INDEX NO:30629

STUDENT NAME:PUSHPAKUMARA HLS

MODULE:PROGRAMMING WITH C LANGUAGE

TUTORIAL 01 (ANSWERS)

(01).

A programming language is a set of rules and syntax used to write instructions for a

computer to execute. It acts as a bridge between humans and computers, allowing

programmers to communicate their intentions and solve complex problems effectively.

(02).

a. \*Source Code vs. Machine Code:\*

- Source Code: It refers to the human-readable code written by programmers in a

programming language like C, Java, or Python. Source code is written using high-level constructs

and abstractions that are closer to natural language, making it easier for programmers to write

and understand.

- Machine Code: It consists of binary instructions directly understood and executed by the

computer's hardware. Machine code is specific to the computer architecture and is not easily

readable or writable by humans. It represents the lowest level of programming language.

b. \*High-Level Language vs. Low-Level Language:\*

- High-Level Language: These languages provide a high level of abstraction from the underlying

hardware. They are designed to be closer to human language, allowing programmers to write

code that is more readable, portable, and easier to understand. Examples include Python, Java,

and Ruby.

- Low-Level Language: These languages are closer to the hardware and provide minimal

abstraction. They are more closely related to machine code and are less readable and

portable. Low-level languages include assembly language and machine code.

 TUTORIAL 01

 INDEX NO : 30620

 STUDENT NAME: Sesanda M N S

 MODULE : Programming With C Language

1.

A programming language is a set of rules and syntax used to write instructions for a

computer to execute. It acts as a bridge between humans and computers, allowing

programmers to communicate their intentions and solve complex problems effectively.

2.

a. \*Source Code vs. Machine Code:\*

- Source Code: It refers to the human-readable code written by programmers in a

programming language like C, Java, or Python. Source code is written using high-level constructs

and abstractions that are closer to natural language, making it easier for programmers to write

and understand.

- Machine Code: It consists of binary instructions directly understood and executed by the

computer's hardware. Machine code is specific to the computer architecture and is not easily

readable or writable by humans. It represents the lowest level of programming language.

b. \*High-Level Language vs. Low-Level Language:\*

- High-Level Language: These languages provide a high level of abstraction from the underlying

hardware. They are designed to be closer to human language, allowing programmers to write

code that is more readable, portable, and easier to understand. Examples include Python, Java,

and Ruby.

- Low-Level Language: These languages are closer to the hardware and provide minimal

abstraction. They are more closely related to machine code and are less readable and

portable. Low-level languages include assembly language and machine code. Compiler: It is a software tool that translates the entire source code into

machine code before execution. The compiler performs various stages like lexical

analysis, syntax analysis, optimization, and code generation. Examples of compiled

languages include C, C++, and Go.

- Interpreter: It executes the source code line by line, translating and executing

each instruction in real-time. Interpreters are typically slower than compilers but

provide more flexibility and dynamic features. Python, JavaScript, and Ruby are

examples of interpreted languages.

d. \*Structured Language vs. Object-Oriented Language:\*

- Structured Language: It is a programming language that emphasizes the use of

structured programming techniques, such as loops, conditionals, and modular

programming. Examples of structured languages include C, Pascal, and Fortran.

- Object-Oriented Language: It is a programming paradigm that organizes code

around objects, which encapsulate data and behavior. Object-oriented languages

provide features like classes, inheritance, and polymorphism. Examples include

Java, C++, and Python.

e. \*C vs. C++:\*

- C: C is a procedural programming language known for its simplicity, efficiency,

and low-level programming capabilities. It provides a minimalistic set of features

and allows direct manipulation of memory. It is commonly used in systems

programming and embedded systems development.

- C++: C++ is an extension of the C programming language with added features,

primarily object-oriented programming. It introduces classes, inheritance, and

polymorphism while maintaining backward compatibility with C. C++ is widely

used for general-purpose software development, game development, and high-

performance applications. f. \*C++ vs. Java:\*

- C++: C++ is a statically typed, compiled language that offers high performance,

direct hardware access, and low-level control. It is popular for system-level

programming, game development, and performance-critical applications.

- Java: Java is a statically typed, platform-independent language that runs on the

Java Virtual Machine (JVM). It emphasizes portability, automatic memory

management, and a vast ecosystem of libraries and frameworks. Java is commonly

used for enterprise software, web development, and Android app development.

g. \*Syntax Error vs. Logical Error:\*

- Syntax Error: It occurs when code violates the rules of the programming

language's syntax. These errors are detected by the compiler or interpreter during

the compilation or execution phase. Syntax errors prevent the code from running

and must be fixed before the program can proceed.

- Logical Error: It occurs when the code does not produce the intended results

due to flaws in the logic or algorithm. Logical errors are more challenging to

detect as they do not cause the program to halt or produce error messages.

Debugging and careful analysis of the code are required to identify and fix logical

errors.

TUTORIAL 02(ANSWERS)

1. In C programming, comments are used to add explanatory or

descriptive information within the code. They are ignored by the

compiler and do not affect the program's execution.

Single-line Comments:

// This is a single-line comment

int x = 10; // Initializing a variable

Multi-line Comments:

/\*

This is a multi-line comment

It can span multiple lines

int x = 10; // This code is commented out

\*/

2.int main ()

3. The scanf function in C is used to read input from the user . It allows the

program to receive data interactively during runtime. The purpose of scanf is to

assign values to variables based on the user's input.

4. Yes

5. valid identifiers: record1, return, name, name\_and\_address

6. a) False

b) False

c) True

d) False

e) True

f) False

g) False

7. \*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

8. a) scanf( "d", value );

The format specifier in scanf is missing the % symbol, which is required

to specify the type of data to be scanned.

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

- The closing quotation mark is placed incorrectly before the newline

escape sequence (\n).

-The format control string in printf is incomplete and lacks the

necessary placeholders for variables.

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

The scanf function is written with a capital 'S', while it should be

written in lowercase.

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

The printf statement is missing a format specifier for the x % y

expression. Additionally, there is no variable assigned to store the result

of the remainder calculation.

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

-The function name should be printf instead of print.

- The comma after the format string is outside the quotation marks,

resulting in a syntax error.

- The sum of x and y is not enclosed within parentheses, causing a

syntax error.

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

-The printf function is written with a capital 'P', while it should be

written in lowercase.

-The closing quotation mark is placed incorrectly before the comma

after the format string.

-The ampersand (&) is unnecessary before value because scanf

requires the address of a variable, not printf.

9. a) 2

b) 4

c) x=

d) x=2

e) 5 = 5

f) Nothing

g) Nothing

h) Nothing

i) A new line

10.

a) True

b) True

c) False: The statement printf("a = 5;"); is not an assignment statement.

It is a printf statement that is used for printing text. The text "a = 5;" will

be printed as it is, but it does not perform any assignment to a variable.

d) True

e) True

TUTORIAL 03(ANSWERS)

1 .x=x+1;

.X+=1;

.X++;

.++x;

2. a) z=y + x++;

b) product\*=2;

c) product=product\*2;

d) if (count>10)

printf(“count is greater than 10”);

e) total-= --x;

f) total+= --x;

g) q=q%divisor;

q%=divisor;

h) printf(“%.2f\n”,123.4567);

i) printf(“%.3f\n”,3.14159);

3.a)scanf(“%d”,&x);

b)scanf(“%d”,&y);

c)int I = 1;

d)int power=1;

e)power\*=x;

f)i++;

g)while (i<=y)

h)printf(“%d\n”,power);

TUTORIAL 04(ANSWERS)

1.error 1 and 2: if numNeighbors >= 3 || numNeighbors = 4 should be

if (numNeighbors >= 3 || numNeighbors == 4)

error 3 :

2. No, I'm here!

No, actually, I'm here!

3. nobelPrizeCandidate = doesSignificantWork && makesBreakthrough;

4.

a. if (taxCode == 'T')

price += taxRate \* price ;

b. if (opCode == 1)

double X, Y;

printf("Enter the value of X: ");

scanf("%lf", &X);

printf("Enter the value of Y: ");

scanf("%lf", &Y);

double sum = X + Y;

printf("The sum of X and Y is: %lf\n", sum);

c. if (currentNumber%2==1)

currentNumber=(currrentNumber+1)\*3;

else

currentNumber=currentNumber/2

d.

e. double cost;

if (distance <= 100)

cost = 5.00;

else if (distance <= 500)

cost = 8.00;

else if (distance < 1000)

cost = 10.00;

else (distance >= 1000)

cost = 12.00;

TUTORIAL 05(ANSWERS)

2. 1

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

while(count<=10)

{

printf("enter %d number",count);

scanf("%d",&no);

if (no%2=0)

even++;

else

odd++;

count++;

}

printf("number of even %d\n",even);

printf("number of odd %d\n",odd);

2.1

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

while(count<=10)

{

printf("enter %d number",count);

scanf("%d",&no);

if(no==-99)

break;

if (no%2==0)

even++;

else

odd++;

count++;

}

printf("number of even %d\n",even);

printf("number of odd %d\n",odd);

3.

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

printf(" enter number %d", count);

do {

scanf("%d", &no);

if (no == -99)

break;

if (no % 2 == 0)

even++;

else

odd++;

count++;

} while (count <= 10);

printf("Number of even: %d\n", even);

printf("Number of odd: %d\n", odd);

4.1

int count,no,total=0;

float avg;

printf("enter number %d",count);

for(count=1;count<=10;count++)

{

scanf("%d",&no);

total=total+no;

}

avg=(float)total/10.0;

printf("average is %2f\n",avg);

2.

int x,y;

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

{

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

{

printf("\*");

}

printf("\n");

}