Concept Questions

1). How would you convert a char (representing an int) to an int?

Subtract the zero character from it.

```
int num = character – '0';
```

2). Where is dynamic memory allocated? Where is static memory allocated?

Dynamic memory is allocated on the heap, whereas static memory is allocated on the stack.

3). Is the size variable allocated statically or dynamically? How about the integer pointer? How about the new array of integers?

```
void function(int size){
        int * arr = new int[size];
}
```

The size parameter is static, the pointer ("arr" itself) is static, and the actual array of integers pointed to by "arr" is dynamic.

4). In the above example, which of the three allocations are freed from memory and where?

The two statically allocated variables are freed at the closing bracket (end of the scope). In this example, the dynamic array is not actually freed at all. To prevent a memory leak, it should be freed before you lose access to it at the end of the function.

5). What are the three instances in which the copy constructor is called?

Creating an object and specifying the copy constructor, passing by value, and returning by value;

```
MyClass a(b);                   //Call the copy constructor explicitly
void func(MyClass c) {          //Called to copy in parameter
        Return d;               //Called to return value
}
```

6). Why is the parameter of the copy constructor passed by reference and not by value?

If the parameter to the copy constructor is passed by value, it will call the copy constructor, which will then call the copy constructor…etc. You must pass the parameter by reference to prevent an infinite loop.

7). What are the three types of constructors?

Default, parameterized, and copy.

8). What are the two types of default constructors?

One with no parameters at all, and one with all default parameters.

9). What are the three ways to call the following defaulted constructor?

```
student(int = 5, int = 10);
```

```
student();

student(100);                    //Uses only the 1st default parameter

student(100,50);
```

10). What's the difference between a class member function and a friend function? What does the friend keyword do?

A friend function is not actually a part of a class, but has access to the private members of that class. This means that it does not have a calling object, as it is essentially a normal function with special access, so you must pass an instance of your class as a parameter. A member function is actually part of a class, and automatically has access to the data members of that class. Member functions also always have a calling object, so they automatically know which instance in particular you are talking about.

```
object.function();               //Member function

function(object);                //Friend function
```

11). How many times can a constructor be called for a single object? How many times can a destructor be called for single object? Can both a copy constructor and default constructor be called for the same object?

Only one type of constructor can be called once for any single object, and it is called when that object is created, or instantiated to be exact. The destructor is also called only once, when the object is either deleted (if dynamic), or goes out of scope.

12). If you do not write a constructor or destructor, what happens?

The compiler automatically generates one for you, but it will probably not do what you want—it will not default values and will not delete your memory.

13). A default copy constructor is provided for every class. When should you write your own?

The default copy constructor does what's called a "shallow" copy, meaning it essentially sets each data member equal to the source with the assignment operator. However, if you have a pointer as a data member, it will cause both pointers to point to the same data. This may be what you want, but you usually want to actually make a copy of that data, and have the new pointer point to the copy. To do this, you must implement your own copy constructor.

14). What is a memory leak and how does it occur? What is a dangling pointer and how does it occur?

A memory leak occurs when you lose access to dynamically allocated memory, which means it can never be freed. This is especially an issue when the code that leaks memory is called more than once, say in a function or a loop, as every time that code is run the program will leak more memory, until

it eventually runs out of memory and crashes. A dangling pointer is a pointer that does not point to valid data, for example a garbage pointer. Always avoid dangling pointers by setting them to NULL when you delete dynamically allocated memory.

15). Structs and classes are both examples of what?

Abstract Data Types (ADTs).

16). How many times is the destructor called for the student class in the following code?

```
student* s = new student[100];
delete[] s;
```

100, once for each "student" in the array.

17). How would you delete the following?

```
student *s1 = new student;
```

```
delete student;                    //Note no brackets, as we did not allocate an array
```

18). Do all character arrays require a null character at the end of the significant characters? When is it necessary?

Arrays of characters in general do not need to end with the null character, but this is the defining feature of C-Style strings, which must always end with the null character.

19). What is the difference between public and private keywords in a class?

Private makes data members only accessible from within member functions of the class, whereas public allow them to be accessed from anywhere.

```
class MyClass {
public:
        int x;
private:
        int y;
};
int main() {
        MyClass o;
        o.x = 5;                   //Valid, x is public
        o.y = 5;                   //Invalid, y is private
}
```

20). When should you make a variable constant? When should you make a class member function constant? Would you ever need to make a non-member function constant?

You make a variable constant when you know that the value should never change, as making it constant will assure that. You make a member function constant when it will never modify the data members of that class. Finally, you can't make non-member functions constant, as they have no associated object and that doesn't make sense.

21). Can a constant object always call all member functions for its corresponding class? If not, which member functions can it call?

A constant object can only call member functions if they are constant themselves, as explained in question 20.

```
const MyClass c;

c.function();              //Function must be a constant member function
```

22). What do the following operators do? *ptr, &, ptr->, ptr[]

The star dereferences a pointer, the ampersand takes the address of a variable, the arrow dereferences a pointer and then uses the dot operator, and the brackets offset from a pointer and then dereference it.

```
*pointer = 5;

pointer = &variable;

classPointer->function();

arrayPointer[3] = 5;
```