Please watch or skim the videos listed on the Canvas assignment corresponding to this LGA. There is a lot more content than usual in this LGA, since it should be review material for most of you. Feel free to fast-forward though any bits you feel pretty confident about already.

Individual self-test questions

All students should answer this set of questions. You will compare your answers with your group during the next lecture period. Note, we may from time to time include trick questions.

- Using the operator ______, we can obtain the address of the value stored in a variable. The address can be stored in a ____________ variable.
- Using the operator ______, known as the __________________ operator, we can get or set the value pointed to by a pointer.
- Array variables are secretly __________________; however, describe one difference:
- To the compiler, `p[j]` is equivalent to __________________.
- If `p` is an `int*`, and `p = 0x1000`, what is the value `p + 3`? ____________.
- The memory segment containing the function call stack is generally called the _______________.
- Dynamically allocated memory for C++ comes from the __________________.
- The _______________ operator is used to dynamically allocate memory; the result of the operation is a _______________ to the memory.
- The _______________ operator is used to deallocate dynamically allocated memory.
- Name two types of pointer you should never try to dereference: ______________________________.
- If your program regularly loses pointers to dynamically allocated memory, your program may have this kind of memory error: _______________.

Group questions

Distribute the following questions across the members of your group. You will share your solutions (and most importantly the *method* of your solutions) during the next lecture period. Divide up the questions so that each question has at least three solutions from different group members. *Note:* from time to time we may include trick questions.

1. (Basic) Provide the missing line in the code below so that the program prints "42":

```
void go(int* ptr) {
                // <- put your code here
}

int main() {
    int x = 17;
    go(&x);
    cout << x << endl;
    return 0;
}
```

2. (Basic) What is the output of the following code?

```
int a = 42;
int b = 17;

int* p = nullptr;
int* q = &a;

*q = 99;

p = q;
q = &b;

*q = 22;
*p = 77;

cout << a << " " << b << " " << q << endl;
```

3. (Intermediate) What is the output of the following code?

```
int upper = 10;
int arr[11];

for (int i = 0; i <= upper; i++) arr[i] = i;

for (int* p = arr; p <= arr + upper; p = p + 3) {
    cout << *p << " ";
}
cout << endl;

for (int* q = arr + upper; q >= arr; q = q - 2) {
    cout << *q << " ";
}
cout  << endl;
```

4. (Intermediate) Given some class `foo` with a public method named `run()`, write the function `run_all()` that takes in an array of `foo` objects and the length of the array and calls the `run()` method on each object. Give *two* different solutions for looping over the array (i.e., one using array indexing and another using pointers):

```
class foo {
public:
    void run();  // actual behavior defined elsewhere
};

void run_all(foo* objects, int len) {


}
```

5. (Basic) Write a program to dynamically allocate an array of 80 `ints`, fill every array position with `17`, then deallocate the array.

6. (Basic) What is problematic with this code?

```
    double* get_sum(double x, double y) {
        double ans = x + y;
        return &ans;
    }
```

7. (Basic) What is problematic with this code?

```
int main() {
    char* s;
    while (true) {
        int n = rand() % 10000;
        delete[] s;
        s = new char[n];
        for (int i = 0; i < n; i++) s[i] = 'a';
    }
}
```

8. (Intermediate) Explain the bug or issue with the code below, or provide a convincing argument that none exists:

```
void release(char* p, int n) {
    if (n % 2 == 0) delete[] p;
}

int main() {
    int a = rand();
    char *x = new char[100];
    if (a % 2 == 1) delete[] x;
    release(x, a);
}
```

9. (Intermediate) Explain the bug or issue with the code below, or providing a convincing argument that none exists:

```
int* generate(int n, int bound) {
    int* arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = rand() % bound;
    }
    return arr;
}

int main() {
    int* result;
    for (int x = 1; x <= 100000; x++) {
        result = generate(x, x);
        for (int i = 0; i < x; i++) {
            cout << result[i] << " ";
        }
        cout << endl;
    }
    delete[] result;
    return 0;
}
```