The '&' symbol in C++ is known as the _____ operator.

address

A _____ refers to the address of a memory location in C++.

pointer

This is the indirection operator.

*

_____ refers to indirectly accessing a variable being pointed to in order to use the variable's value or modify it.

dereferencing

An array name can be treated as a _____.

pointer

Given

double dArray[5];

If (dArray + 1) refers to address 650008, then (dArray + 3) refers to address _____.

650024

This expression will increment the value of the integer variable y by 5 using a pointer variable ptrY.

```
*ptrY += 5;
```

or

```
*ptrY = *ptrY + 5;
```

Given

double dArray[5];

Identify the error in this expression:

***dArray + 3 = 5.25;**

Need parentheses to dereference the array element: *(dArray + 3) = 5.25;

Identify the error in the code shown below:

```
int main() {
        const int SIZE = 3;
        double dArray[SIZE] = { 1.0, 2.0, 3.0 };
        // double the contents of the array
        doubleArray(dArray, SIZE);
        ...
}
void doubleArray(double array, int size)
{
         for (int i = 0; i < size; i++)
               *array++  *=  2.0;
}
```

asterisk not included in array parameter: double *array

Passing a pointer to a function is known as pass-by-_____.

address

The“if” in the code below evaluates to this (true or false)

```
int array[] = { 4, 3, 2, 1 };
if (array[1] < array[3])
…
```

false

The"if" in the code below evaluates to this (true or false)

```
int array[] = { 4, 3, 2, 1 };
if (&array[1] < &array[3])
…
```

true

Local variables in a C++ application are mapped in a memory location known as the _____.

stack

Dynamic memory is allocated from the _____.

heap

The new operator obtains memory from the heap and assigns the address to a _____.

pointer

Once you have completed working with dynamically allocated memory you should return it to the heap using the _____ operator.

delete

Identify the error in the code below which is returning a dynamically allocated array to the heap:

```
delete arrayPtr;
```

missing square brackets:

delete [] arrayPtr;

Failure to delete dynamic memory can result in these.

memory leaks

Failing to set a pointer to nullptr after deleting its associated memory results in a _____.

dangling pointer

Identify the error in the following function:

```
int* getArray(const int sz)
{
    int sqArray[sz];
    return sqArray;
}
```

returning a pointer to a local variable (remember arrays are treated as pointers)

When dynamically allocating memory in a class, do the allocation in the constructor, and the delete in the _____.

destructor

Identify the error in the code below:

```
Rectangle r;
Rectangle *pR = r;
```

missing address-of operator:

Rectangle *pR = &r;

_____ is an implicit pointer which, when referred to from within an object instance, refers to the object itself

this

Identify the error in the code below:

```
int main()
{
   int x = 25;

   doubleAndDisplay (x);
   return 0;
}

void doubleAndDisplay(const int &value)
{
   value *= 2;
   cout << "value is " << value << endl;
}
```

parameter “value” is being passed as a const, so it cannot be modified

Redeclare the following member function so that the class cannot be modified through "this":

```
int Rectangle::doSomething();
```

int Rectangle::doSomething() const;

_____ members are variables or functions associated with a class

static

T/F: static members can be accessed through the class name using "::" notation _or_ using an object instance.

true

T/F: instance members can be accessed through the class name using "::" notation _or_ using an object instance.

false

Given the following declaration, how would you allocate/implement the static variable "s"?

```
class StatDemo
{
        private:
                static int s;
                int counter;
};

        int StatDemo::s;
```

Given the following declaration, how would you call the static function "doStat" without using an object reference variable?

```
class StatDemo
{
        public:
                static void doStat();
};


        StatDemo::doStat();
```

A _____ is a function that is not a member of a class but has access to private members of a class

friend

_____ allows us to represent the "is-a" relationship in our classes

inheritance

In an inheritance relationship, the _____ class is the general class and the _____ class is the specialized class.

base class, derived class

Modify the constructor for the Cat class below to call the Furball base class constructor with a parameter named "weight"

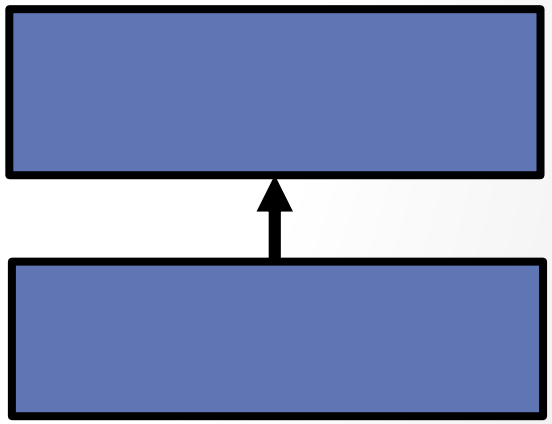
```
Cat(string name) {
        this->name = name;
}
```

```
Cat(string name) : Furball(weight) {
…
```

This is the open mode which should be used to append to an output file

ios::out | ios::app

In a UML diagram, the base class goes in the _____ (top/bottom) box and the derived class goes in the (top/bottom) box



base class: top, derived class: bottom

"Objects of a derived class can be used wherever objects of a base class are expected"; this is a form of _____.

polymorphism

Identify the error in the code shown below (assume Cat is derived from Furball):

```
Cat *c = new Cat();
Furball *f = c;
Cat *c2 = f;
```

A cat is always a furball, so c can be assigned to f. But a furball is not necessarily a cat (could be a dog) so f cannot be assigned to c2 without an explicit cast.

Correct the code shown below so it will build.

```
Cat *c = new Cat();
Furball *f = c;
Cat *c2 = f;
```

Cat *c2 = static_cast<Cat*>(f);

_____ support polymorphism by allowing the most specific version of a member function in an inheritance hierarchy to be selected for execution.

virtual functions

Declare a prototype for a virtual C++ function named "catYears" which returns an integer and takes an integer parameter named "humanYears".

virtual int catYears(int humanYears);

T/F: the virtual characteristic of a function is not inherited, it only applies to functions explicitly declared as virtual.

False: it is inherited throughout the hierarchy

_____ binding occurs when the compiler binds a function name at compile time to the code that should be executed

static

_____ binding occurs at run time for virtual functions which may be associated with multiple types.

dynamic

A _____ function has no body; we declare it in a base class in order to force subclasses to implement it

pure virtual

Modify the following function prototype to declare it as pure virtual:

int showVal(bool withNewline);

virtual int showVal(bool withNewline) = 0;

Pure virtual functions are also known as _____ functions.

abstract

A class containing a pure virtual function is known as a(n) _____ class.

abstract

T/F: an abstract class cannot be instantiated with the "new" operator.

true

The _____ access modifier restricts member access to within a subclass.

protected

If a member-wise assignment operation is performed on an object which contains members that are pointers, you must define a _____ constructor to copy the contents of the member pointers; otherwise only the address values are copied.

copy

T/F: Base class constructors can be called from a copy constructor.

True

T/F: Member-wise assignment occurs via a copy constructor when overloading the "=" assignment operator.

False

Throwing a(n) _____ indicates an error has occurred in a C++ application.

exception

T/F: an exception argument in C++ can be of any type.

True

A C++ _____ supports generic programming.

template

C++ function and class template declarations start with this keyword.

template

T/F: templates should be placed after the main function in a C++ source file.

False

The _____ is now part of the C++ Standard Library.

STL (Standard Template Library)

An STL _____ is a class that stores and organizes data

container

The _____ function is used to add an item to an STL vector.

push_back

A(n) _____ is an object in STL that acts like a pointer and is used to access items stored in containers

iterator

A _____ loop uses an iterator to traverse a sequential container (can also be used with simple arrays).

range-based for