C

1. The value '\012' means the character with value 12 in octal, which is decimal 10
2. Sizes of integer and pointer are compiler dependent. The both sizes need not be same.
3. In C, when an intger value is compared with an unsigned it, the int is promoted to unsigned. Negative numbers are stored in 2's complement form and unsigned value of the 2's complement form is much higher than the sizeof int.
4. Please note that signed, unsigned and long all three are Type specifiers. And int is implicitly assumed in all of three. As per C standard, “int, signed, or signed int” are equivalent. Similarly, “unsigned, or unsigned int” are equivalent. Besides, “long, signed long, long int, or signed long int” are all equivalent. And “long long, signed long long, long long int, or signed long long int“ are equivalent.
5. C language uses 4 storage classes, namely auto,extern,register,static.
6. Final methods cannot be overridden
7. when a function is static, runtime polymorphism doesn't happen.
8. static data members do not contribute in size of an object. So ‘i’ is not considered in size of Test. Also, all functions (static and non-static both) do not contribute in size.

9)When you clone a single dimensional array, such as Object[], a "deep copy" is performed with the new array containing copies of the original array's elements as opposed to references.

10)int cloneArray[] = intArray.clone();

11)A clone of a multidimensional array (like Object[][]) is a "shallow copy" however, which is to say that it creates only a single new array with each element array a reference to an original element array but subarrays are shared.

int cloneArray[][] = intArray.clone();

12)Python uses TIMsort in built in sorting algorithm.

13) Array name in C is implemented by a constant pointer. It is not possible to apply increment and decrement on constant types.

14) Predict the output of below program:

|  |
| --- |
| #include <stdio.h>    int main()  {      int arr[5];      // Assume base address of arr is 2000 and size of integer is 32 bit      printf("%u %u", arr + 1, &arr + 1);        return 0;  } |

Name of array in C gives the address(except in sizeof operator) of the first element. Adding 1 to this address gives the **address plus the sizeof type** the array has. Applying the **Address-of** operator before the array name gives the address of the whole array. Adding 1 to this address gives the **address plus the sizeof whole array**.

15) char c[] = "GATE2011";

// p now has the base address string "GATE2011"

char \*p = c;

// p[3] is 'E' and p[1] is 'A'.

// p[3] - p[1] = ASCII value of 'E' - ASCII value of 'A' = 4

// So the expression p + p[3] - p[1] becomes p + 4 which is

// base address of string "2011"

printf("%s", p + p[3] - p[1]); // prints 2011

16) We have created a 3D array that should have 2\*3\*3 (= 18) elements, but we are initializing only 9 of them. In C, when we initialize less no of elements in an array all uninitialized elements become ‘\0′ in case of char and 0 in case of integers.

17) The function mainly replaces more than once consecutive occurrences of a character with one one occurrence.

#include <stdio.h>

int fun(char \*p)

{

if (p == NULL || \*p == '\0') return 0;

int current = 1, i = 1;

while (\*(p+current))

{

if (p[current] != p[current-1])

{

p[i] = p[current];

i++;

}

current++;

}

\*(p+i)='\0';

return i;

}

int main()

{

char str[] = "geeksskeeg";

fun(str);

puts(str);

return 0;

}

o/p-gekskeg

18) Note that the sizeof() operator would return size of array. To get size of string stored in array, we need to use strlen().

19) The function basically reverses the given string.

#include <stdio.h>

char str1[100];

char \*fun(char str[])

{

static int i = 0;

if (\*str)

{

fun(str+1);

str1[i] = \*str;

i++;

}

return str1;

}

int main()

{

char str[] = "GATE CS 2015 Mock Test";

printf("%s", fun(str));

return 0;

}

20) size of the void pointer i.e. pVoid would equal to size of int.

21)  fields/members of union share same memory, both i1 and i2 refer to same location. Also, since both i1 and i2 are of same type, initializing one would initialize the other as well implicitly.

22) In C, "float" is single precision floating type. "double" is double precision floating type. "long double"is often more precise than double precision floating type.  So the maximum floating type is "long double". There's nothing called "long long double". If someone wants to use bigger range than "long double", we need to define our own data type i.e. user defined data type. Besides, Type Specifiers "signed" and "unsigned" aren't applicable for floating types (float, double, long double). Basically, floating types are always signed only.

23)

**Drawback of C Language**

1) There is no index out of bound checking in C.

2) Also, In C, it is not compiler error to initialize an array with more elements than specified size. But it produces warning and shows error.

C++

To print the text within double quotes

// CPP program to print double quotes

#include<iostream>

int main()

{

std::cout << "\"geeksforgeeks\"";

return 0;

}

* **using**: You are going to use it.
* **namespace**: To use what? A namespace.
* **std**: The std namespace (where features of the C++ Standard Library, such as string or vector, are declared).

After you write this instruction, if the compiler sees string it will know that you may be referring to std::string, and if it sees vector, it will know that you may be referring to std::vector. (Provided that you have included in your compilation unit the header files where they are defined, of course.)

If you *don't* write it, when the compiler sees string or vector it will not know what you are refering to. You will need to explicitly tell it std::string or std::vector, and if you don't, you will get a compile error.

Print all natural numbers upto N without using semi-colon.  
We use the idea of recursively calling main function.

// CPP program to print all natural numbers upto

// N without using semi-colon

#include<iostream>

using namespace std;

int N = 10;

int main()

{

static int x = 1;

if (cout << x << " " && x++ < N && main())

{ } //No statements and no semicolon

return 0;

}