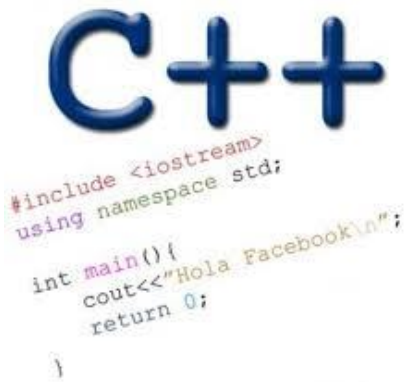
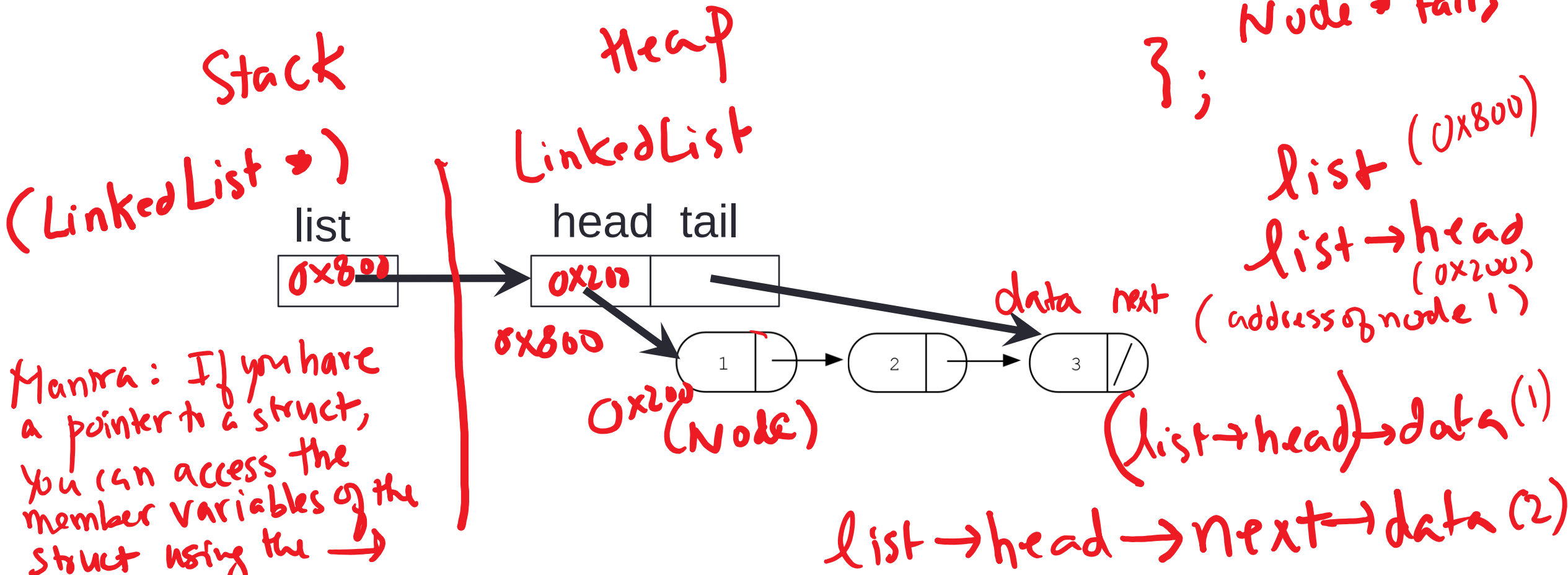
LINKED LISTS (CONTD)
DYNAMIC ARRAYS

Problem Solving with Computers-I

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

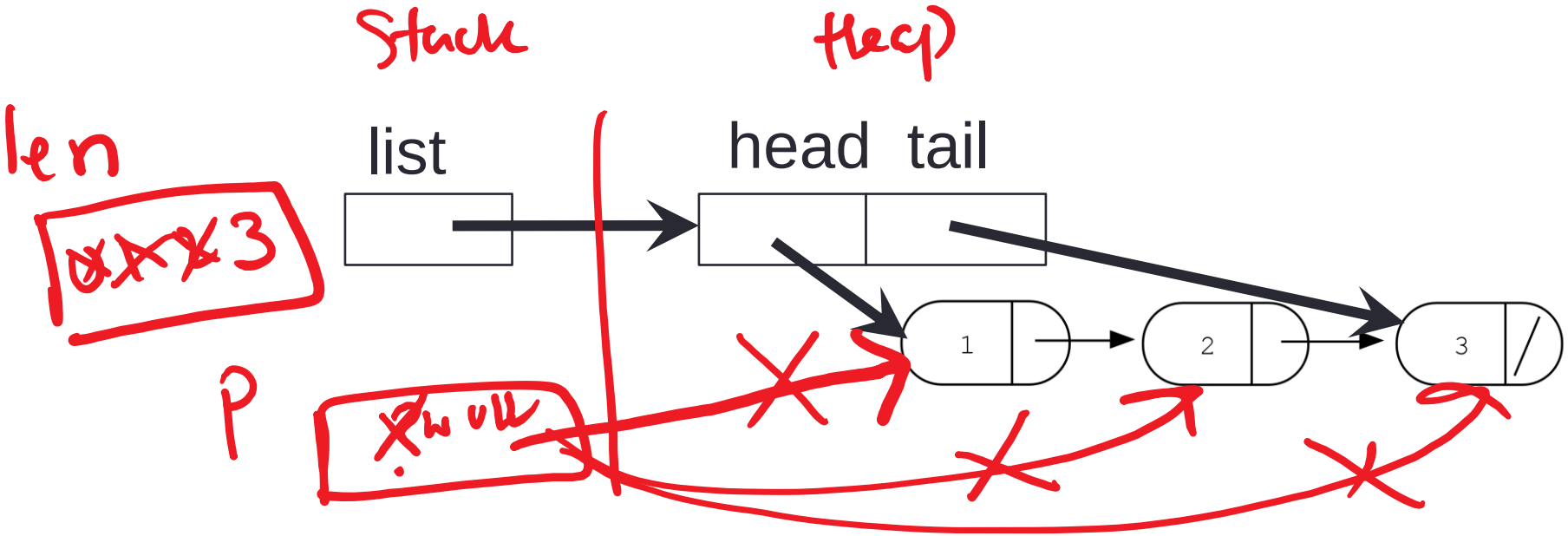


Review:
What are the 'links' in a linked-list?



Iterating through the list

```
int lengthOfList(LinkedList * list) {
        assert(list);
        int len =0;
        Node *p;
        for(p= list->head;    p !=NULL   ; p= p->next)
              len++;
         return len;
}
```

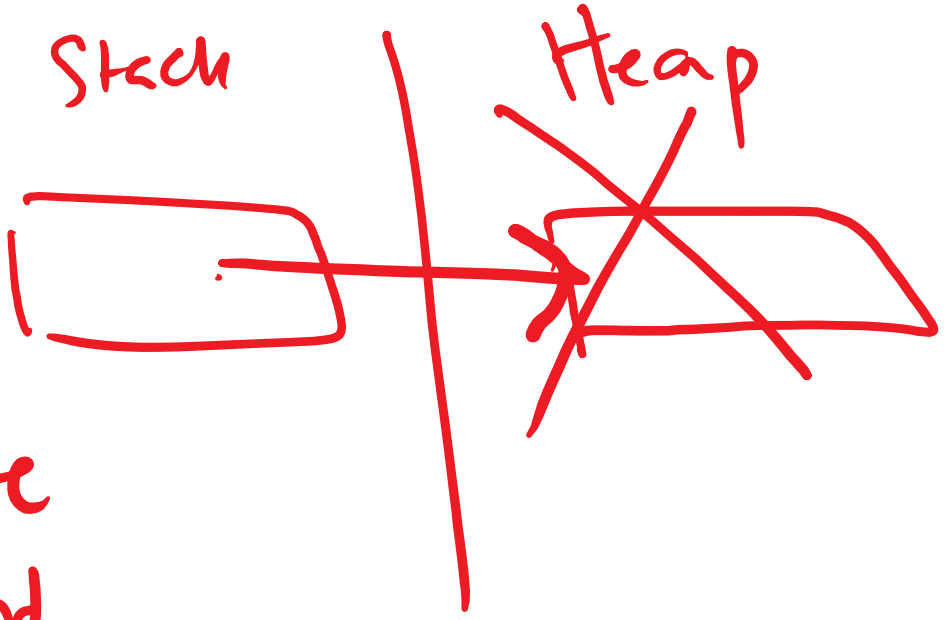


Dynamic memory pitfall: Memory Leaks

- Memory leaks (tardy free)
  - Heap memory not deallocated before the end of program (more strict definition, potential problem)
  - Heap memory that can no longer be accessed (definitely a leak , must be avoided!)

Does calling foo() result in a memory leak?

```
void foo(){
        int * p = new int;
        delete p;//deletes what p is pointing to
}
```

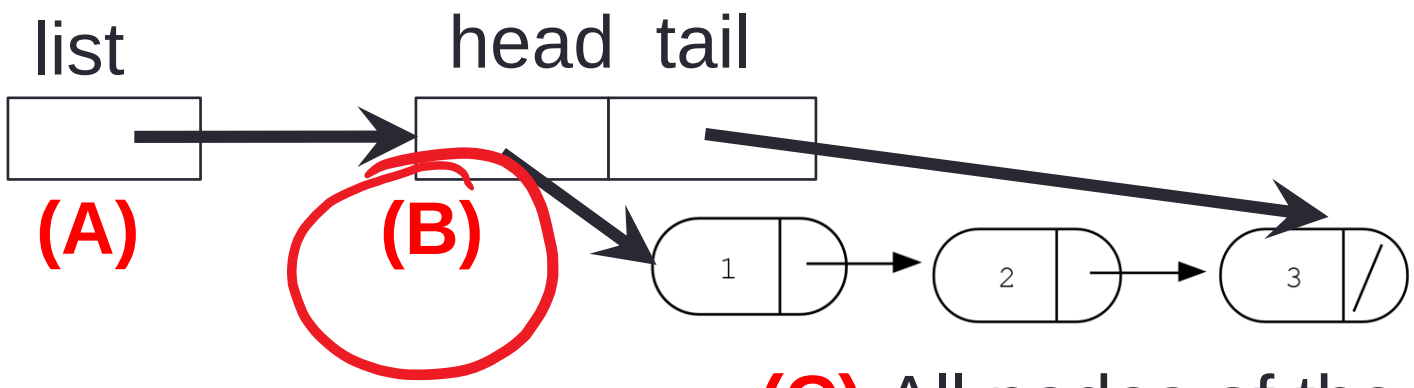


- How to avoid memory leaks? delete
- How to detect memory leaks? valgrind

Deleting the list

int freeLinkedList(LinkedList * list){…}

Which data objects are deleted by the statement: delete list;



(C) All nodes of the linked list

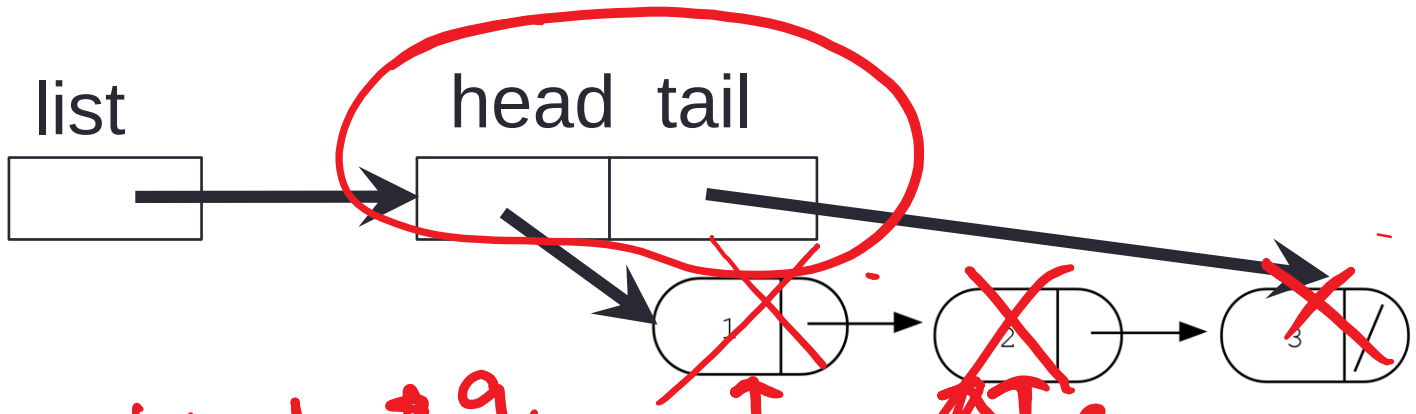
(D) B and C

(E) All of the above

Does this result in a memory leak?

Deleting the list

int freeLinkedList(LinkedList * list);



delete list→tail;

Node *q

q

p

Node *p = list→head;

while (p != NULL) {

q = p;

p = p→next;

delete q;

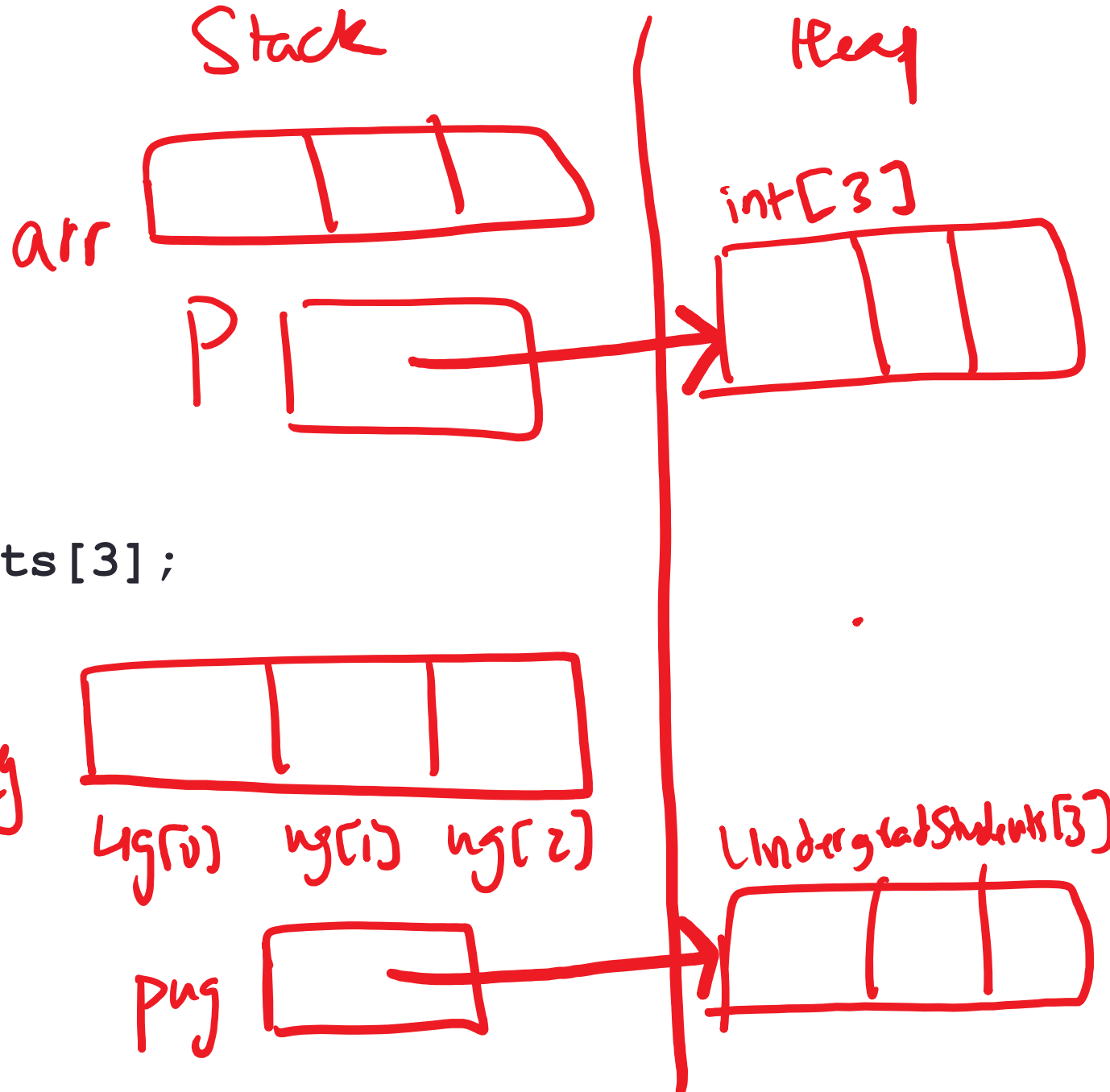
}

Dynamic arrays

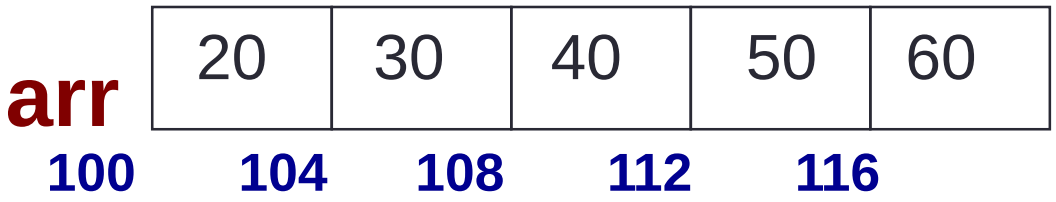
```
int arr[3];
int*p = new int[3];

UndergraduateStudents ug[3];
UndergraduateStudents *pug;
pug= new UndergraduateStudents[3];
```

```
struct UndergradStudents{
    string firstName;
    string lastName;
    string major;
    double gpa[4];
};
```



Arrays and pointers



cout<< arr;
will output 100
for example on left.

```
int arr[]= {20, 30, 40, 50, 60}
```

- `arr` is a pointer to the first element, what is the output of `cout<<arr;`
- `arr[0]` is the same as `*arr`
- `arr[2]` is the same as `*(arr+2)`

- Use pointers to pass arrays in functions (See code from lecture 9)
- Use *pointer arithmetic* to access arrays more conveniently

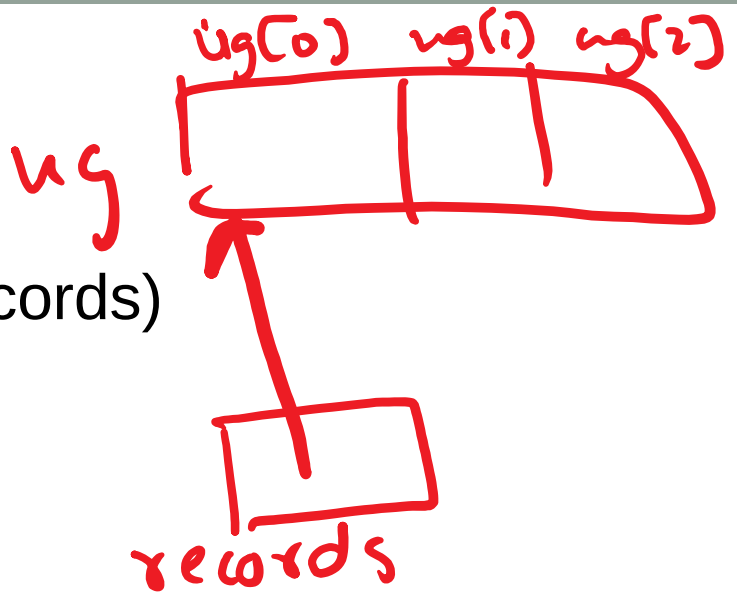
Review of homework 10, problem 5

```
void printRecords(UndergradStudents records [], int numRecords);
int main(){
    UndergradStudents ug[3];
    ug[0] = {"Joe", "Shmoe", "EE", {3.8, 3.3, 3.4, 3.9} };
    ug[1] = {"Macy", "Chen", "CS", {3.9, 3.9, 4.0, 4.0} };
    ug[2] = {"Peter", "Patrick", "ME", {3.8, 3.0, 2.4, 1.9} };
    printRecords(ug, 3);
}
```

Expected output

These are the student records:
ID# 1, Shmoe, Joe, Major: EE, Average GPA: 3.60
ID# 2, Chen, Macy, Major: CS, Average GPA: 3.95
ID# 3, Pan, Patrick, Major: ME, Average GPA: 2.77

Review of homework 10, problem 5




```
void printRecords(UndergradStudents records [], int numRecords)
{
    double avgGPA;
    for(int i=0; i< numRecords; i++){
        cout<< "ID#" <<i <<", " <<records[i].lastName <<", "
            << records[i].firstName << " Avg GPA:" << avgGPA <<endl;
    }
}
```

same as
records

```
avgGPA = 0
for ( int j=0; j<4; j++ )
    avgGPA += records[i].gpa[j];
```

same as

```
*(records+i).gpa[j]
```

evaluated using pointer arithmetic

Review of hw10, P5

```
void printRecords(UndergradStudents *records, int numRecords)
{
    double avgGPA;
    for(int i=0; i< numRecords; i++){
            avgGPA=0;
            for(int j=0; j< NUMGPA; j++){
                avgGPA += (records+i)->gpa[j];
        }
        cout<< "ID#" <<i <<", " <<records[i].lastName <<", "
            << records[i].firstName << " Avg GPA:" << avgGPA/4 <<endl;
    }
}
```

Next time

- Pointer arithmetic
- Complex C++ declarations
- Midterm Review
- More review during section