```
//References vs. Pointers
void swapWithPointer(int* a, int* b) {
	int temp = *a;
	*a = *b;
	*b = temp;
}

void swapWithReference(int &c, int &d) {
	int temp = c;
	c = d;
	d = temp;
}
```

```
//countDucks.cpp
#include <iostream>
#include <cstdlib>
#include <fstream>

using namespace std;

int main(int argc, char *argv[]){
  if (argc!=2) {
    cerr << "Usage: "<< argv[0] << " inputFile" << endl;
    exit(1);
  }

  ifstream ifs;
  ifs.open(argv[1]);

  if(!ifs) {
	cerr << "Failed to open file." << endl;
	cerr << "Usage: " << argv[0] << " inputFile" << endl;
	exit(1);
  }

  string line;
  int numDucks = 0;

  while(ifs) {
	getline(ifs, line);
	int pos = line.find("duck");
	if(pos >= 0) {
		numDucks++;
	}
  }

  cout << "There were " << numDucks << " ducks in " << argv[1]
<< endl;

  return 0;
}
```

```
//shapeFuncs.cpp (lab05)
double distanceBetween(Point p, Point q) {
  return (sqrt(pow(p.x - q.x, 2.0) + pow(p.y - q.y, 2.0)));
}

void initPoint(struct Point *p, double xVal, double yVal) {
  (*p).x = xVal;
  (*p).y = yVal;
}

string pointToString(Point p, int precision) {
  ostringstream oss;
  oss << setprecision(precision);
  oss << "(" << p.x << "," << p.y << ")";
  return oss.str();
}

string boxToString(Box b, int precision) {
  // SAMPLE FORMAT: [ul=(3.4,-5),w=5,h=7]
  ostringstream oss;
  oss << setprecision(precision);
  oss << "ul=(" << b.ul.x << "," << b.ul.y << "),w=" << b.width
<< ",h=" << b.height;
  return oss.str();
}

bool pointsApproxEqual(Point p1, Point p2, double tolerance) {
  return distanceBetween(p1,p2) < tolerance;
}

bool boxesApproxEqual(Box b1, Box b2, double tolerance) {
  if(pointsApproxEqual(b1.ul, b2.ul, tolerance) &&
approxEqual(b1.width, b2.width, tolerance) &&
approxEqual(b1.height, b2.height, tolerance)) {
    return true;
  }
  else {
    return false;
  }
}

void initBox(struct Box *b, double ulx, double uly, double w,
double h){
  (*b).ul.x = ulx;
  (*b).ul.y = uly;
  (*b).width = w;
  (*b).height = h;
}

double areaOfBox(Box b) {
  return (b.width * b.height);
}
```

```
//Spring 2017 Midterm - aliens
string getSmarterAlien(Alien* a1, Alien* a2) {
	if((a1 == NULL) || (a2 == NULL)) {
		cerr << "Invalid pointer(s)";
		exit(1);
	}
	if(a1 -> IQ == a2 -> IQ)
		return "";
	else if(a1 -> IQ > a2 -> IQ)
		return a1 -> name;
	else
		return a2 -> name;
}

int numFriendly(Alien* arr, int len, string planet) {
	if(len == 0)
		return 0;

	int nums = 0;
	for(int i = 0; i < len; i++) {
		if(((arr+i) -> planet == planet) && ((arr+i) ->
isFriendly))
			nums++;
	}
}
```

```
//Data Representations
//Binary - base 2, represented by 0s and 1s
//Decimal - base 10, represented by 0-9
//Hexadecimal - base 16, represented by 0-9 and A-F

//Data Type Memory Sizes
//int, float - 4 bytes
//double - 8 bytes
//char - can only hold one char on keyboard, 1 byte
//pointers - 4 bytes

//Change from type int to double
static_cast<double>(i)

//gcc steps
//$ g++ -s hello.cpp (turns cpp into assembly language)
//$ g++ -c hello.o (makes object file)
//$ g++ -o hello hello.cpp (produces executable named hello)
//$ g++ functions.o main.o -o myhello (links multiple .o files
  into one final executable)

//Test if a pointer is valid
//assert (b != NULL);
```

```
//My implementation of aliens
string getSmarterAlien(Alien num1, Alien num2) {
	if(num1.IQ == num2.IQ) {
		string result = "";
		return result;
	}
	else if(num1.IQ > num2.IQ) {
		return num1.name;
	}
	else if(num2.IQ > num1.IQ) {
		return num2.name;
	}
	else {
		cerr << "Invalid pointer(s)" << endl;
		exit(1);
	}
}

int numFriendly(Alien* aliens, int numElements, string planet) {
	int friendly = 0;
	if(numElements == 0) {
		return 0;
	}
	else {
		for(int i = 0; i < numElements; i++) {
			if(aliens[i].planet == planet) {
				if(aliens[i].isFriendly) {
					friendly++;
				}
			}
		}
	}

	return friendly;
}
```

```
//Writing to files
int main() {
	ofstream ofs;
	//Open a file for writing
	ofs.open("animals.txt");

	//Write to it
	ofs << "Duck\nCow\nGoat\nParrot\n";
	//lab05 has these animals!

	//Close the file
	ofs.close();

	return 0;
}
```