PROGRAMMING IN C

TUTORIAL ANSWERS

KS IGALAGAMA
30411

TUTORIAL 01

1. **Briefly explain the need of a programming language?**

   Programming languages are essential tools for software development, enabling humans to communicate their ideas and instructions to computers efficiently. They empower programmers to build complex systems, solve problems, and create innovative solutions across various domains.

2. **Compare and contrast the differences between followings;**

   a) Source Code vs. Machine Code

SOURCE CODE	MACHINE CODE
• A human-readable program Written by a programmer. • This is written in higher level languages. • Source code is easy to modify. • Instructions written using English words and according to syntax of the language.	• A machine executable program created after compiling a source program. • This program usually contains lower level languages. • Machine code is difficult to understand. • Instructions encoded in Binary digits.

   b) High-Level Language vs. Low-Level Language

HIGH-LEVEL LANGUAGE	LOW-LEVEL LANGUAGE
• High-Level Language is a human-friendly language that is easy to learn and understand. • No need for hardware knowledge for writing programs. • In these languages, modification of programs is easy for humans. Ex: python, PHP, c, c+, etc.	• Low-Level Languages are quite challenging for humans to learn and understand. • Hardware knowledge is necessary for writing programs. • In this language, modification of programs is quite difficult for humans. Ex: Machine Language Assembly Language

c) Compiler vs. Interpreter

COMPILER	INTERPRETER
• A compiler translates the entire source code in a single run. • Both syntactic and semantic errors can be checked simultaneously. • It consumes less time i.e., it is faster than an interpreter. • Ex: C , C++ , Java	• An interpreter translates the entire source code line by line. • Only syntactic errors are checked. • It consumes much more time than the compiler i.e., it is slower than the compiler. • Ex: Python , php , Ruby

d) Structured Language vs. Object Oriented Language

STRUCTURED LANGUAGE	OBJECT ORIENTED LANGUAGE
• It is a subset of procedural programming. • Programs are divided into small programs or functions. • Its main aim is to improve and increase quality, clarity, and development time of computer program. • It generally follows “top-down approach”.	• It relies on concept of objects that contain data and code. • Programs are divided into objects or entities. • Its main aim is to improve and increase both quality and productivity of system analysis and design. • It generally follows “Bottom-Up Approach”.

e) C vs. C ++

C	C ++
• C is a structural or procedural programming language. • C supports built-in data types. • C contains 32 keywords • C language does not support access modifier. • C is a middle level language	• C++ is a structural as well as an object-oriented programming language. • C++ support both built-in and user-defined data types. • C++ contains 63 keywords. • C++ support access modifier. • C++ is a high level language

f) C ++ vs. Java

C++	JAVA
• C++ is platform-dependent. • C++ is mainly used for system programming. • C++ supports operator overloading. • C++ supports structures and unions. • C++ use only compiler	• Java is platform independent. • Java is mainly used for application programming. It is widely used in Windows-based, web-based, enterprise, and mobile applications. • Java doesn't support operator overloading. • Java doesn't support structures and unions. • Java uses compiler and interpreter both.

g) Syntax error vs. Logical error

Syntax error	Logical error
• A syntax error occurs due to fault in the program syntax. • In compiled languages, the compile indicates the syntax error with the location and what the error is. • It is easier to identify a syntax error.	• A logical error occurs due to a fault in the algorithm. • The programmer has to detect the error by himself. • It is comparatively difficult to identify a logical error.

TUTORIAL 02

1. **How do you write comments in a c program? What is the purpose of comments in a program?**

   Single-line comment - // comment

   Multi-line comment - /* comment */

   Purpose of comments

   Comments can be used to explain code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

2. **What is the function that is essential in a C program?**

   main() function

3. **What is the purpose of 'scanf' ?**

   The scanf() function is a commonly used input function in the C programming language. It allows you to read input from the user or from a file and store that input in variables of different data types.

4. **Is 'standard c' a case sensitive language?**

   Yes.

5. **Determine which of the following are valid identifiers. If invalid, explain why.**

   (a) record1 – Valid.

(b) 1record - Invalid. Identifiers cannot start with a digit.

(c) file-3 - Invalid. Identifiers cannot contain hyphens or special characters other than underscores.

(d) return – Valid.

**(e) $tax -** Invalid. Identifiers cannot start with a dollar sign ($) or any special character other than an underscore.

(f) name – Valid.

(g) name and address - Invalid. Identifiers cannot contain spaces.

(h) name-and-address - Invalid. Identifiers cannot contain hyphens or special characters other than underscores.

(i) name_and_address – Valid.

(j) 123 - 45 – 6789 - Invalid. Identifiers cannot start with a digit, and they cannot contain hyphens or special characters other than underscores.

6. **State whether each of the following is true or false. If false, explain why.**
   a) **Function printf always begins printing at the beginning of a new line.**
      False, \n is necessary to create a new line. this is assumed
   b) **Comments cause the computer to print the text enclosed between /* and */ on the screen when the program is executed.**
      False, Text surrounded by /* and */ is ignored by computer
   c) **The escape sequence \n when used in a printf format control string causes the cursor to position to the beginning of the next line on the screen.**
      True
   d) **All variables must be defined before they're used.**
      True
   e) **All variables must be given a type when they're defined.**
      True

f) **C considers the variables, number and NuMbEr to be identical.**

   False. C is case sensitive, so these variables are unique

g) **A program that prints three lines of output must contain three printf statements.**

   False. New lines can be added by using “\n”

7. **What does the following code print?**

   printf("*\n\n***\n****\n*****\n");**

```
*
**
***
****
*****
```

8. **Identify and correct the errors in each of the following statements. (Note: There may be more than one error per statement.)**

   a) **scanf("d", value);**

      scanf(“%d”,&value);

   b) **printf("The product of %d and %d is %d"\n, x, y);**

      printf(“The product of %d and %d is %d\n”, x, y);

   c) **Scanf("%d", anInteger);**

      scanf(“%d”,&anInteger);

   d) **printf("Remainder of %d divided by %d is\n", x, y, x % y);**

printf(“Remainder of %d divided by %d is %d\n”, x, y, x %y);

e) **print("The sum is %d\n," x + y);**

printf(“The sum is %d\n”,x+y);

f) **Printf("The value you entered is: %d\n, &value);**

Printf(“The value you entered is: %d\n”,value);

9. **What, if anything, prints when each of the following statements is performed? If nothing prints, then answer “Nothing.” Assume x = 2 and y = 3 .**

a) **printf("%d", x);**

2

b) **printf("%d", x + x);**

4

c) **printf("x=");**

x=

d) **printf("x=%d", x);**

x=2

e) **printf("%d = %d", x + y, y + x);**

5 = 5

f) **z = x + y;**

Nothing

g) **scanf("%d%d", &x, &y);**

Nothing

h) **/* printf("x + y = %d", x + y); */**

Nothing

i) **printf("\n");**

Nothing

10. **State which of the following are true and which are false. If false, explain your answer.**

a) **C operators are evaluated from left to right.**

False

b) **The following are all valid variable names: _under_bar_ , m928134 , t5 , j7 , her_sales , his_account_total , a , b , c , z , z2 .**

True

c) **The statement printf("a = 5;"); is a typical example of an assignment statement.**

True

d) **A valid arithmetic expression containing no parentheses is evaluated from left to right.**

True

e) **The following are all invalid variable names: 3g , 87 , 67h2 , h22 , 2h**

True

TUTORIAL 03

Q1. Write four different C statements that each add 1 to integer variable x.

```
x = x + 1;

x += 1;

x++;

++x;
```

Q2. Write a single C statement to accomplish each of the following:

a) **Assign the sum of x and y to z and increment the value of x by 1 after the calculation.**

```
z = x++ + y;
```

b) **Multiply the variable product by 2 using the *= operator.**

```
product *= 2;
```

c) **Multiply the variable product by 2 using the = and * operators.**

```
product = product * 2;
```

d) **Test if the value of the variable count is greater than 10. If it is, print "Count is greater than 10."**

```
if ( count > 10 ) {
    printf( "Count is greater than 10.\n" );
  }
```

e) **Decrement the variable x by 1, then subtract it from the variable total.**

```
total -= --x;
```

f) **Add the variable x to the variable total, then decrement x by 1.**

```
total += x--;
```

g) **Calculate the remainder after q is divided by divisor and assign the result to q. Write this statement two different ways.**

```
q = q % divisor;

q %= divisor;
```

h) **Print the value 123.4567 with 2 digits of precision. What value is printed?**

```
printf( "%.2f", 123.4567 );

// 123.45 is displayed
```

i) **Print the floating-point value 3.14159 with three digits to the right of the decimal point. What value is printed?**

```
printf("%.3f", 3.14159 );
// 3.142 is printed
```

Q3. Write single C statements that

a) **Input integer variable x with scanf.**

```
scanf("%d", x );
```

b) **Input integer variable y with scanf.**

```
scanf("%d", y );
```

c) **Initialize integer variable i to 1.**

```
i = 1;
```

d) **Initialize integer variable power to 1.**

```
power = 1;
```

e) **Multiply variable power by x and assign the result to power.**

```
power = power * x;

  or

power *= x;
```

f) **Increment variable i by 1.**

```
i++;
```

g) **Test i to see if it's less than or equal to y in the condition of a while statement.**

```
if ( i <= y )
```

h) **Output integer variable power with printf**

```
printf("%d", power );
```

TUTORIAL 04

1) **What is wrong with the following if statement (there are at least 3 errors). The Indentation indicates the desired behavior.**

```
if numNeighbors >= 3 || numNeighbors = 4

++numNeighbors;

printf("You are dead! \n " );

else

--numNeighbors;
```

```
if (numNeighbors>=3 || numNeighbors==4)
{
        ++numNeighbors;
        printf("You are dead!\n");
}
else
{
--numNeighbors;
}
```

2) Describe the output produced by this poorly indented program segment:

```
int number = 4;
double alpha = -1.0;
if (number > 0)
if (alpha > 0)
printf("Here I am! \n" );
else
```

printf("No, I’m here! \n");

printf(“No, actually, I’m here! \n");

```
Output - No, I’m here!
         No, actually, I’m here!
```

3) **Consider the following if statement, where doesSignificantWork, makesBreakthrough, and nobelPrizeCandidate are all boolean variables:**

```
if (doesSignificantWork)
{
        if (makesBreakthrough)
        nobelPrizeCandidate = true;
        else nobelPrizeCandidate = false;
}
else if (!doesSignificantWork)
nobelPrizeCandidate = false;
```

4) **Write if statements to do the following:**

– If character variable taxCode is ’T’, increase price by adding the taxRate percentage of price to it.

```
if (taxCode == ‘T’)
        price = price + (price * taxRate / 100);
```

– If integer variable opCode has the value 1, read in double values for X and Y and calculate and print their sum.

```
if (opCode == 1)
{
        scanf("%lf%lf", &X, &Y);
        printf("Sum is %.2f", X+Y);
}
```

– If integer variable currentNumber is odd, change its value so that it is now 3 times currentNumber plus 1, otherwise change its value so that it is now half of currentNumber (rounded down when currentNumber is odd).

```
if (currentNumber%2 == 1)
        currentNumber = (currentNumber*3) + 1;
else
        currentNumber = currentNumber / 2;
```

– Assign true to the boolean variable leapYear if the integer variable year is a leap year. (A leap year is a multiple of 4, and if it is a multiple of 100, it must also be a multiple of 400.)

```
if ((year % 4 == 0) || (year % 100 != 0) && (year % 400 == 0))
        leapYear = 1;
```

– Assign a value to double variable cost depending on the value of integer variable distance as follows:

Distance	**Cost**
-----------------------------------	**----------**
0 through 100	**5.00**
More than 100 but not more than 500	**8.00**

More than 500 but less than 1,000	**10.00**
1,000 or more	**12.00**

```
if (distance >= 0 && distance <= 100)
        cost = 5.00;
else if (distance <= 500)
        cost = 8.00;
else if (distance <= 1000)
        cost = 10.00;
else
        cost = 12.00;
```

TUTORIAL 05

Switch

Input two numbers and display the outputs of the basic mathematic operations. The output screen should be displayed as follows;

Enter two numbers ____ ____

1. +

2. –

3. *

4. /

Please enter your Choice ___

```
#include <stdio.h>

 int main()
 {
        int n1,n2,choice,sum,sub,mul,div;
        printf("Enter two numbers: ");
        scanf("%d",&n1);
        scanf("%d",&n2);
        printf("1.+ \n2.- \n3.* \n4./ \n");
        printf("Enter your choice: ");
        scanf("%d",&choice);
        sum=n1+n2;
        sub=n1-n2;
        mul=n1*n2;
        div=n1/n2;
 switch (choice)
        {
                case 1:printf("sum = %d",sum);break;
                case 2:printf("substraction = %d",sub);break;
                case 3:printf("multiplication = %d",mul);break;
                case 4:printf("division= %d",div);break;
         }
}
```

While loop

1. **Input 10 numbers and display the total count of odd & even numbers in the entered number series.**

```
#include <stdio.h>

int main()
{
   int no,odd=0,even=0,counter=1;
   while (counter<=10)
   {   printf("Enter number %d: ",counter);
      scanf("%d",&no);
      if (no%2==0)
         even++;
      else
         odd++;
      counter++;
   }
   printf("Total count of odd numbers: %d\n",odd);
   printf("Total count of even numbers: %d\n",even);
}
```

2. **Modify the above program in to enter series of numbers terminates when the user enter -99 and display the same expected output.**

```
#include <stdio.h>


int main()

{

   int no,odd=0,even=0,counter=1;

   while(no!=-99 && counter<=10)

   {
       printf("Enter number %d: ",counter);
       scanf("%d",&no);
       if
       (no%2==0)
           even++;
       else
           odd++;
   counter++;
   }
   printf("Total count of odd numbers: %d \n",odd);
   printf("Total count of even numbers: %d \n",even);


}
```

Do while loop

Rewrite the programs for the above while loop question 1 & 2 using do while loop

1.

```
#include <stdio.h>

int main()
{
    int no,odd=0,even=0,counter=1;
    do
    {
        printf("Enter number %d: ",counter);
        scanf("%d",&no);
    if (no%2==0)
        even++;
    else
```

```
        odd++;
    counter++;
    }
    while (counter<=10);

    printf("Total count of odd numbers: %d \n",odd);
    printf("Total count of even numbers: %d \n",even);
}
```

2.

```
#include <stdio.h>

int main()
{
    int no,odd=0,even=0,counter=1;
    do
    {
        printf("Enter number %d: ",counter);
        scanf("%d",&no);
    if (no%2==0)
        even++;
    else
        odd++;
    counter++;
    }
    while(no!=-99 && counter<=10);

    printf("Total count of odd numbers: %d \n",odd);
    printf("Total count of even numbers: %d \n",even);
}
```

<u>For loop</u>

1. **Input 10 numbers and display the average value using the for loop**

```
#include <stdio.h>

int main()
{
   int no,counter=1,sum=0;
   float avg;
   for(counter=1;counter<=10;counter++)
   {
      printf("Enter Number %d:",counter);
      scanf("%d",&no);
      sum=sum+no;
   }
   avg=sum/10;
   printf("Average is %.2f ",avg);

}
```

2. Display the following output using the for loop

```
*

**

***

****

*****
```

```
#include <stdio.h>

int main()

{
        int x,y;

        for(x=1;x<=5;x++)

        {

                for(y=1;y<=x;y++)

                {

                        printf("*");

                }

        printf("\n");

        }


}
```