C TUTORIALS

Tutorial 1 :

1. Briefly explain the need of a programming language?

Languages like English, Chinese, etc. are used to communicate between two humans. In the same way a programming language is used to communicate between a human and computer.

Since computers can only understand 1's and 0's (binary format) and humans can understand only English language. So these programming languages can be used to translate the human requirements into computer understandable format.

2. Compare and contrast the differences between followings;

a) Source Code vs. Machine Code

Source code is text written in a computer programming language. source code is written by programmers. Machine code is a system of instructions and data executed directly by a computer's central processing unit. Usually compilers and interpreters convert the source code into machine code. machines can only understand 0 and 1.

b) High Level Language vs. Low Level Language

High level programming languages is languages program than use languages or syntax which close to human languages so, it is easy to understanding the languages or syntax such as pascal or delphi language program. while the low languages programming is languages that close to machine languages such as assembler languages.
Between both of them there are several "intermediate" languages program such as C++ etc. which stand between high and low languages program.

C)Compiler vs. Interpreter

	Compiler	Interpreter
1	Compiler Takes Entire program as input	Interpreter Takes Single instruction as input .
2	Intermediate Object Code is Generated	No Intermediate Object Code is Generated
3	Conditional Control Statements are Executes faster	Conditional Control Statements are Executes slower
4	Memory Requirement : More (Since Object Code is Generated)	Memory Requirement is Less
5	Program need not be compiled every time	Every time higher level program is converted into lower level program
6	Errors are displayed after entire program is checked	Errors are displayed for every instruction interpreted (if any)

7	Example : C Compiler	Example : BASIC

d) Structured Language vs. Object Oriented Language

Structured Programming	Object Oriented Programming
Structured Programming is designed which focuses on process/ logical structure and then data required for that process.	Object Oriented Programming is designed which focuses on data.
Structured programming follows topdown approach.	Object oriented programming follows bottom-up approach.
Structured Programming is also known as Modular Programming and a subset of procedural programming language.	Object Oriented Programming supports inheritance, encapsulation, abstraction, polymorphism, etc.
In Structured Programming, Programs are divided into small self contained functions.	In Object Oriented Programming, Programs are divided into small entities called objects.

e) C vs. C++

C	C++
C is a procedural (aka structural) programming language.	In addition to begin procedural, C++ is also an object oriented programming language.
In C language, the solution is achieved through a sequence of procedures or steps. Therefore, C is a function driven language.	C++ can model the whole solution in terms of objects and that makes the solution better organized. C++ is an object driven language.

f) C++ vs. Java

g) Syntax error vs. Logical error

Syntax Error

These involves validation of syntax of language.

Compiler prints diagnostic message.

Logical Error

Logical error are caused by an incorrect algorithm or by a statement mistyped in such a way that it doesn't violet syntax of language. Difficult to find

Tutorial 2 :

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

Adding // before the comment (for one line comment) or Text surrounded by /* and */

Used to describe program or for our own reference .(multi line comment)

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

int main() function

3.What is the purpose of 'scanf' ?

The function scanf() is used for formatted input from the standard input and provides many of the conversion facilities.

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 - First character should be alphabet or an underscore

(c) file-3 - Invalid - Dash symbol is not allowed

(d) return - Invalid – keywords cannot be used as identifiers

(e) $tax - Invalid - "$" symbol is not allowed

(f) name - Valid

(g) name and address - Invalid - Spaces not allowed between words

(h) name-and-address - Invalid - Dash is not allowed

(i) name_and_address - Valid

(j) 123 - 45 - 6789 - Invalid - First character should be alphabet or a underscore and spaces, dashes not allowed

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 use to beginning of a new line

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); 6 = 6

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, they are evaluated according to their precedence and associativity.

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.
   - False, this statement will just print a = 5; to the screen. It does not perform any assignment in the program.

d) A valid arithmetic expression containing no parentheses is evaluated from left to right.
   - False, it will be evaluated according to their precedence and associativity.

e) The following are all invalid variable names: 3g , 87 , 67h2 , h22 , 2h
   - False, h22 is a valid variable name. All of the other variable names start with digits and are therefore invalid variable names.

Tutorial 03:

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

x. x=x+1;

x+=1;

x++;

++x;

02. 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>10);
}
```

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

```
x--1;
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);
```

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

```
Printf("%.3f ",3.14159);
3.142
```

03.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*1;
```

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 4 :

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          Needs two equal signs before 4  ++numNeighbors;
No bracket on If statement
printf("You are dead! \n " );                     Curly braces are missing within else
if statement
--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");

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;
```

Tutorial 5 :

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.  /
5.Please enter your Choice ___
```

```
#include <stdio.h>

int main()
{
        float no1,no2;
      int choice;
        printf("Enter  two  numbers:\n");
      scanf("%f          %f",&no1,&no2);
```

```
        printf("1.+\n  2.-\n  3.*\n  4./\n");
        printf("Enter    your    choice\n");
        scanf("%d",&choice);

          switch(choice)
          {
             case 1: printf("addition is %.2f\n",no1+no2);break;
        case 2: printf("subtraction is %.2f\n",no1-no2);break;
        case 3: printf("multiplication is %.2f\n",no1*no2);break;
        case 4: printf("division is %.2f\n",no1/no2);break;
        default:printf("invalid input:");
          }
    return 0;
  }
```

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,c=1,odd=0,even=0;
while(c<=10)
  {
     printf("Enter %d number :",c);
scanf("%d",&no);       if(no%2=
=0)          odd=odd+1;       else
        even=even+1;
c++;
  }
   printf("The count of odd 's %d \n",odd);
printf("The count of even's %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.

Do while loop

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

```
#include <stdio.h> int
main()
{
  int no,c=1,odd=0,even=0;
do
  {
     printf("Enter %d number :",c);
scanf("%d",&no);       if(no%2==0)
odd=odd+1;
      else
       even=even+1;
      c++;
  }while(c<=10);
  printf("The count of odd 's %d \n",odd);
printf("The count of even's %d \n",even);
}
```

For loop

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

```
#include <stdio.h>
#include <stdlib.h> int
main()
{
  int no,c=1,total=0;
float avg;
for(c=1;c<=10;c++)
  {
     printf("enter %d number :",c);
scanf("%d",&no);       total=total+no;
    }
  avg=(float)total/10;    printf("The
average is %.2f\n",avg);
}
```

2.Display the following output using the for loop

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

int main()
{   int
x,y;
  for(x=1;x<=5;x++)
  {
     for(y=1;y<=x;y++)
     {
        printf("*");
```

```
    }
    printf("\n");
  }
}
```

Tutorial 6 :

Write a C program for the followings;

By using an array store the temperature values of a week.

1. Display the stored temperature values as follows;

Day	Temperature
1	26
2	28
3	32
4	27
5	31
6	29
7	30

```
#include <stdio.h>
int main()
{
   int i, temp [7], totalTemp=0, max,
min;    float avgTemp;    for (i=0;i<7;i++)
   {
      printf("Enter the Temperature of day %d :
",i+1);       scanf("%d",&temp[i]);       totalTemp =
totalTemp + temp [i];
```

```
    }
    printf("\nDay\tTemperature\n");

    for (i=0;i<7;i++)       printf("%d\t
%d\n",i+1,temp[i]);

    avgTemp = (float)totalTemp/7;    printf("\nAverage
temperature is  %.2f\n",avgTemp);    max=temp[0];
    for (i=0;i<7;i++)
        {
            if (temp [i] > max)
max = temp [i];
        }
    printf("\nHighest temperature is  %d\n",max);

    min=temp[0];
for (i=0;i<7;i++)
        {
            if (temp [i] < min)
min = temp [i];
        }
    printf("\nLowest temperature is  %d\n",min);
}
```

Tutorial 7 :

Write a C program for the followings;

Declare two 3 x 3 square matrices and display the matrix sum.

Following illustration shows the process of calculating the matrix sum. The values are used as samples.

3	2	4		2	6	3	=	5	8	7
1	4	6		4	3	2		5	7	8
4	3	2	+	5	1	7		9	4	9

```
#include <stdio.h> int
main()
{    int r,c,x[3][3],y[3][3],z[3][3];    printf("X is first 3x3 matrix\nY is second 3x3
matrix\nZ is matrix sum of X and Y\n\n");    for (r=0;r<3;r++)        for (c=0;c<3;c++)
        {
           printf("Enter a value for X (row %d, column %d) : ",r+1,c+1);
scanf("%d",&x[r][c]);
        }
    for (r=0;r<3;r++)
for (c=0;c<3;c++)
        {
           printf("Enter a value for Y (row %d, column %d) : ",r+1,c+1);
scanf("%d",&y[r][c]);
        }
    for (r=0;r<3;r++)
for (c=0;c<3;c++)
        {
           z[r][c]=x[r][c]+y[r][c];

           printf("value for Z (row %d, column %d) : %d\n",r+1,c+1,z[r][c]);
        }

  return 0;
}
```

Tutorial 8 :

1.Write a function that will read 2 numbers and calculate and display sum and difference.

```
#include <stdio.h> void

display()

{

  int no1,no2;    printf("Enter two

numbers:");    scanf("%d %d",&no1,&no2);
```

```
printf("The sum is %d \n",no1+no2);
printf("The differance is %d \n",no1-no2);
}
int main()
{
    display();
}
```

2.Write a function that accepts 2 numbers as parameters and calculate and display sum and difference.

```
#include <stdio.h>
void val(int a,int b)                                    {
int sum,diff;
printf("The sum is %d \n",a+b);
printf("The difference is %d\n",a-b);
}                                                        int
main()
{
   val(50,20);
}
```

OR

```
#include <stdio.h>
void val(int a,int b)
{
   int sum,diff;    printf("The sum is %d
\n",a+b);       printf("The difference is
%d\n",a-b);
}
int main()
```

```
{    int
a,b;
   printf("Enter two numbers:");
scanf("%d %d",&a,&b);
val(50,20);
}
```

3.Write a function that accepts 2 whole numbers as parameters and calculate and return the product.

```
#include <stdio.h> int
findProduct (int a, int b)
{
   int answer;
answer = a*b;
return answer;
}
int main ()
{    int
x,y;
   printf("Enter two numbers : ");    scanf("%d
%d",&x,&y);    printf("The product of %d
\n",findProduct(x,y));
}
```

4.Write a function that accepts 2 whole numbers as parameters and calculate and return the quotient.

```
#include <stdio.h>
int findQuotient (int a, int b)
{
   int answer=a/b;
return answer;
}
int main ()
{    int
x,y;
   printf("Enter two numbers : ");    scanf("%d
%d",&x,&y);    printf("The quotient of %d
\n",findQuotient(x,y));
}
```

5.Write a function to read 2 numbers and display the sum. Call this function from the main function several times.

```
#include <stdio.h> void
display()
{    int x,y,sum;    printf("Enter
two numbers : ");    scanf("%d
%d",&x,&y);    sum=x+y;
printf("Sum is %d\n\n",sum);
}
int main()
{
display();
display();
display(); }
```

6.Write a function which accepts 2 integers as parameters and display the sum, difference and product using a single printf statement.

```
#include <stdio.h>

 void display(int a,int b)
 {
    int
 sum,diff,product;
 sum=a+b;    diff=a-b;
 product=a*b;
    printf("The sum is %d\n difference is %d\n and product is %d\n",sum,diff,product);
 }
 int main()
 {    int
 a,b;
```

```
   printf("Enter two numbers:");
  scanf("%d %d",&a,&b);
  display(a,b);
  }
```

7.Write a function which accepts an integer and a float value as parameters and return the product as a double value. Display the result from the main function.

```
#include <stdio.h>
double display (int a, float b)
{
   double product;
product=a*b;    return
product;
}
int main()
{    int a;    float b;
printf("Enter two numbers : ");
scanf("%d %f",&a,&b);
printf("Product is
%.2f\n",display(a,b));
 }
```

8.Give the function header for each of the following functions.

a. Function hypotenuse that takes two double-precision floating-point arguments, side1 and side2, and returns a double-precision floating-point result. double hypotenuse(double a,double b)

b. Function smallest that takes three integers, x, y, z, and returns an integer.

   int smallest(int x, int y, int z)

c. Function instructions that does not receive any arguments and does not return a value. [Note: Such functions are commonly used to display instructions to a user.]

   int instructions ()

d. Function intToFloat that takes an integer argument, number, and returns a floatingpoint result.

float intToFloat (int x)