Q1. Pointer & Function Review (15 min) Write a function with the following header:

```
void sum(int* list1, int list1_size, int* list2, int list2_size);
```

`list1` and `list2` are two integer arrays representing numbers
`list1_size` is the number of digits in the first number
`list2_size` is the number of digits in the second number

`sum` prints the sum of the numbers.

Example:

```
int list1[] = { 8, 5, 3, 1 };
int list2[] = { 5, 3, 2, 9 };
sum(list1, 4, list2, 4);
// prints 13860, the sum of 8531 and 5329

int list3[] = {    5, 3, 1 };
int list4[] = { 5, 3, 2, 9 };
sum(list3, 3, list4, 4);
// prints 5860, the sum of 531 and 5329
```

Q2. (15 min) Write a function with the following header:

```
bool rangeSearch(int sorted_nums[], int n, int target,
                 int& start, int& end);
```

`sorted_nums` is an array of integers sorted in decreasing order
`n` is the number of elements in `sorted_nums`
`target` is a number to search for within `sorted_nums`

`rangeSearch` should return *true* if `target` is found in `sorted_nums` and *false* otherwise.

If `rangeSearch` returns *true*, `start` should be set to the first index where `target` appears and `end` should be set to the last index where `target` appears, so that the integers of indices from `start` to `end` should only contain `target`.

If `rangeSearch` returns *false*, `start` and `end` should not be altered.

Example:

```
int foo[7] = { 5, 4, 3, 3, 1, -2, -3 };
```

```
int s = 21;
int e = 14;
rangeSearch(foo, 7, 0, s, e); // returns false, s remains 21, e remains 14
rangeSearch(foo, 7, 3, s, e); // returns true, now s == 2 and e == 3
```

Q3. Classes (15 min) Write a class called Complex, which represents a complex number. Complex should have a default constructor and the following constructor: (Michelle Lee)

```
Complex(int real, int imaginary);
// -3 + 8i would be represented as Complex(-3, 8)
```

Additionally, the class should contain two member functions: sum and print. Calling sum should set the calling object to the sum of the 2 complex numbers passed as arguments. Print should print the complex number that the object represents. You may declare any public getters/setters or private data members that you deem necessary. Your code should work with the example below.

```
int main() {
(1)   Complex c1(5, 6);
(2)   Complex c2(-2, 4);
(3)   Complex* c3 = new Complex();

(4)   c1.print();
(5)   c2.print();
(6)   cout << "The sum of the two complex numbers is:" << endl;
(7)   c3->sum(c1, c2);
(8)   c3->print();
(9)   delete c3;
}

// The output of the main program:
5+6i
-2+4i
The sum of the two complex numbers is:
3+10i
```

Bonus: What would happen if we swapped line (8) and (9)?

Bonus: What would happen if swapped the order of (8) and (9)? How would it change the output?

Q4.

a. What's the main difference between declaring a type with the keyword **`struct`** and declaring it with the keyword **`class`**?

Where can you use public members? Where can you use private members?

Q5.

a. When should you write a destructor for a class?
b. (True/False) A class may have more than one destructor.
c. What happens if you forget to deallocate dynamically allocated memory once you're done with it?

Q6.

a. If you have an object pointed to by a pointer, which operator is used with the pointer to access the object's members?

b. What does the following program output?

```
int main() {
    int a = 100, b = 30;
    cout << a + b << endl;                // (1) 130

    int* ptr = &a;
    cout << *ptr + b << endl;             // (2) 130

    *ptr = 10;
    cout << *ptr + b << endl;             // (3) 40  (a = 10, b = 30)

    ptr = &b;
    *ptr = -12;
    cout << *ptr + 2*b << endl;           // (4) -12 + 2(-12) = -36

    int c = a + *ptr;
    cout << c << endl;                    // (5) -2

    b = -5;
    cout << a + b << endl;                // (6) 5

    int arr[5] = {4, 5, 10, 11, -1};
    ptr = arr + 1;
    cout << *arr + *ptr << endl;          // (7) 9
```

```
    int cs;
    int& pic = cs;
    ptr = &pic;
    pic = 31;
    cs++;
    cout << *ptr << endl;           // (8) 32
}
```