1. **What are the different data types available in ‘C’?**
   The basic data types available in ‘C’.
   int , float, char , double

2. **What are Keywords?**
   Keywords are certain reserved words that have standard and pre-defined meaning in ‘C’. These keywords can be used only for their intended purpose.

3. **What is an Operator and Operand?**
   An operator is a symbol that specifies an operation to be performed on operands.
   Example: *, +, -, / are called arithmetic operators.
   The data items that operators act upon are called operands.
   Example: a+b; In this statement a and b are called operands.

4. **What is Ternary operators or Conditional operators?**
   Ternary operators is a conditional operator with symbols ? and :
   Syntax: variable = exp1 ? exp2 : exp3
   If the exp1 is true variable takes value of exp2. If the exp2 is false, variable takes the value of exp3.

5. **What are the Bitwise operators available in ‘C’?**
   & - Bitwise AND
   | - Bitwise OR
   ~ - One’s Complement
   >> - Right shift
   << - Left shift
   ^ - Bitwise XOR are called bit field operators
   Example: k=~j; where ~ take one’s complement of j and the result is stored in k.

6. **What are the logical operators available in ‘C’?**
   The logical operators available in ‘C’ are
   && - Logical AND
   || - Logical OR
   ! - Logical NOT

7. **What is the difference between Logical AND and Bitwise AND?**
   Logical AND (&&): Only used in conjunction with two expressions, to test more than one condition. If both the conditions are true the returns 1. If false then return 0.
   AND (&): Only used in Bitwise manipulation. It is a unary operator.

8. **What is the difference between ‘=’ and ‘==’ operator?**
   Where = is an assignment operator and = = is a relational operator.
   Example:
   while (i=5) is an infinite loop because it is a non zero value and while (i==5) is true only when i=5.

9. **What is type casting?**

Type casting is the process of converting the value of an expression to a particular data type.

Example:

int x,y;

c = (float) x/y; where a and y are defined as integers. Then the result of x/y is converted into float.

10. **What is conversion specification?**

The conversion specifications are used to accept or display the data using the INPUT/OUTPUT statements.

11. **What is the difference between ‘a’ and “a”?**

‘a’ is a character constant and “a” is a string.

12. **What is the difference between if and while statement?**

if	**while**
(i) It is a conditional statement	(i) It is a loop control statement
(ii) If the condition is true, it executes some statements.	(ii) Executes the statements within the while block if the condition is true.
(iii) If the condition is false then it stops the execution the statements.	(iii) If the condition is false the control is transferred to the next statement of the loop.

13. **What is the difference between while loop and do…while loop?**

In the while loop the condition is first executed. If the condition is true then it executes the body of the loop. When the condition is false it comes of the loop. In the do…while loop first the statement is executed and then the condition is checked. The do…while loop will execute at least one time even though the condition is false at the very first time.

14. **What is a Modulo Operator?**

‘%’ is modulo operator. It gives the remainder of an integer division

Example:

a=17, b=6. Then c=%b gives 5.

15. **How many bytes are occupied by the int, char, float, long int and double?**

int - 2 Bytes char - 1 Byte float - 4 Bytes long int - 4 Bytes double - 8 Bytes

16. **What are the types of I/O statements available in ‘C’?**

There are two types of I/O statements available in ‘C’.

- Formatted I/O Statements
- Unformatted I/O Statements

17. **What is the difference between ++a and a++?**

    ++a means do the increment before the operation (pre increment) a++ means do the increment after the operation (post increment)

    Example:

    ```
    a=5;
    x=a++; /* assign x=5*/
    y=a; /*now y assigns y=6*/
    x=++a; /*assigns x=7*/
    ```

18. **What is a String?**

    String is an array of characters.

19. **What is a global variable?**

    The global variable is a variable that is declared outside of all the functions. The global variable is stored in memory, the default value is zero. Scope of this variable is available in all the functions. Life as long as the program's execution doesn't come to an end.

20. **What are the Escape Sequences present in 'C'**

    \n - New Line
    \b - Backspace
    \t - Form feed
    \' - Single quote
    \\ - Backspace
    \t - Tab
    \r - Carriage return
    \a - Alert
    \" - Double quotes

21. **Construct an infinite loop using while?**

    ```
    while (1)
    {
    }
    ```

    Here 1 is a non zero, value so the condition is always true. So it is an infinite loop.

22. **What will happen when you access the array more than its dimension?**

    When you access the array more than its dimensions some garbage value is stored in the array.

23. **Write the limitations of getchar( ) and sacnf( ) functions for reading strings**

    getchar( ) To read a single character from stdin, then getchar() is the appropriate.

    scanf( ) scanf( ) allows to read more than just a single character at a time.

24. **What is the difference between scanf( ) and gets() function?**

    In scanf( ) when there is a blank was typed, the scanf( ) assumes that it is an end.

    gets ( ) assumes the enter key as end. That is gets( ) gets a new line (\n) terminated string of characters from the keyboard and replaces the '\n' with '\0'.

25. **What is a Structure?**

Structure is a group name in which dissimilar data's are grouped together.

26. **What is meant by Control String in Input/Output Statements?**

Control Statements contains the format code characters, specifies the type of data that the user accessed within the Input/Output statements.

27. **What is Union?**

Union is a group name used to define dissimilar data types. The union occupies only the maximum byte of the data type. If you declare integer and character, then the union occupies only 2 bytes, whereas structure occupies only 3 bytes.

28. **What is the output of the programs given below?**

```
main()                              main()
{                                          {
float a;                                    float a;
int x=6, y=4;                               int x=6, y=4;
a=x\y;                                     a=(float) x\y;
printf("Value of a=%f", a);                 printf("Value of a=%f",a);
}                                          }
```

Output: 1.000000 **Output:** 1.500000

29. **Declare the Structure with an example?**

```
struct name
{
    char name[10];
    int age;
    float salary;
} e1, e2;
```

30. **Declare the Union with an example?**

```
union name
{
            char name[10];
            int age;
            float salary;
} e1, e2;
```

31. **What is the output of the following program when, the name given with spaces?**

```
main( )
{
char name[50]; printf("\n name\n"); scanf("%s, name); printf("%s",name);
}
```

Output: knreddy (if i/p is knreddy cpds notes) (It only accepts the data upto the spaces)

32. **What is the difference between while(a) and while(!a)?**

while(a) means while(a!=0)

while(!a) means while(a= =0)

33. **Why we don't use the symbol '&' symbol, while reading a String through scanf()?**

The '&' is not used in scanf() while reading string, because the character variable itself specifies as a base address.

Example: name, &name[0] both the declarations are same.

34. **What is the difference between static and auto storage classes?**

	Static	**Auto**
Storage	Memory	Memory
Initial value	Zero	Garbage value
Scope	Local to the block in which the variables is defined	Local to the block in which the variable is defined.
Life	Value of the variable persists between different function calls.	The block in which the variable is defined.

35. **What is the output of the program?**

```
   main( )                increment()
{                         {
increment();              static int i=1;
increment();              printf("%d\n",i)
increment();              i=i+1;
}                         }
```

OUTPUT:

1 2 3

36. **Why header files are included in 'C' programming?**

This section is used to include the function definitions used in the program.

Each header file has 'h' extension and include using '# include' directive at the beginning of a program.

37. **List out some of the rules used for 'C' programming.**

All statements should be written in lower case letters. Upper case letters are only for symbolic constants.

Blank spaces may be inserted between the words. This improves the readability of statements.

It is a free-form language; we can write statements anywhere between '{' and '}'.

```
 a = b + c;
d = b*c;
```

Opening and closing braces should be balanced.

38. **What do you mean by variables in 'C'?**

- A variable is a data name used for storing a data value.
- Can be assigned different values at different times during program execution.
- Can be chosen by programmer in a meaningful way so as to reflect its function in the program.
- Some examples are: Sum percent_1 class_total

39. **Differentiate break and continue statement**

break	**continue**
Exits from current block / loop Control passes to next statement Terminates the program	Loop takes next iteration Control passes to beginning of loop Never terminates the program

40. **Distinguish between while..do and do..while statement in C**

While..DO	**DO..while**
Executes the statements within the while block if only the condition is true.	Executes the statements within the while block at least once.
The condition is checked at the starting of the loop	The condition is checked at the end of the loop

41. **Compare switch( ) and nestedif statement.**

switch( ) case	**nested if**
Test for equality ie., only constant values are applicable.	It can equate relational (or) logical expressions.
No two case statements in same switch.	Same conditions may be repeated for a number of times.
Character constants are automatically converted to integers.	Character constants are automatically converted to integers.
In switch( ) case statement nested if can be used.	In nested if statement switch case can be used.

42. **Distinguish Increment and Decrement operators.**

Increment ++	**Decrement - -**
Adds one to its operand	Subtracts one from its operand
Equivalent x = x + 1	Equivalent x = x - 1
Either follow or precede operand	Either follow or precede operand
Example : ++x; x++;	**Example** : --x; x--;

43. **Give the syntax for the 'for' loop statement**

```
for (Initialize counter; Test condition; Increment / Decrement)
{
        statements;
}
```

- Initialization counter sets the loop to an initial value. This statement is executed only once.
- The test condition is a relational expression that determines the number of iterations desired or it determines when to exit from the loop. The 'for' loop continues to execute as long as conditional test is satisfied. When condition becomes false, the control of program exists the body of the 'for' loop and executes next statement after the body of the loop.
- The increment / decrement parameter decides how to make changes in the loop.
- The body of the loop may contain either a single statement or multiple statements.

44. **What is use of sizeof( ) operator?**

The sizeof ( ) operator gives the bytes occupied by a variable.

No of bytes occupied varies from variable to variable depending upon its data types.

Example:

```
int x,y;          printf("%d",sizeof(x));
```

Output:

```
2
```

45. **What is a loop control statement?**

Many tasks done with the help of a computer are repetitive in nature. Such tasks can be done with loop control statements.

46. **What are global variable in 'C'?**

This section declares some variables that are used in more than one function. Such variable are called as global variables. It should be declared outside all functions.

47. **Write a program to swap the values of two variables (without temporary variable).**

```
#include <stdio.h>
#include <conio.h>
void main( )
{
        int a =5; b = 10;
        clrscr( );
        prinf("Before swapping a = %d b = %d ", a , b);
        a = a + b;
        b = a – b;
        a = a – b;
        prinf("After swapping a = %d b = %d", a,b);
        getch( );
}
```

Output:

```
Before swapping a = 5 b = 10          After swapping a = 10 b = 5
```

48. **Write short notes about main ( ) function in 'C' program.**
    - Every C program must have main ( ) function.
    - All functions in C, has to end with '( )' parenthesis.
    - It is a starting point of all 'C' programs.
    - The program execution starts from the opening brace '{' and ends with closing brace '}', within which executable part of the program exists.
49. **What is meant by Recursive function?**

    If a function calls itself again and again, then that function is called Recursive function.

50. **What is an array?**

    An array is a group of similar data types stored under a common name.

    int a[10]; Here a[10] is an array with 10 values.

51. **What is a Pointer? How a variable is declared to the pointer?**

    Pointer is a variable which holds the address of another variable.

    Pointer Declaration:

    datatype *variable-name;

    Example:

    int *x, c=5;

    x=&a;

52. **What are the uses of Pointers?**
    - Pointers are used to return more than one value to the function
    - Pointers are more efficient in handling the data in arrays
    - Pointers reduce the length and complexity of the program
    - They increase the execution speed
    - The pointers saves data storage space in memory
53. **What are * and & operators means?**

    '*' operator means 'value at the address'

    '&' operator means 'address of'

54. **What is meant by Preprocessor?**

    Preprocessor is the program, that process our source program before the compilation.

55. **How can you return more than one value from a function?**

    A Function returns only one value. By using pointer we can return more than one value.

56. **Is it possible to place a return statement anywhere in 'C' program?**

    Yes. The return statement can occur anywhere.

57. **What are the main elements of an array declaration?**
    - Array name
    - Type and
    - Size

58. **List the header files in ‘C’ language.**
- **<stdio.h>** contains standard I/O functions
- **<ctype.h>** contains character handling functions
- **<stdlib.h>** contains general utility functions
- **<string.h>** contains string manipulation functions
- **<math.h>** contains mathematical functions
- **<time.h>** contains time manipulation functions

59. **What are the steps involved in program development life cycle?**
- Requirements
- Analysis
- Design
- Coding/Implementation
- Program Testing & Debugging

60. **What are the types of errors occurred in C program?**
- Syntax errors
- Runtime errors
- Logical errors
- Latent errors

61. **What is testing?**
Testing is the process of executing the program with sample or tested data.

62. **What are the types of testing?**
- Human testing
- Computer based testing

63. **How do you define enumerated data type?**
enum mar_status{ single,married,widow };
enum mar_status person1,person2;
person1=married; Here the person1 is assigned to value zero.

64. **What is meant by debugging?**
Debugging is the process of locating and isolating the errors.

65. **Specify any five syntax error messages.**
Missing semicolon, Missing braces, Missing quotes, improper comment characters, undeclared variables

66. **What are the pre-processor directives?**
Macro Inclusion , Conditional Inclusion , File Inclusion

67. **What is dynamic memory allocation?**
Allocating the memory at run time is called as dynamic memory allocation.

68. **What are the various dynamic memory allocation functions?**
- **malloc ( ) -** Used to allocate blocks of memory in required size of bytes.
- **free( ) -** Used to release previously allocated memory space.
- **calloc( ) -** Used to allocate memory space for an array of elements.
- **realloac( ) -** Used to modify the size of the previously allocated memory space.

69. **What is the deference between declaring a variable and defining a variable?**
- Declaring a variable means describing its type to the compiler but not allocating any space for it.
- Defining a variable means declaring it and also allocating space to hold the variable.

A variable can also be initialized at the time it is defined. To put it simply, a declaration says to the compiler,
- ✓ "Some where in the program there will be a variable with this name, and this is the kind of data type it is." On the other hand, a definition says, "Right here is this variable with this name and this data type". Note that a variable can be declared any number of times, but it must be defied exactly once. For this reason, definitions do not belong in header files, where they might get #included into more than one place in a program.

70. **Why does n++ execute than n=n+1?**

The expression n++ requires a single machine instruction such as INR to carry out the increment operation whereas; n+1 requires more instructions to carry out this operation.

71. **Why is it necessary to give the size of an array in an array declaration?**

When an array is declared, the compiler allocates a base address and reserves enough space in the memory for all the elements of the array. The size is required to allocate the required space. Thus, the size must be mentioned.

72. **Where in memory are variables stored?**

Variables can be stored in several places in memory, depending on their lifetime.
- Variables that are defined outside any function (whether of global or file static scope), and variables that are defined inside a function as static variables, exist for the lifetime of the program's execution. These variables are stored in the data segment. The data segment is a fixed-size area in memory set aside for these variables.
- Variables that are the arguments functions exist only during the execution of that function. These variables are stored on the stack. The stack is an area of memory that starts out as small and grows automatically up to some predefined limit.
- The third area is the one that does not actually store variables but can be used to store data pointed to by variables. Pointer variables that are assigned to the result of a call to the function malloc( ) contain the address of a dynamically allocated area of memory. This memory is in an area called the heap.

73. **What is an heap memory?**

The heap is another area that starts out as small and grows, but it grows only when the programmer explicitly calls malloc() or other memory allocation functions, such as calloc(). The heap can share a memory segment with either the data segment or the stack, or it can have its own segment, it all depends on the compiler options and operating system. The heap, like the stack, has a limit on how much it can grow, and the same rules apply as to how that limit is determined.

74. **What is the difference between an array and pointer?**

Difference between arrays and pointers are as follows.

Array	Pointer
1.Array allocates space automatically. 2.It cannot be resized. 3.It cannot be reassigned. 4.Size of(array name) gives the number of bytes occupied by the array.	1.Pointer is explicitly assigned to point to an allocated space. 2.It can be resized using realloc (). 3.Pointers can be reassigned. 4.Sezeof(pointer name) returns the number of bytes used to store the pointer variable.

75. **What is the purpose of the function main( )?**

The function main ( ) invokes other functions within it. It is the first function to be called when the program starts execution.

Some salient points about main() are as follows:

- It is the starting function .
- It returns an int value to the environment that called the program.
- Recursive call is allowed for main() also.
- It is a user-defined function.
- Program exection ends when the closing brace of the function main() is reached.
- It has two arguments (a) argument count and (b)argument vector (reprensents strings passed.) Any user-defined name can also be used as parameters for main( ) instead of argc and argv

76. **What is dangling pointer?**

In C, a pointer may be used to hold the address of dynamically allocated memory.

After this memory is freed with the free( ) function, the pointer itself will still contain the address of the released block. This is referred to as a dangling pointer. Using the pointer in this state is a serious programming error. Pointer should be assigned NULL after freeing memory to avoid this bug.

77. **Compare arrays and structures.**

Comparison of arrays and structures is as follows.

Arrays	Structures
An array is a collection of data items of same data type. Arrays can only be declared. There is no keyword for arrays. An array name represents the address of the starting element. An array cannot have bit fields.	A structure is a collection of data items of different data types. Structures can be declared and defined. The deyword for structures is struct. A structrure name is known as tag. It is a shorthand notation of the declaration. A structure may contain bit fields.

78. **Compare structures and unions.**

Structure	**Union**
Every member has its own memory. The keyword used is struct. All members occupy separate memory location, hence different interpretations of the same memory location are not possible. Consumes more space compared to union.	All members use the same memory. The keyword used is union. Different interpretations for the same memory location are possible. Conservation of memory is possible.

79. **Is it better to use a macro or a function?**
Macros are more efficient (and faster) than function, because their corresponding code is inserted directly at the point where the macro is called. There is no overhead involved in using a macro like there is in placing a call to a function.
However, macros are generally small and cannot handle large, complex coding constructs. In cases where large, complex constructs are to handled, functions are more suited, additionally; macros are expanded inline, which means that the code is replicated for each occurrence of a macro.

80. **List the characteristics of Arrays.**
All elements of an array share the same name, and they are distinguished form one another with help of an element number.
Any particular element of an array can be modified separately without disturbing other elements.

81. **What are the types of Arrays?**
One-Dimensional Array
Two-Dimensional Array
Multi-Dimensional Array

82. **What is the use of '\0' character?**
When declaring character arrays (strings), '\0' (NULL) character is automatically added at end. The '\0' character acts as an end of character array.

83. **Define sscanf() and sprint() functions.**
The sscanf():
This function allows to read character from a character Array and writes to another array. Similar to scanf(), but instead of reading from standard input, it reads from an array.
The sprintf():
This function writes the values of any data type to an array of characters.

84. **Define Strings.**

Strings:

The group of characters, digit and symnbols enclosed within quotes is called as Stirng (or) character Arrays. Strings are always terminated with '\0' (NULL) character. The compiler automatically adds '\0' at the end of the strings.

Example:

char name[ ]={'C','O','L','L','E','G','E','\0'};

The character of a string are stored in contiguous memory locations as follows:

C	O	L	L	E	G	E	\0

85. **What is the use of 'typedef''?**

It is used to create a new data using the existing type. Syntax: typedef data type name;

Example:

typedef int hours: hours hrs;/* Now, hours can be used as new datatype */

86. **What is 'C' functions? Why they are used?**

A function is a self-contained block (or) a sub-program of one or more statements that performs a special task when called. To perform a task repetitively then it is not necessary to re-write the particular block of the program again and again. The function defined can be used for any number of times to perform the task.

87. **Differentiate library functions and User-defined functions.**

Library Functions	**User-defined Functions**
Library functions are pre-defined set of functions that are defined in C libraries. User can only use the function but cannot change (or) modify this function.	The User-defined functions are the functions defined by the user according to his/her requirement. User can use this type of function. User can also modify this function.

88. **What are the steps in writing a function in a program.**

a) Function Declaration (Prototype declaration):

Every user-defined functions has to be declared before the main().

b) Function Callings:

The user-defined functions can be called inside any functions like main(), user-defined function, etc.

c) Function Definition:

The function definition block is used to define the user-defined functions with statements.

89. **What is a use of 'return' Keyword?**
90. The 'return' Keyword is used only when a function returns a value.

91. **Classify the functions based on arguments and return values.**

Depending on the arguments and return values, functions are classified into four types.

a) Function without arguments and return values.
b) Function with arguments but without return values.
c) Function without arguments but with return values.
d) Function with arguments and return values.

92. **Distinguish between Call by value Call by reference.**

Call by value	**Call by reference.**
In call by value, the value of actual agruments is passed to the formal arguments and the operation is done on formal arguments. Formal arguments values are photocopies of actual arguments values. Changes made in formal arguments valued do not affect the actual arguments values.	In call by reference, the address of actual argurment values is passed to formal argument values. Formal arguments values are pointers to the actual argument values. Since Address is passed, the changes made in the both arguments values are permanent.