Question 1: What is Composition and Aggregation?

The difference is only in the ownership. In composition, when the owning object is destroyed, so are the contained objects. In aggregation, this is not necessarily true.

For example, a university owns various departments (e.g., chemistry) , and each department has a number of professors. If the university closes, the departments will no longer exist, but the professors in those departments will. Therefore, a University can be seen as a composition of departments, whereas departments have an aggregation of professors.

Question 2: What is the difference between pointer and reference?

1. Pointer can refer NULL, Reference cannot be NULL.
2. Pointer can dereference to another address, Reference once initialized stays that way till it dies.
3. Pointer may not associate with a legitimate memory but References should associate with certain memory.
4. Pointer may not be initialized while created but reference has to be initialized when it is created.

Question 3: what is the order of calling of constructors and destructors in a class hierarchy?

Every time an instance of a class is created the constructor method is called. The constructor has the same name as the class and it doesn't return any type, while the destructor's name it's defined in the same way, but with a '~' in front.

Even if a class is not equipped with a constructor, the compiler will generate code for one, called the implicit default constructor. This will typically call the default constructors for all class members, if the class is using virtual methods it is used to initialize the pointer to the virtual table, and, in class hierarchies, it calls the constructors of the base classes. Both constructors in the above example use initialization lists in order to initialize the members of the class.

The construction order of the members is the order in which they are defined, and for this reason the same order should be preserved in the initialization list to avoid confusion.

http://www.cprogramming.com/tutorial/constructor_destructor_ordering.html

Question 4: what is RTTI in C++?

RTTI is a compiler which works if we don't declare data-type to particular variable, then this compiler take defaultly the data-type of the variable according to the data defined to it.

E.g. if we write only

```
#include <stdio.h>
Void main ()
{
x=5; y=6.9; z= 'A';
. . ..
}
```

Then compiler takes defaultly “x” as integer “y” is float & “z” as char types.

Question 5: What is the Difference between "vector" and "array", List and Deque?

Vector and Array:

Vector is also an array but the size of a vector can change dynamically where in array its fixed. we can create some reserve space in vector where in array we cannot.

List and Deque:

- List: manages its elements as a doubly linked list and does not provide random access.
- Deque: manages its elements with a dynamic array, provides random access.

- List: inserting and removing elements is fast at each and every position.
- Deque: provides Fast insertions and deletions at both the end and the beginning. Inserting and deleting elements in the middle is relatively slow.

Question 6: Explain "passing by value", "passing by pointer" and "passing by reference"

Pass by Value example:

The function receives a copy of the variable. This local copy has scope, that is, exists only within the function. Any changes to the variable made in the function are not passed back to the calling routine. The advantages of passing by values are simplicity and that is guaranteed that the variable in the calling routine will be unchanged after return from the function. There are two main disadvantages. First, it is inefficient to make a copy of a variable, particularly if it is large such as an array, structure or class. Second, since the variable in the calling routine will not be modified even if that's what is desired, only way to pass information back to the calling routine is via the return value of the function. Only one value may be passed back this way.

Consider function:

```
 void foo(int i);
int j=3;
foo(j);
```

Pass by Reference example:
C++ provides this third way to pass variables to a function. A reference in C++ is an alias to a variable. Any changes made to the reference will also be made to the original variable. When variables are passed into a function by reference, the modifications made by the function will be seen in the calling routine. References in C++ allow passing by reference like pointers do, but without the complicated notation. Since no local copies of the variables are made in the function, this technique is efficient. Additionally, passing multiple references into the function can modify multiple variables.

Consider function:

```
void foo(int& i);
int j=3;
foo(&j);
```

Pass by Pointer example:

A pointer to the variable is passed to the function. The pointer can then be manipulated to change the value of the variable in the calling routine. The function cannot change the pointer itself since it gets a local copy of the pointer. However, the function can change the contents of memory, the variable, to which the pointer refers. The advantages of passing by pointer are that any changes to variables will be passed back to the calling routine and that multiple variables can be changed.

Consider function :

```
void foo(int* i);
int j=3;
int* p=&j;
foo(p);
```

if you look at pass by ref & pass by pointer it is almost the same but in pass by pointer you can do pointer arithmetic operation whereas in ref you cannot.

Question 7: Difference between a Thread and Process?

The major difference between threads and processes is

1. Threads share the address space of the process that created it; processes have their own address.
2. Threads have direct access to the data segment of its process; processes have their own copy of the data segment of the parent process.
3. Threads can directly communicate with other threads of its process; processes must use inter process communication to communicate with sibling processes.
4. Threads have almost no overhead; processes have Considerable overhead.
5. New threads are easily created; new processes require duplication of the parent process.
6. Threads can exercise considerable control over threads of the same process; processes can only exercise control over child processes.
7. Changes to the main thread (cancellation, priority change, etc.) may affect the behavior of the other threads of the process; changes to the parent process do not affect child processes.

Question 8: Difference between Stack and Heap memory?

STACK memory is referred as temporary memory, if you come out of the program the memory of the variable will be no more. [Eg: int a; memory for a will not maintained after we came out from the program].

HEAP memory is referred as Permanent Memory. This means memory allocated for the object will be maintained even if we came out of the program. [Eg: Memory for OBJECT will remain there ever].

Question 9: Implement a algorithm to sort an array?

Refer Notebook for the answer.

Question 10: Explain pre-order, post-order, and in-order in a binary tree traversal?

Pre-order, in-order, and post-order traversal visit each node in a tree by recursively visiting each node in the left and right subtrees of the root. If the root node is visited before its subtrees, this is preorder; if after, post-order; if between, in-order.

Question 11: What is the difference between char a[] = "string"; and char *p = "string";?

“a” is a constant pointer, whereas p is not. This means 'a' will point to a fixed location (value of 'a' or address of *a can't change, remains fixed) - though contents of “a” can be changed (by way of accessing a[i])

Question 12: What is the difference between structure & union?

The difference between structure and union is that, if we declared two structure variables as **struct strct1 x y;** then the two structure variables x and have different memory location.
But if we declare two union variables as **union uni1 x y;** then the two union variables x and have same memory location.

Question 13: explain compiling and linking in brief?

Compilation:-
Compilation refers to the processing of source code files (.c, .cc, or .cpp) and the creation of an 'object' file. This step doesn't create anything the user can actually run. Instead, the compiler merely produces the machine language instructions that correspond to the source code file that was compiled. For instance, if you compile (but don't link) three separate files, you will have three object files created as output, each with the name <filename>.o or <filename>.obj (the extension will depend on your compiler). Each of these files contains a translation of your source code file into a machine language file -- but you can't run them yet! You need to turn them into executables your operating system can use. That's where the linker comes in.

Linking:-

Linking refers to the creation of a single executable file from multiple object files. In this step, it is common that the linker will complain about undefined functions (commonly, main itself). During compilation, if the compiler could not find the definition for a particular function, it would just assume that the function was defined in another file. If this isn't the case, there's no way the compiler would know -- it doesn't look at the contents of more than one file at a time. The linker, on the other hand, may look at multiple files and try to find references for the functions that weren't mentioned.

Question 14: What is DLL? And explain few advantages of DLL?

A dynamic link library (DLL) is a collection of small programs, any of which can be called when needed by a larger program that is running in the computer.

The advantage of DLL files is that, because they don't get loaded into random access memory (RAM) together with the main program, space is saved in RAM. When and if a DLL file is needed, then it is loaded and run. For example, as long as a user of Microsoft Word is editing a document, the printer DLL file does not need to be loaded into RAM. If the user decides to print the document, then the Word application causes the printer DLL file to be loaded and run.

Question 15: What is an interrupt?

Interrupt is a process in which an external device can use the micro-processor. It is an asynchronous signal and it is considered as an emergency signal. Whenever the microprocessor receives an interrupt signal, it stops executing its current program and jumps into the interrupt service routine (ISR). There can be more than one "source" for the interrupt signal and each source has its own ISR

Question 16: difference between UDP and TCP internet protocol?

Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both TCP and UDP work at transport layer TCP/IP model and both have very different usage.

Difference between TCP and UDP:

TCP	**UDP**
Reliability: TCP is connection-oriented protocol. When a file or message send it will get delivered unless connections fails. If connection lost, the server will request the lost part. There is no corruption while transferring a message.	*Reliability*: UDP is connectionless protocol. When you a send a data or message, you don't know if it'll get there, it could get lost on the way. There may be corruption while transferring a message.
Ordered: If you send two messages along a connection, one after the other, you know the first message will get there first. You don't have to worry about data arriving in the wrong order.	*Ordered*: If you send two messages out, you don't know what order they'll arrive in i.e. **no ordered**
Heavyweight: - when the low level parts of the TCP "stream" arrive in the wrong order, resend requests have to be sent, and all the out of sequence parts have to be put back together, so requires a bit of work to piece together.	*Lightweight*: No ordering of messages, no tracking connections, etc. It's just fire and forget! This means it's a lot quicker, and the network card / OS have to do very little work to translate the data back from the packets.
Streaming: Data is read as a "stream," with nothing distinguishing where one packet ends and another begins. There may be multiple packets per read call.	*Datagrams*: Packets are sent individually and are guaranteed to be whole if they arrive. One packet per one read call.
Examples: World Wide Web (Apache TCP port 80), e-mail (SMTP TCP port 25 Postfix MTA), File Transfer Protocol (FTP port 21) and Secure Shell (OpenSSH port 22) etc.	*Examples*: Domain Name System (DNS UDP port 53), streaming media applications such as IPTV or movies, Voice over IP (VoIP), Trivial File Transfer Protocol (TFTP) and online multiplayer games etc

Question 17: Explain In brief Client-Server Technology?

The most common model for distributing a system is the client-server model. The model is fairly simple to explain and use. In your distributed system you have one or more servers. These servers provide services to other parts of the system, called clients. When a server is started it first opens up a particular port through which clients can access it. It then sits down and waits until somebody (the client) attempts to connect to it. When that happens, the server and client exchange some messages and ultimately of the two close the connection. This connection takes place using so-called sockets.

Question 18: how do you get the no of instances created from a class from within a class?

Question 19: Define and Differentiate between RAM and ROM

RAM is Random Access Memory. It is a volatile type of memory that needs electricity to flow to retain information. It is the type of memory that computers use to process programs.

ROM is Read Only Memory. Essentially it is a piece of permanently written information stored as memory. There are versions of this memory that can be rewritten but it is then called EPROM (Erasable Programmable Read Only Memory) and generally takes ultraviolet light to clear.

Question 20: What is the difference between HTTP- Get and HTTP-Post?

As their names imply, Both HTTP- Get and HTTP-Post uses HTTP as their underlying protocol. The GET method creates a query string and appends it to the script's URL on the server that handles the request. The POST method creates a name/value pairs that are passed in the body of the HTTP request message.