

```
int i;  
int *i;  
int a[10];  
int f( );  
int **p;  
int (*p)[];  
int (*fp)( );  
int *p[];  
int af[]( );  
int *f();  
int fa()[];  
int ff()();  
int (**ppa)[];  
int (*apa[ ])[ ];
```

# Next time

- Recursion