Technical Aptitude Workshop (C-Language)

3 Days Session on C Language

Prof. T. R. Mahore

Description

This course is designed for understanding what and how to prepare before facing Interviews in IT Industry. The contents of this course is focused on C-Language. Specifically this particular document possesses questions related to C-Language that are generally asked in Tests or Interviews. These questions are prepared in such a way that one can start with the basics and then go up to the advanced level. This particular document is divided into two parts:

1. Fist Part Contains Beginenr to Advanced Level Interview Questions
2. Second Part Contains Code Examples which either can be asked in Tests or Interviews

Questions and Answers (Day 1)

Beginner Level Questions

Q.1. What are the Basic Datatypes Supported in C Language?

They are categorised in 4 different Categories :

1. Basic Datatypes
2. Derived Datatypes
3. Enumeration Datatypes
4. Void Datatypes

Datatype Name	Datatype Size	Datatype Range
short	1 byte	-128 to +127
unsigned short	1 byte	0 to 255
char	1 byte	-128 to 127
unsigned char	1 byte	0 to 255
int	2 bytes	-32,768 to 32,767
unsigned int	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,648
unsigned long	4 bytes	0 to 4,294,967,295
float	4 bytes	3.4E-38 to 3.4E+38
double	8 bytes	1.7E-308 to 1.7E+308
long double	10 bytes	3.4E-4932 to 1.1E+4932

Q.2. What do you mean by a Pointer Variable?

Pointer variable is the varibale which is decleared to store the address of the actual variable which is already decleared in the program. The pointer variable always starts with a '*' symbol. **Example:**

```
int x = 10              | int *p = &x
```

```
[x : Name]              | [*p : Name]
[10 : Data]             | [**&x i.e. address (10245) : Data**]
[**10245 : Address**]   | [32987 : Address of Pointer Variabl]
```

Q.3. What do you mean by Scope of the Variable?

Scope of the variable can be defined as the part of the code area where the variables declared in the program can be accessed directly. In C, all identifiers are lexically (or statically) scoped.

```c
#include<stdio.h>

// function prototype, also called function decleration
float square(float x);
// main function, program starts from here

int main() {
  float m = 10,n;
  // function call
  n = square(m);
  printf("Square of the given number %f is %f",m,n);
  return 0;
}

// function defination
float square(float x) {
  float p;
  p = x * x;
  return p;
}
```

```
Square of the given number 10.000000 is 100.000000
```

Q.4. What are Static Variables and Functions?

Static variables and static functions are almost the same variables and functions that we use in the C Programming Language. But the only difference is that we declare these variables and functions along with the 'static' keyword. The syntax of the declearation of static variable and function is as follows :

```
static data_type var_name = var_value;
Eg: static int x = 10;
```

```c
#include<stdio.h>

int fun() {
  static int count = 0;
  count++;
  return count;
}

int main() {
  printf("%d ",fun());
  printf(" %d",fun());
  return 0;
}
```

```
1  2
```

Q.5. Differentiate between calloc() and malloc()?

calloc() and malloc() are both dynamic memory allocation functions, along with them we have two more dynamic memory allocation functions i.e. realloc() and free(). The difference between them is that calloc() will load all the assigned memory locations with value 0 but malloc() will not.

Q.6. Where can we apply break control statement?

Break is the control statement which tells the compiler to break the execution of the program at that particular point. Break Control Statement is valid to be used inside loop and Switch Control Statements.

```
#include<stdio.h>

int main() {
  char opt;
  float x,y;

  //printf("Enter an Operator");
  //scanf("%c",opt)
  //printf("\nEnter two operands");
  //scanf("%ld %ld",x,y)

  opt = '+';
  x = 10, y = 20;

  switch(opt) {
  case '+':
    printf("%.1lf + %.1lf = %.1lf",x,y,x+y);
    break;
  case '-':
    printf("%.1lf + %.1lf = %.1lf",x,y,x-y);
    break;
  case '*':
    printf("%.1lf + %.1lf = %.1lf",x,y,x*y);
    break;
  case '/':
    printf("%.1lf + %.1lf = %.1lf",x,y,x/y);
    break;
    //operator doesn't match any case constant +,-,*,/
  default:
    printf("Error Operator is not Correct");
  }
  return 0;
}
```

```
10.0 + 20.0 = 30.0
```

Q.7. How can a negative integer be stored?

To store the negative number we have to use the one's compliment method:

```
Example : 1011 (-5)
Step 1 - One's compliment of 5: 1010
Step 2 - Add 1 to above, giving 1011, which is -5

int x = 0b1011;
```

Q.8. Differentiate between Actual and Formal parameters.

The parameters which are sent from main function to the subdivided function are called as **Actual Parameters** and the parameters which are declared inside the subdivided function end are called as **Formal Parameters**.

Q.9. Can I compile C program without main()?

The program will be compiled but will not be executed. To execute any C program, main() is required.

Q.10. What is a Nested Structure?

When a data member of one structure is reffered by the data member of another function, then the structure is called **Nested Structure**.

```
#include<stdio.h>

struct address {
  int city;
  int pin;
  int phone;
};

struct employee {
  int name;
  struct address add;
};

int main() {
  struct employee emp;
  //printf("Enter Employee Information\n");
  //scanf("%s %s %d %s",emp.name,emp.add.city, &emp.add.pin, emp.add.phone);
  emp.name = 123;
  emp.add.city = 456;
  emp.add.pin = 789;
  emp.add.phone = 000;

  printf("Name: %d City: %d Pincode: %d Phone: %d",emp.name,emp.add.city,emp.add.pin,emp.add.phone);
  return 0;
}
```

```
Name: 123 City: 456 Pincode: 789 Phone: 0
```

Q.11. What is a C Token"?

Keywords, Constants, Special Symbols, Strings, Operators, Identifiers used in C program are referred to as **C Tokens**.

Q.12. What is a PreProcessor?

A Preprocessor Directive is considered as a built-in predefined function or macro that acts as a directive to the compiler and it gets exceted before the actual C Program is executed. The best example of preprocessor is '#include<stdio.h>', the first line of the program while gets compiled, the compiler understands that it must provide 'standard input and output' functions in order to execute the program.

Q.13. How is C the Mother of all Languages?

C introduced many core concepts and data structures like **arrays, lists, functions, strings,** etc. Many languages designed after C are designed on the basis of C Language.

Q.14. Mention the Features of C Language?

1. High Level Language
2. Structured Language
3. It has Rich Library
4. It is Extensible
5. It supports Recursion
6. It has Pointers
7. It is Faster
8. Best Memory Management

Q.15. What is the use of printf() and scanf()?

printf() is used to print the values on the screen, and on the other hand **scanf()** is used to scan the values. We need an appropriate datatype format specifier for both printing and scanning purpose.

Q.16. What is an Array?

Array is a datastructure which stores similar kind of values in a sequential order. There are mainly three types of arrays one dimensional array, two dimensional array and multi dimensional array.

```
#include<stdio.h>

int main() {
  int i = 0;
  int id[4];
  id[0] = 1001;
  id[1] = 1002;
  id[2] = 1003;
  id[3] = 1004;
  for(i=0;i<4;i++) {
    printf("%d ",id[i]);
  }
  return 0;
}
```

Q.17. What is \0 character?

The symbol mentioned is called **Null Character**. It is considered as the terminating charcter used in strings to notify the end of the string to the compiler.

```
#include<stdio.h>
#include<string.h>
int main() {
  int length;
  char ch[20] = {'B','A','T','M','A','N','\0'};
  printf("Length of a String is: %d",strlen(ch));
  return 0;
}
```

```
Length of a String is: 6
```

Q.18. Differentiate between Compiler and an Interpreter?

Compiler translates the complete code into the Machine Code in one shot. Interpreter is designed to compile code in line by line fashion.

Q.19. Can I use int datatype to store 32768 value?

No Integer datatype will support the range between -32768 and 32767. Any value exceeding that will not be stored. We can either use **float** or **long int**.

Intermediate Level Questions

Q.1. How is a Function Decleared in C Language?

```
return_type function_name(formal parameters)
{
function_body;
}
```

Q.2. What is Dynamic Memory Allocation? Mention it's syntax?

Dynamic memory allocation is the method which is used to allocate memory to the program in **Run Time**.

```
ptr = (cast_type*)malloc(bytesize);
ptr = (cast_type)calloc(n,element-size);
ptr = realloc(ptr,newsize);
free(ptr)
```

```
#include<stdio.h>
#include<stdlib.h>

int main() {
  int *ptr;
  int n, i, sum = 0;
  n = 5;
  //printf("Enter No. of Elements: %d\n",n)
  n = 10;
  ptr = ((int*)malloc(n * sizeof(int)));
  if(ptr == NULL) {
    printf("Memory not Allocated");
    exit(0);
  }
  else {
    printf("Memory Successfully Allocated using malloc ");
    for(i = 0; i < n; i++) {
      ptr[i] = i + 1;
    }
    printf("The elements of array are: ");
    for(i = 0; i < n; i++) {
      printf("%d, ",ptr[i]);
    } }
  return 0;
}
```

```
Memory Successfully Allocated using malloc The elements of array are: 1  2  3  4  5  6  7  8  9  10
```

Q.3. Where can we not use &(address operator) in C?

We cannot use **&** on **constants** and on a **variable**, which is declared using the **register storage** class.

Q.4. Write an example for Structure in C language?

Structure is defined as a user-defined data-type that is designed to store multiple data members of different data types as a single unit.

```
#include<stdio.h>

int main() {
  char names[2][10], dummy;
  int roll_numbers[2],i;
  float marks[2];
  //for(i=0;i<3;i++) {
  //printf("Enter the Name, Roll No., and Marks of the student %d",i + 1);
  //scanf("%s %d %f",&names[i],&roll_numbers[i],&marks[i]);
  //scanf("%c",&dummy);
  //}
  names[0] = "Batman", roll_numbers[0] = 001, marks[0] = 80;
  names[1] = "Superman", roll_numbers[1] = 002, marks[1] = 90;
  for(i=0;i<2;i++) {
    printf("%s %d %f",names[i],roll_numbers[i],marks[i]);
  }
  return 0;
}
```

Q.5. Difference between CallByValue and CallByReference?

Factor	Call by Value	Call by Reference
Saftey	Actual arguments cannot be changed and remain safe	Operations are performed o
Memory Location	Seperate memory locations are created for Actual and Formal arguments	Actual and Formal argumer
Arguments	Copy of actual arguments are sent	Actual arguments are passe

Example: Call By Value

```
#include<stdio.h>
int change(int num) {
  num = num + 100;
  printf("Inside Function Number %d ",num);
  return 0;
}

int main() {
  int x = 100;
  printf("Before Function Call ");
  change(x);
  printf("After Function Call");
  return 0;
}
```

```
Before Function Call Inside Function Number 200 After Function Call
```

Example: Call by Reference

```c
#include<stdio.h>

int swap(int, int);

int main() {
  int a = 10;
  int b = 20;
  printf("Before swapping the values in main a = %d, b = %d ",a,b);
  swap(a,b);
  printf("After swapping values in main a = %d, b = %d",a,b);
  return 0;
}

int swap(int a, int b) {
  int temp;
  temp = a;
  a = b;
  b = temp;
  printf("After Swapping a = %d & b = %d",a,b);
}
```

Before swapping the values in main a = 10 b = 20 After Swapping a = 20 & b = 10After swapping values in main a = 10

Q.6. Differentiate between getch() and getche()

- **getch():** reads from the keyboard but does not use buffers. So, data is not displayed on the screen.
- **getche():** reads from the keyboard and it uses a buffer. So, data is displayed on the screen.

Q.7. Explain toupper() with an example

toupper() is a string function designed to convert lowercase words/characters into upper case.

```c
#include<stdio.h>
#include<string.h>
#include<ctype.h>

int main() {
  char str[20] = "batman";
  int i = 0;
  //printf("Enter the String");
  //gets(str);
  printf("String is %s ",str);
  // printf("Upper Case String is %s",strupr(str));
  while(str[i]) {
    putchar (toupper(str[i]));
    i++;
  }
  return 0;
}
```

```
String is batman BATMAN
```

Q.8. Write a simple code to generate Random Numbers in C

Example Code:

```c
#include<stdio.h>
#include<stdlib.h>

int main() {
  int c, n;
  printf("Ten Random Numbers in [1-100] ");

  for(c=1;c<=10;c++) {
    n = rand() % 100 + 1;
    printf("%d ",n);
  }
  return 0;
}
```

```
Ten Random Numbers in [1-100] 8 50 74 59 31 73 45 79 24 10
```

Q.9. Can I create Customized Header File in C?

It is possible to create a new header file. Create a file with function prototypes that need to be used in the program. Include the file in the '#include' section in its name.

Q.10. What do you mean by Memory Leak?

Memory leak can be defined as a situation where programmer allocates dynamic memory to the program but fails to free or delete the used memory after the completion if the code.

Q.11. Explain Static Local Variables and what is their use?

A local static variable is a variable whose life dosen't end with a function call where it is declared. It extends for the lifetime of the complete program.

Q.12. Differentiate between declaring header file with <> and ""

- < >: the compiler searches for the header file in the Built in Path
- " ": the compiler will search for the header file in the current working directory. If not found, it searches for the file in other locations.

Q.13. When should we use register storage specifier?

We use Register Storage Specifier if a certain variable is used very frequently. The variable will be decleared in one of the CPU registers.

Q.14. Which Statement is efficient and why? x = x + 1 / x++?

x++ is the most efficient statement as it's just a single instruction to the compiler while the other is not.

Q.15. Can I declare Same Variable in two different Scopes?

Yes, same variable name can be declared to the variables with different variables scopes as the following example. **Example Code:**

```c
#include<stdio.h>
#include<stdlib.h>

int var;

int function() {
```

```
  int variable;
  return 0;
}

int main() {
  int variable;
  return 0;
}
```

Q.16. How to access members of Union of Pointer types?

Arrow Operator (->) can be used to ccess the members of a Union if the Union Variable is declared as a Pointer Variable.

Q.17. Mention File Operations in C Language.

Basically we have four operations for Files:

Function	Operation
fopen()	To Open a file
fclose()	To Close a file
fgets()	To Read a file
fprintf()	To Write into a file

Q.18. What are the different Storage Class Specifiers in C?

1. auto
2. register
3. static
4. extern

Q.19. What is Typecasting?

Typecasting is a process of converting one data type into another.

Q.20. Explain Dangling Pointer in C Language

A **Pointer** in C Programming is used to point the memory location of an existing variable. In case if that particular variable is deleted and the pointer is still pointing to the same memory location, then that particular pointer variable is a **Dangling Pointer Variabe**.

Q.21. Swap two numbers without using a third variable

Example Code:

```
#include<stdio.h>
#include<stdlib.h>

int main() {
  int a = 10, b = 20;
  printf("Before Swap a=%d, b=%d ",a,b);
  a=a*b;
  b=a/b;
  a=a/b;
  printf("After Swap a=%d, b=%d",a,b);
```

```
  return 0;
}
```

Before Swap a=10 b=20 After Swap a=20 b=10

Advanced Level Questions

Q.1. Print a string with a % symbol in it.

There is no escape sequence provided for the symbol '%' in C. So, to print '%' we use '%%' in the program.

Q.2. Write the code to print the following pattern:

1 12 123 1234 12345

Example Code:

```
#include<stdio.h>

int main() {
  int i,j,rows;
  //printf("Enter the Number of Rows: ");
  //scanf("%d",&rows)
  rows = 5;
  for(i=0;i<=rows;i++) {
    for(j=1;j<=i;j++) {
      printf("%d",j);
    }
    printf("\n");
  }
}
```

1
12
123
1234
12345

Q.3. Explain #pragma Directive.

- This is a preprocessor directive used to turn on or off certain features
- It is of two types '#pragma startup', '#pragma exit', and pragma warn
- '#pragma startup' allows us to specify functions called upon program startup.
- '#pragma exit' allows us to specify functions called upon program exit
- '#pragma warn' tels the computer to supress any warning or not.

Q.4. How to remove duplicates in array?

Example Code:

```
#include<stdio.h>

int main()
{
```

```
  int arr[10] = {1,1,2,3,4}, i, j, k, Size;

  //printf("\n Please Enter Number of elements in an array  :   ");
  //scanf("%d", &Size);

  Size = 5;

  /*printf("\n Please Enter %d elements of an Array \n", Size);
    for (i = 0; i < Size; i++) {
    scanf("%d", &arr[i]);
    } */

  for (i = 0; i < Size; i++) {
    for(j = i + 1; j < Size; j++) {
      if(arr[i] == arr[j]) {
        for(k = j; k < Size; k++) {
          arr[k] = arr[k + 1];
        }
        Size--;
        j--;
      }
    }
  }

  printf("Final Array after Deleteing Duplicate Array Elements is: ");
  for (i = 0; i < Size; i++) {
    printf("%d\t", arr[i]);
  }
  return 0;
}
```

```
Final Array after Deleteing Duplicate Array Elements is: 1    2    3    4
```

Q.5. Explain Bubble Sort with a program.

Example Code:

```
#include <stdio.h>

int main()
{
  int array[100] = {24,11,34,55,6}, n, c, d, swap;

  //printf("Enter number of elements\n");
  //scanf("%d", &n);
  n = 5;
  /*printf("Enter %d integers\n", n);

    for (c = 0; c < n; c++)
    scanf("%d", &array[c]); */

  for (c = 0 ; c < n - 1; c++)
    {
      for (d = 0 ; d < n - c - 1; d++)
```

```
        {
          if (array[d] > array[d+1]) /* For decreasing order use '<' instead of '>' */
            {
              swap       = array[d];
              array[d]   = array[d+1];
              array[d+1] = swap;
            }
        }
    }

  printf("Sorted list in ascending order: ");

  for (c = 0; c < n; c++)
    printf("%d ", array[c]);

  return 0;
}
```

Q.6. Explain Round Robin Algorithm with Example.

Round Robin Algorithm is one of the algorithms employed by process and network schedulars in computing in order to evenly distribute resources in the system.

```
#include<stdio.h>

int main()
{

int count,j,n,time,remain,flag=0,time_quantum;
int wait_time=0,turnaround_time=0,at[10],bt[10],rt[10];
printf("Enter Total Process:\t ");
scanf("%d",&n);
remain=n;
for(count=0;count<n;count++)
{
printf("Enter Arrival Time and Burst Time for Process Process Number %d :",count+1);
scanf("%d",&at[count]);
scanf("%d",&bt[count]);
rt[count]=bt[count];
}
printf("Enter Time Quantum:\t");
scanf("%d",&time_quantum);
printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(time=0,count=0;remain!=0;)
{
if(rt[count]<=time_quantum && rt[count]>0)
{
time+=rt[count];
rt[count]=0;
flag=1;
}
else if(rt[count]>0)
{
rt[count]-=time_quantum;
time+=time_quantum;
```

```
}
if(rt[count]==0 && flag==1)
{
remain--;
printf("P[%d]\t|\t%d\t|\t%d\n",count+1,time-at[count],time-at[count]-bt[count]);
wait_time+=time-at[count]-bt[count];
turnaround_time+=time-at[count];
flag=0;
}
if(count==n-1)
count=0;
else if(at[count+1]<=time)
count++;
else
count=0;
}
printf("\nAverage Waiting Time= %f\n",wait_time*1.0/n);
printf("Avg Turnaround Time = %f",turnaround_time*1.0/n);

return 0;
}
```

Q.7. How do I link Program with Operating System?

- File links the operating system to the program
- The **file** is defined in the header file "stdio.h"
- It contains the information about:
  - The file being used
  - It's current **size**
  - It's **location** in memory

Q.8. How to avoid the limitations of scanf()

- **scanf()** cannot work with string of characters
- We cannot enter **multiword** string into **single** variable using scanf()
- To avoid this the **gets()** function is used

Q.9. Differentiate between Macros and Functions

- The **Macro** call makes the program run faster but also increases the program size
- Macro is simple and avoids **errors** related to the function calls
- In a function, call **control** is transfered to the function along with arguments. This makes the functions **small** and **compact**

Q.10. Suppose a global variable and a local Variable have the same name. Is it being possible to access a global variable from a block where local variables are defined?

No. It is not possible in C. It is always the most local variable that gets preference.

Guess the Output (Day 2 and Day 3)

Day 2

Q.1.

```c
#include<stdio.h>

#define x 5+2 //Macros will not execute or Macros are going to rplace

int main() {
  int i;
  i = x*x*x; //5+2*5+2*5+2
  printf("i is %d",i);
  return 0;
}
```

a. 343 b. 27 c. compiler error d. 21

Q.2.

```c
#include<stdio.h>

int main() {
  int n = 65;

  switch(n) {
  case 64:
    printf("Case 64");
    break;
  case 'A':
    printf("Case A");
    break;
  default:
    printf("Default Case");
  }
  return 0;
}
```

a. Case 64 b. Case A c. Default Case d. Compiler Error: Incompatible datatypes in case lable and switch expression

Q.3.

```c
#include<stdio.h>

int main() {
  int i = 1;
  switch(i) {
  case 1:
    printf("Hi");
  default:
    printf("Bye");
  }
  return 0;
}
```

a. Compilation Error b. Bye c. Hi d. HiBye

Q.4.

```
#include<stdio.h>

int main() {
  int i = 65;
  switch(i) {
  case 65:
    printf("Integer 65");
    break;
  case 'A':
    printf("Char 65");
    break;
  default:
    printf("Bye");
  }
  return 0;
}
```

a. Char 65 b. Integer 65 c. Bye d. Compilation Error

Q.5.

```
#include<stdio.h>

int main() {
  int i = 1;
  i++;
  switch(i--) {
  case 1:
    printf("Case 1 is Executed");
    break;
  case 2:
    printf("Case 2 is Executed");
    break;
  default:
    printf("Default Block is Executed");
  }
  return 0;
}
```

a. Case 2 is Executed b. Case 1 is Executed c. Default Block is Executed d. Compilation Error

Q.6.

```
#include<stdio.h>

int main() {
  int a=7;
  switch(a) {
  case 1:
  case 5:
  case 6:
    printf("Mumbai");
```

```
    break;
  case 2:
    printf("Amravati");
    break;
  case 3:
  case 7:
  case 10:
    printf("Nagpur");
    break;
  default:
    printf("Pune");
    break;
  }
  return 0;
}
```

a. Nagpur b. Pune c. Amravati d. Compiler Error

Q.7.

```
#include<stdio.h>

int main() {
  const int a = 1;
  const int b = 2;

  switch(a) {
  case a: //not valid if variable is passed
    printf("Mumbai");
    break;
  case b:
    printf("Amravati");
    break;
  case 3:
    printf("Nagpur");
    break;
  default:
    printf("Pune");
    break;
  }
  return 0;
}
```

a. Mumbai b. Amravati c. Nagpur d. Pune e. Compiler Error

Q.8.

```
#include<stdio.h>

int main() {
  switch(printf("hi")) {
  case 1:
    printf("Mumbai");
    break;
  case 2:
    printf("Amravati");
```

```
    break;
  case 3:
    printf("Nagpur");
    break;
  case 4:
    printf("Pune");
    break;
  default:
    printf("Delhi");
    break;
  }
  return 0;
}
```

a. hi b. Pune c. 4 d. hiPune e. Compiler Error

Q.9.

```
#include<stdio.h>

int main() {
  int a = 5;
  switch(a) {
  case 1:
    printf("Mumbai");
    break;
  case 2:
    printf("Amravati");
    break;
  case 3:
    printf("Nagpur");
    break;
  case 4:
    printf("Pune");
    break;
    //default:
    //  printf("Delhi");
    //  break;
  }
  return 0;
}
```

a. Compiler Error:Default Missing b. Empty Output Screen with Successful Compilation and Execution c. Compiler Error:Case 5 unavailable d. Pune

Q.10.

```
#include<stdio.h>

int main() {
  int a=3;
  switch(a) {

  }
  return 0;
}
```

a. error:case missing b. error:ambigious switch case c. Returns Garbage Value d. Empty O/P Screen with successful compilation and execution

Q.11.

```
#include<stdio.h>

int main() {
  if('A'<'a')
    printf("Hi");
  else
    printf("Bye");
  return 0;
}
```

a. Hi b. Bye c. Compilation Error d. Runtime Error

Q.12.

```
#include<stdio.h>

int main() {
  int k=30;
  printf("%d %d %d",k<=30,k=40,k==30);
  return 0;
}
```

a. 1 40 0 b. Compilation Error c. 0 40 1 d. Run Time Error

Q.13.

```
#include<stdio.h>
#define TEST 5

int main() {
  printf("TEST");
  return 0;
}
```

a. Compilation Error b. 5 c. TEST d. 10

Q.14.

```
#include<stdio.h>

int main() {
  int x=25;
  if(x=10)
    printf("TRUE");
  else
    printf("FALSE");
  return 0;
}
```

a. TRUE b. FALSE c. Error d. None

Q.15. Which of the following are incorrect Statement? If int a=10

1. if(a==10) printf("Hi")
2. if(10==a) printf("Hi")
3. if(a=10) printf("Hi")
4. if(10=a) printf("Hi")

Options a. 3 and 4 b. 3 only c. 4 only d. 1,3 and 4

Q.16.

```
#include<stdio.h>

int main() {
  int i=1;
  for(printf("Hi ");i<=3;i++) {
    printf("Hello Batman No. %d ",i);
  }
  return 0;
}
```

Write Down the Output

Q.17.

```
#include<stdio.h>

int main() {
  int i=1;
  for(printf("Hi ");i<=3;printf("Hello ")) {
    i++;
  }
  return 0;
}
```

Write Down the Output

Q.18.

```
#include<stdio.h>

int main() {
  int i=0;
  for(printf("Hi ");i<printf("ok")/*i<2*/;printf("Bye ")) {
    printf("No. %d ",i);
    i++;
  }
  return 0;
}
```

Write Down the output

Q.19.

```
#include<stdio.h>

int main() {
  int i=1;
```

```
  for( ;i<3; ) {
    printf("Hello Batman No. %d ",i);
    i++;
  }
  return 0;
}
```

Write Down the output

Q.20.

```
#include<stdio.h>

int main() {
  for(;;) {
    printf("Batman is Dangerous");
  }
  return 0;
}
```

Write Down the Output

Day 3

Q.21.

```
#include<stdio.h>

int main() {
  int i;
  for(i=0;i<=5;i++); {
    printf("Hi %d ",i);
  }
  return 0;
}
```

Write Down the Output

Q.22.

```
#include<stdio.h>

int main() {
  int _=1;
  int __=2;
  int ___=_+__;
  printf("%d %d %d",_,__,___);
  return 0;
}
```

a. Compilation Error b. 1 2 3 c. 3 2 1 d. None

Q.23.

```
#include<stdio.h>

int main() {
```

```
  int a=5,b=3;
  printf("%d",++(a*b+1));
  return 0;
}
```

a. Error b. 17 c. 15 d. None

Q.24.

```
#include<stdio.h>

int main() {
  int a = 5;
  a = printf("good") + printf("boy");
  printf("%d",a);
  return 0;
}
```

Write Down the Output

Q.25.

```
#include<stdio.h>

int main() {
  int a,b,c;
  c=4;
  a=b=c;
  c=a==b;
  printf("c:%d",c);
  return 0;
}
```

Write Down the Output

Q.26.

```
#include<stdio.h>

int main() {
  int i = 10;
  i = !i>14;
  printf("%d",i);
  return 0;
}
```

Write Down the Output

Q.27.

```
#include<stdio.h>

int main() {
  int i;
  i= 10+010+0x20;

  // 10 + (0*8^0 + 1*8^1) + (0*16^0 + 2*16^1)
```

```
  // 10 + 8 + 32

  printf("%d",i);
  return 0;
}
```

Write Down the Output

Q.28. Iterate Numbers from 1 to 10 without using any loop

```
#include<stdio.h>

int main() {

  int i=0;

  lable1: {
    if(i<10) {
      i++;
      printf("%d ",i);
      goto lable1;
    }
  }
  return 0;
}
```

Q.29. Represent Infinite Loop in GOTO

Q.30. Creating Infinite loop in a Function

```
#include<stdio.h>

int print();

int main() {
  print();
  return 0;
}

int print() {
  printf("This is a Function");
  print();
}
```

Q.31.

```
#include<stdio.h>

int main() {
  int a,x;
  a = 5, x = a++;
  printf("%d %d ",x,a);

  a = 5, x = ++a;
  printf("%d %d ",x,a);
```

```
  a = 5, x = a--;
  printf("%d %d ",x,a);

  a = 5, x = --a;
  printf("%d %d ",x,a);

  return 0;
}
```

Write Down the output

Q.32.

```
#include<stdio.h>

int main() {
  int a = 6;
  printf("%d ",a--);
  printf("%d ",++a);
  printf("%d ",a++);
  return 0;
}
```

Write Down the output

Q.33.

```
#include<stdio.h>

int main() {
  int x = 2;
  printf("%d %d %d",x*x,++x,x++);
  return 0;
}
```

Write Down the output

Q.34.

```
#include<stdio.h>

int main() {
  float me = 1.1;
  double you = 1.1;
  if(me==you)
    printf("Something is Wrong");
  else
    printf("Everything is Fine");
  return 0;
}
```

Write Down the output

Q.35.

```
#include<stdio.h>
```

```c
int main() {
  int x;
  x = 20;
  x*=30+5;
  printf("%d",x);
  return 0;
}
```

Write Down the output

Q.36.

```c
#include<stdio.h>

int main() {
  int x=1;
  if(x==2)
    printf("Hi ");
  printf("Hello");
  return 0;
}
```

Write Down the Output

Q.37. How many times 'Hi' will be Printed

```c
#include<stdio.h>

int main() {
  int i=1;

  while(i<10) {
    printf("Hello ");
  }
  printf("Hi ");
  return 0;
}
```

Q.38. How many times 'Hello' gets printed

```c
#include<stdio.h>

int main() {
  int a = 10, b = 20;

  do {
    printf("Hello ");
  }
  while(a>b);

  printf("Hi ");
  return 0;
}
```

Q.39.

```c
#include<stdio.h>

void f(int *p,int *q) {
  p=q;
  *p=2;
}

int i=0,j=1;

int main() {
  f(&i,&j);
  printf("%d %d",i,j);
  return 0;
}
```

Write Down the Output

Q.40. Whats wrong in the Given Code.

```c
#include<stdio.h>

int main() {
  int i, c = 0, n = 7;
  for(i=0;i<=10;i++) {
    if(n%i==0) {
      c++;
    }
    if(c==2)
      printf("Prime");
    else
      printf("Not Prime");
  }
  return 0;
}
```

Q.41.

```c
#include<stdio.h>

int main() {
  int x=10,y=15;
  if(++x>10||++y>15) {
    x++;
  }
  else {
    y++;
  }
  printf("x:%d y:%d",x,y);
  return 0;
}
```

Write Down the Output

Q.42.

```c
#include<stdio.h>

int main() {
  int x=10,y=15;
  if(++x<10 && ++y>15) {
    x++;
  }
  else {
    y++;
  }
  printf("x:%d y:%d",x,y);
  return 0;
}
```

Write Down the Output

Q.43.

```c
#include<stdio.h>

int main() {
  int a = 1, b = 2, c = 3, d = 4;
  int x;
  x = a = b = c = d;
  printf("%d",x);
  return 0;
}
```

Write Down the Output

Q.44.

```c
#include<stdio.h>

int main() {
  int x;
  x  = printf("Hello%d ",100);
  printf("%d",x);
  return 0;
}
```

Write Down the Output

Q.45.

```c
#include<stdio.h>

int main() {
  int x;
  x = printf("%d ",printf("Hello "));
  printf("%d",x);
  return 0;
}
```

Write Down the Output

Q.46

```c
#include<stdio.h>

int main() {
  int a;
  a = printf("Hi%dWolf",printf("Mumbai"));
  printf("%d",a);
  return 0;
}
```

Write Down the Output

Q.47.

```c
#include<stdio.h>

int main() {
  int x;
  x = printf("%d",printf("Hi"),printf("Hello"),printf("GoodMorning"));
  printf("%d",x);
  return 0;
}
```

Write Down the Output

Q.48.

```c
#include<stdio.h>

int main() {
  int i = 2, j = 2;
  while(i+1?--i:--j) {
    printf("%d ",i);
  }
  return 0;
}
```

Write Down the Output

Q.49.

```c
#include<stdio.h>

int main() {
  int x=3, y=3;
  while(--x&&--y) {
    printf(" %d %d",x,y);
  }
  return 0;
}
```

Write Down the Output

Q.50.

```c
#include<stdio.h>
```

```c
int main() {
  int x,y,z,a;
  y=2;
  if(x=y%2)
    z=2;
  a=2;
  printf("%d %d %d",z,x,a);
  return 0;
}
```