* **Fundamental Types:**
* **bool** – definite size of 1 byte
* **char –** definite size of 1 byte
* short
* int
* long
* float
* double
* long long
* long double

**Implicit Data Types Conversion**

* right handed data type values will get converted into left handed data type value.
* The one with higher data type values/details.

**Arrays**

**Java**

* Can't compiled if not bounded
* Will have display error syntax if array is out of bound
* Array declaration syntax has “new” keyword
* Array size can be dynamic

**C++**

* Can compiled if not bounded
* Array size can't be dynamic
* Will print out the memory address if array is out of bound
* Array declaration doesn't need “new” keyword
* Can be array as pointers

**Pointers & References (syntax, capabilities)**

**Pointers**

* are not constant, can have their targets changed
* “\*” keyword
* don't need to be initialized
* need to point to the variable address
* can't point to another pointer, but can point to the address that another pointer points to
  + i.e. \*a = &x & \*b = a => \*b = &x
* can't point to a different data type (type-safe)

**References**

* are constant, non-literals
* “&” keyword
* needs to be initialized or it won't compiled, and can't initialized with an integer literal
* part of lvalues
* Can pass an integer literals

**Declaration**

* Statement of existence
* declare before main()
* doesn't need know the data type inside the parameter
* doesn't need to know the variable name inside the parameter

**Definition**

* Statement of behavior
* body of function

**#include**

* Search for .h through source files in the current directory first, then paths specified as part of compiler/IDE settings
* looks for <> in the system library because it's in angled brackets
* If compiler can't locate the header file because it's not located in the directory containing the source file or paths specified to be part of the compiler

Example Exam Questions:

A mystery function is shown below. What value is returned by the function?

Int mystery(){

int matrix[3][3] = {

{1, 2, 1},

{2, 1, 2},

{1, 2, 3}

};

int plus = 0, minus = 0;

for (int I = 0; I < 3; i++){

plus += matrix[0][i] \* matrix[1][(i+1)%3] \* matrix[2][(i+2)%3];

minus += matrix[2][i] \*matrix[1][(i+1)%3] \* matrix[0][(i+2)%3];

}

return plus – minus;

}

I = 0

plus += 1 \* 1 \* 3

minus += 1 \* 1 \* 1

i= 1

plus += 2 \* 1 \* 3

minus += 2 \* 1 \* 1

I = 2

2. Write a function called SwapTwo, which takes two double variables as parameters and swaps their values, *so that the original variables passed as arguments to the method are also changed*. The method should not do any input or output with the user, and should have a void return type. Once you have written the function, fills in the main below to show how you would call the function.

Void SwapTwo(double \*original, double \*swap){

//write the function here

double temp;

temp = \*swap;

\*swap = original;

\*original = temp;

}

int main(int argc, char\* argv[]){

double a = 5.7, b = 6.5 //these values are arbitrary and irrelevant to your answer

//show how to call SwapTwo so that a and b have their values swapped.

SwapTwo(&a, &b);