# C/C++

1. What is a static function? After I answered this, he asked me how it is accomplished.

Hiding it from other translations units: encapsulation.

1. Assuming you have three N bit unsigned integers a, b and c, what is the min number of bits you would need to store the result of a \* b + c?

2N

1. const char pointer: const char \*

1) The chars the pointer points to cannot be modified.

const pointer to char: char \* const

2) The address the pointer points to cannot be modified.

1. Assuming there's no Array data structure in C, how would you implement it

Use the pointer.

1. Structure padding, why network packets are not padded, how to avoid padding.

#pragma pack instructs the compiler to pack structure members with particular alignment. Most compilers, when you declare a struct, will insert padding between members to ensure that they are aligned to appropriate addresses in memory (usually a multiple of the type's size). This avoids the performance penalty (or outright error) on some architecture associated with accessing variables that are not aligned properly. Avoid padding: \_\_attribute\_\_((\_\_packed\_\_))

1. WHY padding increases systems performance

Padding is done to fast access the memory. Let us take ex of structure padding. In it, structure members are aligned to be based on the memory pointer size. If word size is 4 bytes then data should be read at the offset of multiple of 4.

1. What's the difference between union and struct in C.

Union has shared memory

1. macros and functions:

1) Macro is preprocessed, function is compiled

2) No type checking for macro

3) Code length increases for macro

4) Use of macro can leads to side effects

5) Macro is faster, function has overhead.

6) Before compiling, macro name is replaced by macro value, during function call, transfer of control takes place

7) Macro is useful for small code appears many times; function is good for large code appears many times.

8) Macros do not extend beyond one line.

9) Macro does not check compile errors.

1. \*\*If we add a signed integer and an unsigned integer, will the result be signed or unsigned?

Unsigned. Signed integer max 2^31, Unsigned 2^32.

1. What does the Static key word used in C qualify?
2. given a structure,

struct {int a, float b, char c } x,y;

what are the 4 ways to copy contents of x into y?

1) y = x;

2) y.a=x.a, y.b=x.b, y.c=x.c

3) memmove(&y,&x,sizeof(x));

4) memcpy(&y,&x,sizeof(x));

1. Addition of two signed characters, what happens to the carry bit, and to the signed bit? Can the carry bit be over written on a signed bit?

char a= 127;

a++;

printf("%d",a); // will print -128

1. what type of values cannot be used in switch case can we use alphabets, floats, strings

Accept char no strings

1. \*\*\*constants defined through $define are placed in what type of memory and do they follow block scope will o/p of print be 1

void main() {

funct();

printf("%d",x)

}

funct() {

#define x 1

}

1) All macro will be replaced by the value, so it will be anywhere when the related string was used in the instruction store

2) Replace from the first line it shows

1. What happens when the following piece of code is executed?

char \*ptr;

While(1) ptr = malloc(1024\*1024);

Used up the space in heap, and begin to return NULL.

1. pass array by value to a function

Wrap it in a struct.

1. Given two numbers "a" and "b" and an average formula (a+b)/2. Find one condition where it won’t work. Also, give solution to it.

int avg = (a/2) + (b/2) + ((a%2 + b%2)/2)

1. Differences between C & C++? Give an instance of C code that won't compile under a C++ compiler.

main() {

int new;

}

enum month {jan, feb, march}

int main () {

month cur\_month = feb;

return (cur\_month++);

}

1. Tell me ten features of C++ (that are not in C) that have nothing to do with object oriented programming.

1) Use constant or enumrate instead of Macros

2) Use inline to avoid function calling

3) Use namespace to avoid name clash

4) Use new not malloc to guarantee type safe

5) Use STL template rather than arrays

6) Decelerate variables anywhere

1. What's the difference between abstract class and interface

1) A virtual function can be made to a pure virtual function by initializer = 0. A class with one or more pure virtual functions is abstract class.

2) An important use of abstract classes is to provide an interface without exposing any implementation details.

1. WHY CONSTRUCTORS NEED NOT TO BE DECALRED AS VIRTUAL

A virtual call is a mechanism to get work done given partial information. In particular, "virtual" allows us to call a function knowing only any interfaces and not the exact type of the object. To create an object you need complete information. In particular, you need to know the exact type of what you want to create. Consequently, a "call to a constructor" cannot be virtual.

# Graphics

1. z-buffer
2. What is high dynamic range rendering and why is it important for realistic rendering?

HDRR is the rendering of computer graphics scenes by using lighting calculations done in a larger dynamic range. This allows preservation of details that may be lost due to limiting contrast ratios.

1. What is mip-maps and why are they used?

mipmaps (also MIP maps) are pre-calculated, optimized collections of images that accompany a main texture, intended to increase rendering speed and reduce aliasing artifacts. Since mipmaps cannot be calculated in real time, additional storage space is required to take advantage of them.

1. Why we use 4X4 matrix for representing and calculations in transformation of 3D points when that can be done only with 3X3 matrix.

By homogenous representation, coordinate transformation can be done with one matrix multiplication.

1. What the different shaders(vetex, geomtery, pixel) describe their roles and the order in which they are executed in graphics pipeline?

Graphics pipeline: 3D geometric primitives->modeling->camera transformation->lighting->projection transformation->clipping->Scan conversion or rasterization->Texturing, fragment shading.

Vertex shaders: The purpose is to transform each vertex's 3D position in virtual space to the 2D coordinate at which it appears on the screen, Vertex shaders can manipulate properties such as position, color and texture coordinate, but cannot create new vertices. The output of the vertex shader goes to the next stage in the pipeline.

Geometry shaders: Geometry shader programs are executed after vertex shaders. They take as input a whole primitive, possibly with adjacency information. For example, when operating on triangles, the three vertices are the geometry shader's input. The shader can then emit zero or more primitives, which are rasterized and their fragments ultimately passed to a pixel shader.

Pixel shaders, also known as fragment shaders, compute color and other attributes of each fragment. In 3D graphics, a pixel shader alone cannot produce very complex effects, because it operates only on a single fragment, without knowledge of a scene's geometry. However, pixel shaders do have knowledge of the screen coordinate being drawn, and can sample the screen and nearby pixels if the contents of the entire screen are passed as a texture to the shader. This technique can enable a wide variety of two-dimensional postprocessing effects, such as blur, or edge detection/enhancement for cartoon/cel shaders.

# Brain Teaser

1. Find the highest floor you can drop an egg.
2. angle between hands of the clock (hits:2)

5.5t

1. 8 ball problem. if given 8 balls with only one ball being a lighter weight and all other 7 being the same weight and given a scale, how will you find the light weight ball. do this in minimum number of steps

2 times

1. find the nth prime number in fastest possible way

Every prime no is of the form 6k+1 or 6k-1 except the initial cases i.e. 2 and 3 which has to be taken care of.

# Debugging

1. How would you debug kernel code

"printk", "gdb"

1. How would you do benchmarking (compare the performance) in a device driver code? Apart from timing or time is there any other standard way?
2. How do you debug?

gdb