C Interview Questions

1)What is **C Language**?

a) C Programming Language is developed by Dennis Ritchie and Ken Thompson for use on the UNIX Operating System.

b) C Language is designed for Procedural Programming.

c) C is a high-level and general-purpose programming language that is an ideal language for developing portable system applications.

2)What are the **advantages** of C Language?

a)Simple

b)Portable

c)Structured Programming

d)Fast Speed

e)Memory Management

f)Extensible

3)What is the **Source Code**?

Source Code is a Code which is written by the Programmer in human Readable form with Proper Programming Syntax.

4)What is the **Executable Code**?

Executable Code is the machine-understandable code, which can be executed by a machine (OS).

5)What is **Object Code**?

Object Code is a Sequence of Statements in machine language, and is the output after the Compiler (or) Assembler Converts the Source Code.

6)What is a **Compiler**?

The compiler is a Software module which translate (convert) the Source Code Of a Program into Executable Code.

7)Difference between **Native Compiler** and **Cross Compiler**?

a) The Compiler used to compile source code for the same type of platform only.

The Compiler used to compile source code for different kinds of platform.

b) A Native Compiler generates codes for the same machine.

A Cross Compiler generates code for another machine.

c) Ex: -Turbo C, GCC

Ex:-C51,ARMCC

8)What are the **Compilation stages** and their **commands**?

a)**Preprocessor**

1) Responsible for including the header files and removal of comments.

2) Responsible for replacements of macros.

3) Responsible for Conditional Compilation.

Command	cc – e filename.c – o filename.i

b)**Translator**

1)Responsible for checking syntactical error and for converting source code into assembly code.

2) It will decide the scope (visibility).

Command	cc – s filename.i – o filename.s

c)**Assembler**

1) Responsible for converting assembly code into opcode.

Command	cc – c filename.s – o filename.o

d)**Linker**

1)Responsible for linking with libraries and adds operating System information.
2) It will create an executable file.
3) Whenever you get the undefined error it's generated by the linker.

Command	cc filename.o – o filename

9) What are the **types of Errors** occurred in C?

There are two types of errors
1)Compile-time
2)Run time
a)Compile Time Error:
This error is generated by Compiler.
a)Pre-Processor Error
b)Translator Error
c)Linker Error
b)Runtime Error
This error is generated by Operating System.
a)Segmentation fault
b)Bus Error
c) Floating-point Exception Error(Divide with zero)

10)What is **Code-Optimization**?

a)Code Optimization is a program transformation technique which tries to improve the code by making it consumes fewer resources so that faster running machine code will result.
b) Optimization should increase the speed and performance of the program.

11)Write a logic for **Set a Bit**, **Clear a Bit** and **Complement a bit?**

a) Set a Bit:

```
num=num | 1 << pos
```

b) Clear a Bit:

num=num & ~ (1 << pos)

c)Complement a Bit:

num=num ^ 1 << pos

12)What is a **Token**?

a)The Token is an identifier. It can be constant, keyword, string literal, etc.
A token is the smallest individual unit in a program.
b)C has the following tokens:

Identifiers: Identifiers refer to the name of the variables, arrays & functions.

Keywords: Keywords are the predefined words that are explained by the compiler.

Constants: Constants are the fixed values that cannot be changed during the execution of a program.

Operators: An operator is a symbol that performs the particular operation.

Special characters: All the characters except alphabets and digits are treated as special characters.

13)What is a **Variable?**

Variable is a name that can be used to store values, it can take different values but one at a time.

14)What is an **Expression?**

1)An Expression is a combination of operators, constants,

variables, and function calls.

2) An Expression can be arithmetic, logical (or) relational.

15)Difference between **While** and **Do-While**?

1) Condition is Checked first then the statement is executed.

Statement is executed at least once, therefore Condition

is checked.

2) No semicolon at the end of the while.

Semicolon at the end of the while.

3)While loop is an entry controlled loop

Do-While is exit controlled loop

16)D**ifference** between **Declaration** and **Definition?**

1) Declaration tells the Compiler about data type and size of the Variable.
Definition allocates memory to the Variable.
2) A Variable (or) the function can be declared any number of times.
A Variable (or) a function can be defined only once.
3) The memory will not be allocated during declaration
Memory will be allocated

17)**Difference** between **Local Variable** and **Global Variable**?

1) A variable which is declared inside function (or) block is known as local variables.
A Variable which is declared outside the function (or) block is known as global variable.
2) Variables are stored in Stack unless specified.
The Compiler decides the storage location of a Variable.

18)What is **Header File**?

Header File is a File that contains the declaration of a library function, global variables, and macro definitions.
The extension of the header file is ".h"

19)**Difference** between **Little Endian** and **Big Endian**?

1)Little Endian means that the lower-order byte of the number is stored in memory at the lowest address, and the higher order byte is stored at the highest address.
2)Big Endian means that the higher order byte of the number is stored in memory at the lowest address, and the lower order byte is stored at the highest address.

20)What is **Pointer**?

The pointer is one of the derived data the type which is useful for storing the address of another variable through which we can access the data indirectly.

21)What are the **Uses** of **Pointer**?

a) Accessing array elements
b) To get the address of a Variable.
c) Returning more than one value from a function.
d) Accessing Dynamically Allocated Memory.
e) Implementing data structures like Linked Lists, Trees.

22)What is **Void Pointer**?

a) Void Pointer is also known as Generic Pointer.
b) A Void Pointer is a Pointer that has no associated data type with it.
c) A Void Pointer can hold the address of any type and can be typecast to any type.
d) Void pointers cannot be dereferenced directly. If we want to dereference we need to typecast it. Because by typecasting it can be known to the compiler how many bytes it needs to be fetched.
e) The generic pointer can hold any type of pointers like char pointer, struct pointer, an array of pointers.
f) malloc () and calloc () return void*type and this allows these functions to be used to allocate memory of any datatype.
g) Void pointer in C is used to implement generic functions in C.

23)What is **Type-Casting**?

It is a way of Converting a variable from one data-type to another data-type.

Syntax : (type)expression

24)What is **NULL Pointer**?

a) If the pointer is holding zero as an address that pointer is called a NULL pointer.
b) A NULL pointer is a pointer that is pointing to nothing.
c)In case, if we don't have an address to be assigned to

a pointer, then we can simply use NULL.
d)If you try to dereference the NULL pointer, the result will be Segmentation fault.

25)What is **Dangling Pointer**?

a) After freeing the dynamic memory if the pointer is still pointing to the freed memory such pointers are called as Dangling pointer
b) To Avoid Dangling Pointer makes the pointer as NULL Pointer.

26)What is **Wild Pointer**?

a)A Pointer which has not been initialized to anything is known as a wild pointer.

b)The pointer may be initialized to a non-null garbage value that may not be a valid address.

c) Uninitialized pointers are known as wild pointers.

d)These Pointers usually point to some arbitrary memory locations and may cause a program to crash (or) misbehave.

27)How to **Avoid Wild Pointer**?

a) Initialize them with the address of a known variable.
b) Explicitly allocate the memory and put the values in the allocated memory.

28)What are the **advantages** of a **Pointer**?

a) Pointer reduces the code and improving the performance, it is used to retrieving string, trees.
b) We can return multiple values from a function using the pointers.

29)What is an **Array**?

The array is one of the derived data the type which is a collection of similar types of elements and which are in contiguous memory locations.

30)What is a **String**?

a) String is null-terminated character array.
b) Strings are collection of characters ended with null character ('\0').

31)Difference between A**rray** and P**ointer**?

a)A data structure consists of a collection of elements

each identified by the array index.

A Programming language object that stores the

memory address of another value located in the

computer memory.

b)

Syntax: data type variable name[elements];

Syntax : data type *variable name ;

c) Stores the value of the variable of homogeneous data type.

Store the address of the variable of same data type as the

pointer's a data type.

d) An array of pointers can be generated.

A pointer to an array can be generated.

e) Used to allocate fixed-size memory.

Used for dynamic memory allocation.

32)What is **Pointer to an Array**?

a) It is also known as the Array Pointer.
b) We are using the pointer to access the components of the array

Syntax: datatype(*var name) [size of array]

c) Ex: int (*a) [10]
Declares a pointer to an array of int.
d) The pointer to the array must be dereferenced to access the value of each element.

33)What is an **Array of Pointers**?

a) It is an Array of the Pointers variables.
b) It is also known as Pointer Array.

Syntax: data type *varname[Array Size]

c) Ex: int *a [10];
Declares and allocates an array of pointers.
d) Each element must be dereferenced Individually.

34)What are the **Functions** in C?

a) The function blocks where the program code is written.
b) The function is a set of instructions placed together to perform a specific task.
c) A function is a subroutine that is designed to perform one task.

35)What is the **use** of writing a **function in C** Programming Language?

a) C functions are used to avoid the rewriting the same code again and again in our program.
b) C functions can be called any number of times from any place of our program.
c)When a program is divided into functions, then any part of our program can easily be tracked.
d)C functions provide the reusability concept, i.e., it breaks the big task into smaller tasks so that it makes the C program more understandable.

36)What is the A**rgument**?

The calling functions send some values to the called function for communication. These values are called arguments (or) parameters.

37)What are the **Actual Arguments**?

a)The Arguments that are mentioned in the function call is known as actual Arguments.
b) These are the values which are actually sent to the called function.

38)What are the **Formal Arguments**?

a)The name of the arguments which are mentioned in the function definition is called formal (or) dummy arguments.

39)What is the difference between **Call By Value** and **Call By Reference?**

1)When a copy of the value is passed to the function, then the original value is not modified.
When a copy of the value is passed to the function, then the original value is modified.
2) Actual arguments and formal arguments are created in separate memory locations.
Actual arguments and formal arguments are created in the same memory location.
3) In this case, actual arguments remain safe as they cannot be modified.
In this case, actual arguments are not reliable, as they are modified.
4) The copies of the actual arguments are passed to the formal arguments.
The addresses of actual arguments are passed to their respective formal arguments.

40)What is **recursion in C**?

When a function calls itself and this process is known as recursion. The function that calls itself is known as a recursive function.

41)What are the A**dvantages** of **recursion**?

a) The use of recursion makes the code more compact and elegant.
b) It simplifies the logic and hence makes the program easier to understand.

42)What are the **Disadvantages of recursion**?

a) The recursive solution is always logical and

it is very difficult to trace.
b) The code written using recursion is less efficient.
Since recursion is a slow process because of many function calls involve overheads.
c) Recursion uses more processor time.

43)What are **Storage Classes**?

a)In addition to data type, each variable has one more

an attribute is known as a storage class.

b) The proper use of storage classes makes our program

efficient and fast.

c) We can specify a storage class while declaring a variable.

Stax :- storage class data type variable name

d)A Storage Class decides about these four aspects of a

variable

(a)Life Time --- Time between the creation and destruction

of a variable.

(b)Scope --- Locations where the variable is available for

use.

(c)Initial Value --- Default value taken by an uninitialized

variable.

(d)Place of Storage---Place in memory where the storage is

allocated for the variable.

44)What are the different **types** of **Storage Classes**?

a) Automatic
b) External
c)Static
d)Register

45)Explanation of **Storage Classes**?

Storage class	**Linkage**	**Storage**	**Initial Value**	**Scope**	**Life**
Auto	None	Stack	**Garbage**	With In Block	End of Block
Extern	External	Data Segment	**Zero**	Global Multiple Files	Till End of Program
Static	Internal	Data Segment	**Zero**	With In Block	Till End of Program
Register	None	CPU Register	**Garbage**	With In Block	End of Block

46)What are the types of **linkages** in C?

There are three types of linkages in C are -
(a) External Linkage
(b)Internal Linkage
1)Local Variables have no linkage, so their scope is only within the block where they are declared.
2)Global Variables and functions have external linkage, so they can be used in any file of the program.
3)Static Global Variables and Static functions have internal linkage so their scope is only in the file where they are declared.

47)What are the **command line arguments**?

a)The argument passed to the main() function while executing the program is known as command line argument.
b)Command-line arguments are the mechanism where we can supply the input at load time.

48)What is **Dynamic Memory Allocation**?

a) In the case of dynamic memory allocation, memory is allocated at runtime and memory can be increased while executing the program. It is used in the linked list.
b) The malloc() or calloc() the function is required to allocate the memory at the runtime.
c)An allocation or deallocation of memory is done at the execution time of a program.

d) No dynamic pointers are required to access the memory.
e) The dynamic memory is implemented using data segments.
f)Less memory space is required to store the variable.

49)What is **Static Memory Allocation?**

a) In the case of static memory allocation, memory is allocated at compile-time, and memory can't be increased while executing the program. It is used in the array.
b) The lifetime of a variable in static memory is the lifetime of a program.
c) The static memory is allocated using the static keyword.
d) The static memory is implemented using stacks or heap

50)**Difference** between **malloc()** and **calloc()** ?

Prototype: void *malloc(size_t size);
Prototype: void *calloc(size_t nmeb,szie_t size);

a) Assigns single block of demanded memory.
Assigns Multiple blocks of the requested memory.
b)The allocated memory is not initialized to zero.
The allocated memory is Initialized to zero by using calloc()
c) malloc is faster.
calloc is safer.

51)What is a **Memory leak**?

If we lose the base address of dynamically allocated memory that memory becomes a leak. (or) It occurs when programmers create a memory in heap and forget to delete it.

52)Define **Structure**?

Structure is one of the user-defined data types which is a collection of different types of data which are in contiguous memory locations.

53)What is **Self Referential Structure** in C?

In One Structure if one of the member is a pointer of the same structure type then that structure is called a Self Referential Structure.

54)What are **Structure Bit fields**?

a)Bit fields are a mechanism where we can allocate memory for storing the data in the form of "bits".

b) Bitfield variable can declare within a structure (or) with in a union. outside of them it is not possible to declare it.

55)What is **Structure Padding**?

a) Allocating extra bytes for a structure variable than the required one is called as structure Padding.
b) That extra unused byte is called as holes in a structure.
c) Structure padding is a compiler dependent concept.
d) Offset Calculation easy, and processing becomes faster.

56)How to **avoid** Structure Padding?

To Avoid Structure Padding, we can use Pragma Pack as well as an attribute.
Ex: - #Pragma pack (1)
Informing the Compiler whatever the data type it should be multiple of one.

57)What is meant by **union**?

The union is one of the user-defined datatype which is a collection of different types of data that are in the same memory location.

58)Difference between **Structure** and **Union**?

a) The keyword struct is used to define a structure.
The keyword union is used to define a union.
b) When a Variable is associated with structure, the compiler allocates the memory for each member. The size of the structure is greater then (or) equal to the sum of the size of Its members.
When a Variable is associated with the union, the compiler allocates the memory by Considering the size of the largest memory. So, the size of
the union is equal to the size of the largest member.
c) Each member within a structure is assigned a unique storage area of location.
Memory allocated is shared by individual members of the union.
d) An individual member can be accessed at a time.
Only one member can be accessed at a time.

59)What is **Typedef**?

Type def is one of the user-defined data types which is useful for providing another name to the existing data type.so,

those declarations can be simplified.

Syntax: typedef olddatatypename new name

60)What is **enum**?

a) enum is one of the user-defined datatype which is useful for providing a name to a Constants. So that understanding the program becomes an easy.
b) It is a Collection of named integer Constants means each element of enumeration is assigned by an integer value.

61)What is the **need of Enumeration**?

a) Enums can be declared in the local scope.
b)Enum names are automatically initialized by the compiler

62)What is a **file**?

a)A file is a named location which stores data (or) information permanently. A file is always stored inside a storage device using the file name.

b) The file is a collection of data that is present in the secondary memory.

63)What are **Macros**? What are its **advantages** and **disadvantages**?

a) Macros are processor the directive which will be replaced at compile-time by Pre-Processor.
c)The disadvantage with macros are that they just replace the code they do not function calls. So, the length of the code increases. Simultaneously it increases the execution time.
d) Similarly the advantage is they can reduce the time for replacing the same values.

64)Difference between **#define** and **typedef**?

a) It is a pre-processor directive.
It is a user-defined data type.
b) #define action takes place in the pre-processor stage.
Typedef action takes place in the translator stage.
c)Replacement takes place
No Replacement takes place

Syntax: #define macro name macro body

	Syntax : typedef olddatatypename new name	

65)What is a **debugger**?

A debugger or debugging tool is a a computer program that is used to

test and debug other programs.

66)In **header files** whether **functions** are **declared** or **defined**?

Functions are declared within the header file. That is function prototypes

exist in a header file, not function bodies. They are defined in the library (lib).

67)How do **signed** and **unsigned** numbers affect memory?

In the case of signed numbers, the the first bit is used to indicate whether positive or negative, this leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range an unsigned 8-bit the number has range 0-255, while the The 8-bit signed number has a range
-128 to +127.

68)What is **Garbage Collection**?

Periodic Collection of all the free memory space to form Contiguous

block of free space by an The operating system is called Garbage Collection.

69)What is **Conditional Compilation**?

a) Conditional Compilation directives help to compile a specific portion of the program (or) let us skip compilation of some specific part of the program based on some conditions.
b) It is used to reduce the size of an executable file.

70)**Difference** between **printf()** , **sprintf()** , **fprintf()**?

a) Print function is used to print character stream of data on stdout console.

Syntax: int printf (const char*,...)

b) String Print function instead of printing on console store it on

char buffer which is specified in sprintf.

Syntax: int sprintf(char*str,const char*string,..)

c)fprintf is used to print the string content in the file but not on stdout console.

Syntax: int fprintf (FILE*fptr, const char*str,...)

71)**Difference** between **scanf()**, **sscanf()**, **fscanf()**?

a) scanf () reads formatted input from stdin.

Syntax: int scanf (const char*,...)

b) sscanf () is used to read formatted input from the string.

Syntax: int sscanf(const char*s,const char * format ,...)

c) fscanf () reads formatted data from file and stores it into variables.

Syntax: int fscanf(FILE*stream,const char*format,...)

72)What is the **advantage** of the **register storage class**?

The main advantage of storing the variable as the register is they are stored in CPU memory which is accessed by very fast compared to RAM. This makes the program to get executed faster. Hence these types of variables are mainly used where quick access to them is required.

73)Which one is **faster n++** (or) **n+1** ?

The Expression n++ requires a single machine instruction. The Expression n+1 requires more instructions to carry out this operation. Hence n++ executes faster.

74)How to write a **running C code without main()**?

We will use Pre-processor directives #define with arguments to give an

impression that the the program runs without main. But in reality, it runs

with a hidden main function.

a) Using a macro that defines main

b) Using Token Pasting Operator

75)How can you **avoid including a header** more than once?

One easy technique to avoid multiple inclusions of the same header are to use the #ifndef, #define and #endif preprocessor directives. When you create a header for your the program, you can #define a symbolic name
that is unique to that header.

76)How to find the **size of a variable** without using the **sizeof() operator**?

```
#include<stdio.h>
#define MY_SIZE(var)   ((char*)(&var+1) - (char*) (&var))
int main()
{
   int a;
   printf("Value of my_size(a)=%d\n",MY_SIZE(a));
}
```

77)How to **print a semicolon** without **using a semicolon in the code**?

```
#include<stdio.h>

int main()

{

if(printf("%c",59))

{

}

}
```

78)How to **print 1 to 100** without **using a loop**?

```
#include<stdio.h>
void fun(n)
{
if(n<100)
{
n++;
printf("%d",n);
fun(n);
}
}
main()
{
int n=0;
fun(n);
printf("\n");
}
```

79)How to **deallocate memory** without **using free() in C**?

a) Standard Library function realloc() can be used to

deallocate previously allocated memory.

b) If the size is zero then call to realloc is equivalent to

free (ptr).

Ex: realloc(ptr,0);

80)How to **write a program containing two main() functions?**

```
#include<stdio.h>

void main()

{

printf("In 1st Main \n");

func1();

}

#define main func1

void main()
```

```
{
printf("In 2nd Main\n");
}
```

81)**Difference** between **#include<stdio.h>** and **#include"stdio.h"** ?

Both are same, but #include<stdio.h> will search
in predefined path only whereas #include “stdio.h"
searches in its present working directory.

82)What is a **main()** and **Difference** between **int main() and void main()**?

a) main () is an entry point which is in most programming
languages, when compiler begins to compile the program,
it looks for an entry point, and main () acts as an entry
the point in the C program

b) We can say main is thread/process/function that invokes
automatically by the compiler when the program is being
executed.

a) Every function returns a value to the calling
function, at that time main will be a called
function for compiler/os it will return some
value to the compiler before exit here void and
int defines that main will return a void (nothing)
and int will return an integer values to the compiler.

83)When should we use the **register storage specifier**?

If a variable is used most frequently then it should

be declared using register storage specifier, then

possible the compiler gives CPU for its storage

to speed up the lookup of the variable.

84) Tell some **situations where segmentation error** comes?

a) When try to de-referencing a NULL pointer.

b) Trying to access memory which is already deallocated.

c) Using Wild Pointers.

d) Accessing out of array index bounds

e)Stack Over Flow

f)Improper use of scanf()

85)What is **the Function Pointer**?

Function the pointer is like normal pointers but they

have the capability to point a Function (or)

A function pointer is a variable that stores the

address of a function that can later be called

through that function pointer.

86)What is the **use of a Function Pointer**?

Function Pointers can be useful when you want to create a call back mechanism, and need to pass the address of a function to another function.

87)What are the **advantages of Function Pointer**?

a) Unlike normal pointers, a function pointer points to code,

not data. Typically, a function pointer stores the start of

executable code.

b) Unlike normal pointers, we do not allocate deallocate memory using function pointers.

c) A function's name can also be used to get the function's address.

88)Difference between **sizeof() and strlen() ?**

sizeof()	**strlen()**
sizeof is a unary operator	strlen is a predefined function
It can be used with any data type	It can be used with only strings and character arrays
It gives the size in bytes	It just counts the length of the string
It includes slash zero	It doesn't include slash zero
Char str[]="tie" printf("%d",sizeof(str)) = 4	Char str[]="tie" Printf("%d",strlen(str))=3

89)What is meant by **the Reentrant function**?

a)A function is said to be reentrant if there is a provision to interrupt the function in the course of execution, service the interrupt service routine and then resume the earlier going on function, without hampering its earlier course of action.

b)Reentrant functions are used in applications like hardware interrupt handling, recursion, etc.

90)What is **Linker Error?**

Linker is to link the functions calls with function definitions.

If at this stage function definition not found, linker will generate error.

91)Tell some **situations where Linker error** comes?

1)Due to wrong function prototyping.

2)Incorrect Header Files.

3)One of the linker error is writing Main() instead of main()

4)Another linker error is writing print() instead of print().

92)What is the **L-value error?**

This error occurs when we put constants on the left-hand side of

Equal to Operator and variables on the right-hand side of it.

93)What is a **Bus Error?**

a) A bus occurs when a process is trying to access the memory that the CPU cannot physically address (or) memory tried to access by the program is not a valid memory address.

b) A bus error is a low-level hardware related error. When a program tries to access the memory location whose address is not present in the RAM, bus error generates.

94)What are the **causes** of the **bus error?**

Some common causes of bus errors are invalid file descriptors, unreasonable I/O requests, bad memory allocation, misaligned data structures, compiler bugs, and corrupt boot blocks.

95)What is **a.out** file?

a) a.out remains the default output file name for executables created by certain compilers and linkers
b) This is an abbreviated form of assembler output
c) It may store executable code shared libraries or object code.

96)How does **free() (Or) realloc()** know the size of memory to be **deallocated?**

When memory allocation is done, it allocates extra bytes to hold an integer containing the size of the block. This integer is located at the beginning of the block.

97)What is the **size of the empty structure in c?**

zero

98)Where do **macros get stored in the memory layout in C programming**?

1.Macros are not stored in memory anywhere in the final program but instead, the code for the macro is repeated whenever it occurs.

2.As far as the actual compiler is concerned they don't even exist, they've been replaced by the preprocessor before they get that far.

99)what is the **stack frame?**

The set of values pushed for one function call is termed as a stack frame

100)What are the **memory segments** of a **C Programming?**

1.Stack

2.Text segment

3.Data segment

4.Heap

101)What is meant by the **stack segment?**

1.Stack is of LAST IN FIRST OUT Structure.

2.It is used to store all local variables and auto variables

3.It is used for passing arguments to the functions along with the return address of the instruction which is used to be executed after the function call is over.

102)What is meant by **Text Segment?**

1.Text segment is also called a Code segment

2.It contains executable instructions

3.It is read-only and shared memory

103)What are the **types of data segments?**

There are two types of data segments

1.Initialized Data Segment

2.Uninitialized Data Segment

104)What is meant by **Initialized Data Segment?**

1.It is also called a Data Segment.

2.All the global, static, const, and extern variables initialized

by the program are stored in the Data Segment.

3.It is not read-only since it can be changed during run time.

4.It can further be classified as read-only area and read-write area

Example:

```
#include<stdio.h>
int a=15;
// Global Variable stored in Initialized data segment in read-write area //
const char  s[]="techinfoex"
//Global Variable stored in initialized data segment in read-only area//
const char * p="techinfoex"
//Global Variable stored in initialized data segment in read-only area//
int main()
```

```
{
static int b=16;   // Static variable stored in Initialized data segment //
}
```

105) What is meant by **Uninitialized Data Segment**?

1.It is also called as Block Started by Symbol(BSS).

2.Every member of this segment is initialized by the

kernel to zero before the program starts executing.

3.All the global and static which are not initialized by the

program are stored in this segment.

Example:

```
#include<stdio.h>
 char c; //   Uninitialized Global Variable //
int main()
{
static int i; // Uninitialized Static Variable//
}
```

106)What is meant by the **Heap segment?**

1.Heap is the segment where dynamic memory allocation

usually takes place.

2.Using malloc and calloc we can allocate memory in the heap segment.

3.The Heap area is shared by all shared libraries and dynamically

loaded modules in a process.

107)**Difference** between **++(*p) ,++*p,*(++p),*++p,*p++,*(p++) and (*p)++ ?**

1)*p represents the value stored in a pointer.

2)++ is an increment operator used in prefix and postfix expressions

3)* is a dereference operator

4) Precedence of prefix ++ and * is the same.

Associativity of both is right to left

5) Precedence of postfix ++ is higher than both * and prefix ++.

Associativity of postfix ++ is left to right

a)The expression ++(*p) has three operators. In this

Parentheses is having the highest precedence.

Here, the value is incremented at that particular location.

b) The expression ++*p has two operators of same

precedence, so the compiler looks for associativity.

Associativity of operators is right to left.

Therefore, the expression is treated as ++(*p).

c)The expression *(++p) has three operators. In this

Parentheses is having the highest precedence.

Here the pointing location is incremented to next

location and then the value is fetched.

d)The expression *++p has two operators of same

precedence, so the compiler looks for associativity.

Associativity of operators is right to left.

Therefore, the expression is treated as *(++p).

e) The expression *p++ has two operators. In this

postfix is having higher precedence.

So, pointing location is incremented to next

location and then the value is fetched.

f) The expression *(p++) has three operators. In this

Parentheses is having highest precedence.

Here, the pointing location is incremented to

next location and then the value is fetched.

g) The expression (*p) ++ has three operators. In this

Parentheses is having the highest precedence.

Here, the value is incremented at that particular location.

108)Exlain the following

a)**const int *ptr**

b)**int const*ptr**

c)**const int * const ptr**

d)**int *const ptr**

e)**int const *const ptr**

a)const int * ptr

ptr is a pointer to an integer constant

value cannot change

ptr can start pointing to other locations

b)int const *ptr

ptr is a pointer to a constant integer

value cannot change

ptr can start pointing to other locations

c)const int *const ptr

ptr is a constant pointer to an integer constant

value cannot be change

ptr cannot point to other variables once assigned

d)int *const ptr

ptr is a constant pointer to an integer

value can change

ptr cannot point to other variables once assigned

e) int const *const ptr

ptr is a constant pointer to an integer constant

value cannot be changed

ptr cannot point to other variables once assigned

109)What is the **use of return 0 in the main()** function?

return 0 indicates Successful "EXIT_STATUS" of the Program

'0' is the value returned by the main( ) function to the Operating System(OS) after the Program is Executed Successfully

But if during the Program Execution some "ERROR" occurs then instead of '0' some random value gets returned by the main( ) function to the Operating System. This indicates that some "ERROR" had occurred during the Program Execution