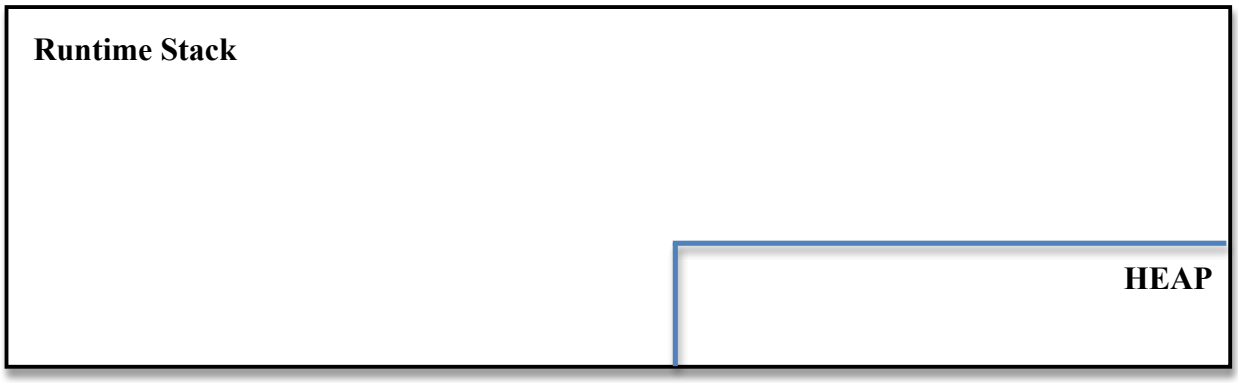
(0) **Do this question *before* you start coding.** Assume your C++ program declared the following variables and arrays:

```
double a=1, b=2, c=3;
double d[4] = {100,101,102,103};
double e=18, f=19;
double g[4] = {200,201,202,203};
short  h=22, i=23;
short* p = new short[3];
p[0] = 34;  p[1] = 35;  p[2] = 36;
```

Make a hypothesis (**drawing**) of what the memory in `main()`'s runtime stack and Heap (RAM) would look like given these declarations. Make a *neat* drawing; assign each variable or array an integer memory location (you can make up the integer values for the memory addresses). **Take care to label contiguous (next to each other) memory locations with the correct values (e.g., g[2] and g[3] are *next* to each other).** Also indicate which compiler/IDE you are using.

My hypothesis as to how the _____ C++ compiler will allocate memory



Call me over when you have your *neatly labeled* drawing finished.

(1) Download the Starter Kit from onCourse. Create a console project and load the `.cpp` file called: **`1_whereIs.cpp`**. Use this program to prove where/how memory is actually allocated by this compiler when you declare variables and arrays in your main program. Remember, if your index `[i]` is *outside* an array's bounds (off the left or right end), you will be manipulating some other memory location, right? **Make a new drawing based on your program's output.** Find some places in the code that are surprising/dangerous. Be prepared to point out things that you have learned. Note how I have printed memory locations as (`unsigned long`). The default is that memory value are printed as hexadecimal. **(Remove one of the `(unsigned long)` casts to check).**

```
cout << "    a = " << (unsigned long) &a    << "     " << a << endl;
```

Empirical verification of how _____ 's C++ compiler allocates memory