CHAPTER NO 2: -(USER INTERACTION)

SHORTS QUESTIONS/ANSWERS

1. ***Define I/O function?***

   Each programming languages has its keywords or standard library function for I/O operations. C language offer printf function to display the output and scanf function to take the input from the user.

2. ***Differentiate between printf () & scanf ()?***

Printf ()	***Scanf ()***
Printf is a built-in function in C language to show the output on the monitor screen. *Its name comes from "print formatted".* *Example: -* *printf("Hello world"):*	*Scanf is a built in function in C language that takes the input from the user into the variable. Its name comes from "scan formatted"* *Example: -* *int I;* *printf("Enter the value of I:\t);* *scanf("%d", &I);*

3. ***Differentiate between getch () & scanf ()?***

Getch ()	***Scanf ()***
Getch function is used to read the character from the user. The character entered by the user does not display on the screen. *Example: -* *printf("Enter any value of* *if u want to exit the program"):* *getch();*	*Scanf is a built in function in C language that takes the input from the user into the variable. Its name comes from "scan formatted"* *Example: -* *int I;* *printf("Enter the value of I:\t);* *scanf("%d", &I);*

4. ***What is the common mistake of scanf () function?***

   *It is a very common mistake to forget the **"&"** sign in the scanf function. Without **&** the program get executed but does not the behavior as expected.*

5. ***Define clrscr ()?***

   clrscr() is a function in C that is used to clear the console screen. It stands for "clear screen." This function is typically provided by the conio.h library, which is specific to some compilers like Turbo C/C++.It is used to remove all the contents displayed on the console and move the cursor to the top left corner of the screen.

   Example: -

```
#include <conio.h>
#include <stdio.h>
int main() {
clrscr(); // Clears the console screen
return 0;
}
```

6. Define format specifier?

A format specifier in programming languages like C is a placeholder within a string that specifies how an argument should be formatted. It is used in functions like `printf()` and `scanf()` to control the display of variables. Here are some common format specifiers in C: -

- *`%d` or `%i`: Integer*
- *`%f`: Floating-point number*
- *`%c`: Character*
- *`%s`: String*
- *`%x`: Hexadecimal integer*

7. Define Escape Sequence?

An escape sequence in programming is a series of characters that starts with a backslash (\) followed by one or more characters that together represent a special character. There are two escape sequence mostly used in the programming: -

- *\n: Newline*
- *\t: Horizontal tab*

8. Write down the purpose of Escape sequence?

The purpose of escape sequences is to allow the inclusion of special characters in strings that otherwise would be difficult to represent directly. They enable control over the output format, such as inserting new lines, tabs, or quotation marks within strings, and represent non-printable characters.

9. Write down the formation of Escape sequence?

Escape sequences begin with a backslash (\) followed by a specific character. Here are some common escape sequences:

- *\n: Newline*
- *\t: Horizontal tab*
- *\\: Backslash*
- *\": Double quote*
- *\': Single quote*
- *\r: Carriage return*
- *\b: Backspace*

10. Draw the structure of Format Specifier according to data types?

Data Type	Format Specifier
int	%d or %i
float	%f
char	%c

11. How many categories of Escape Sequence?

There are many categories of Escape sequence are as follow: -

- *\n: Newline*
- *\t: Horizontal tab*
- *\\: Backslash*
- *\": Double quote*
- *\': Single quote*
- *\r: Carriage return*
- *\b: Backspace*

12. Define operator and how many types of operator?

The name computer suggest that the computation is the most important aspect of computers. We need to perform computation data through programming. We have a lot of mathematical function t perform the calculations on the data. There are many types of operators are as follow: -

1. *Arithmetic Operator*
2. *Logical Operator*
3. *Relational Operator*
4. *Assignment Operator*

13. Define assignment operator?

Assignment operator is used to assign a value to a variable or assign a value to another variable. Equal sign (=) is used as assignment operator in C language.

For example: -

```
int sum=5;
int age=person;
```

14. Define statement terminator?

A statement terminator is identifier for the compiler which identify end of a line in C language semicolon (;) is used as a statement terminator.

15. Define delimiter?

A delimiter is identifier for the compiler which identify end of a program in C language delimiter { } is used as a statement terminator

16. Differentiate between assignment operator & equal operator?

Assignment Operator (=):

- *Used to assign a value to a variable.*
- *It takes the value on the right-hand side and stores it in the variable on the left-hand side.*
- *Example: `int a = 5;` assigns the value `5` to the variable `a`.*

Equality Operator (==):

- *Used to compare two values or expressions.*
- *It evaluates to `true` if the values are equal and `false` otherwise.*
- *Example: `if (a == 5)` checks if the variable `a` is equal to `5`.*

17. Define arithmetic operator?

An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. In programming, common arithmetic operators include:

- *+ : Addition*
- *- : Subtraction*
- ** : Multiplication*
- */ : Division*
- *% : Modulus (remainder after division)*

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
int a = 10, b = 3;
printf("Addition: %d\n", a + b);       // Output: 13
printf("Subtraction: %d\n", a - b);    // Output: 7
printf("Multiplication: %d\n", a * b); // Output: 30
printf("Division: %d\n", a / b);       // Output: 3
printf("Modulus: %d\n", a % b);        // Output: 1
return 0;
}
```

18. Differentiate between division & modulus operator?

Division Operator (/):

- *Divides the numerator by the denominator and returns the quotient.*
- *Example: `10 / 3` returns `3` in integer division.*

Modulus Operator (%):

- *Divides the numerator by the denominator and returns the remainder.*
- *Example: `10 % 3` returns `1`.*

19. Differentiate between multiplication and division operator?

Multiplication Operator (*):

- *The multiplication operator (*) multiplies two operands.*
- *Example: 5 * 4 returns 20.*

Division Operator (/):

- *The division operator (/) divides the numerator by the denominator.*
- *Example: 20 / 4 returns 5.*

20. Define ternary operator?

The ternary operator is a shorthand for the if-else statement. It takes three operands and is often used as a compact form of an if-else statement to assign a value to a variable based on a condition.

The syntax is: -

condition? expression_if_true: expression_if_false;

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
int a = 10;
int b = (a > 5) ? 1 : 0; // If a is greater than 5, b is set to 1, otherwise 0
printf("b = %d\n", b);   // Output: b = 1
return 0;
}
```

21. Draw the truth table for OR operator?

Expression	Result
False || False	False
False || True	True
True || False	True
True || True	True

22. Draw the truth table for AND operator?

Expression	Result
False && False	False
False && True	False
True && False	False
True && True	True

23. Differentiate between addition and subtraction operator?

Addition Operator (+):

- *The addition operator (+) adds two operands..*
- *Example: 5 + 3 results in 8.*
- *Usage: Used to sum values, like adding numbers or concatenating strings in some languages.*

Subtraction Operator (-):

- *The subtraction operator (-) subtracts the second operand from the first.*
- *Example: 5 - 3 results in 2.*
- *Usage: Used to find the difference between values*

24. Differentiate between logical & relational operator?

***Logical Operators**:*

- *Logical operators are used to perform logical operations, typically with Boolean values.*
- *Examples:*
  - *&& (Logical AND): Returns true if both operands are true.*
  - *|| (Logical OR): Returns true if at least one operand is true.*
  - *! (Logical NOT): Returns true if the operand is false.*
- *Usage: Commonly used in control flow statements like if, while, etc.*

***Relational Operators**:*

- *Relational operators are used to compare two values.*
- *Examples:*
  - *== (Equal to): Checks if two values are equal.*
  - *!= (Not equal to): Checks if two values are not equal.*
  - *> (Greater than): Checks if the left value is greater than the right value.*
  - *< (Less than): Checks if the left value is less than the right value.*
  - *>= (Greater than or equal to): Checks if the left value is greater than or equal to the right value.*
  - *<= (Less than or equal to): Checks if the left value is less than or equal to the right value.*
- *Usage: Commonly used in conditions to control program flow.*

25. Differentiate between logical and arithmetic operator?

***Logical Operators**:*

- *Logical operators are used to perform logical operations, typically with Boolean values..*
- *Examples: &&, ||, !*
- *Purpose: To evaluate and combine conditions.*
- *Example: if (a > 5 && b < 10)*

***Arithmetic Operators**:*

- *An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them.*
- *Examples: +, -, *, /, %*
- *Purpose: To perform mathematical calculations.*
- *Example: 5 + 3, 10 / 2*

26. What is the common mistake of while writing arithmetic statement in C level language?

Common mistakes include:

- ***Integer Division**: Assuming division of two integers will yield a float result.*
  Example: 5 / 2 results in 2, not 2.5.
- ***Operator Precedence**: Not using parentheses to control the order of operations.*
  *Example: a + b * c evaluates as a + (b * c).*
- ***Incorrect Use of Modulus Operator**: Using the modulus operator with non-integer types.*
- ***Division by Zero**: Forgetting to check if the denominator is zero before performing division, which leads to runtime errors.*

27. What is the use of arithmetic operator?

An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. Arithmetic operators are used to perform basic mathematical operations, such as:

- ***Addition (+)**: Summing values.*
- ***Subtraction (-)**: Finding the difference between values.*
- ***Multiplication (*)**: Calculating the product of values.*
- ***Division (/)**: Dividing one value by another.*
- ***Modulus (%)**: Finding the remainder after division.*

28. What is the purpose of tab sequence(\t)?

The tab sequence (\t) is used to insert a horizontal tab in a string. It is commonly used for formatting output to align text in columns.

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
   printf("Name\tAge\n");
   printf("Alice\t30\n");
   printf("Bob\t25\n");
   return 0;
}
```

29. What is the purpose of new line sequence(\n)?

The new line sequence (\n) is used to move the cursor to the beginning of the next line. It is used to separate lines in output.

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
   printf("Hello, World!\n");
   printf("This is a new line.\n");
   return 0;
}
```

30. What is the library function is used to getch ()?

The getch() function is provided by the conio.h library in C. It reads a single character from the keyboard but does not display it on the screen.

Coding Example: -

```
#include <conio.h>
using namespace std;
int main() {
   char ch;
   printf("Press any key to continue...\n");
   ch = getch();
   printf("You pressed: %c\n", ch);
   return 0;
}
```

31. Differentiate between AND & OR operator?

AND Operator (&&):

- *Evaluates to true if both operands are true.*
- *Example: (a > 5 && b < 10) is true if both a > 5 and b < 10 are true.*
- *Usage: Used when all conditions need to be true.*

OR Operator (||):

- *Evaluates to true if at least one of the operands is true.*
- *Example: (a > 5 || b < 10) is true if either a > 5 or b < 10 is true.*
- *Usage: Used when at least one condition needs to be true.*

32. Define NOT operator?

The NOT operator (!) is a logical operator that inverts the truth value of its operand. If the operand is true, it returns false; if the operand is false, it returns true.

33. Differentiate between binary operator & unary operator?

***Binary Operators**:*

Binary operators operate on two operands. They include arithmetic, logical, and relational operators. such as:

- ***Addition (+)**: Adds two operands.*
- ***Subtraction (-)**: Subtracts the second operand from the first.*
- ***Multiplication (*)**: Multiplies two operands.*
- ***Division (/)**: Divides the numerator by the denominator.*
- ***Usage:** Used for operations involving two values.*

***Unary Operators**:*

- *Unary operators operate on a single operand and perform various operations like negation, incrementing, and decrementing.*
- ***Examples: -** (negation), ! (logical NOT), ++ (increment), -- (decrement)*
- ***Example:** -a, ++a*
  ***Usage:** Used for operations involving a single value, such as changing its sign or incrementing/decrementing it.*

34. Define operator precedence?

Operator precedence determines the order in which operators are evaluated in expressions. Operators with higher precedence are evaluated before operators with lower precedence. Parentheses can be used to override the default precedence.

Coding example: -

```
result =18/2*3+7%3+(5*4);
result=18/2*3+7%3+20;
result=9*3+7%3+20;
result=27+1+20;
result=28+20;
result=48;
```

LONG QUESTIONS

1. Write down escape sequence?

An escape sequence in programming is a series of characters that starts with a backslash (\) followed by one or more characters that together represent a special character. There are two escape sequence mostly used in the programming: -

- *\n: Newline*
- *\t: Horizontal tab*

Purpose: -

The purpose of escape sequences is to allow the inclusion of special characters in strings that otherwise would be difficult to represent directly. They enable control over the output format, such as inserting new lines, tabs, or quotation marks within strings, and represent non-printable characters

Formation: -

Escape sequences begin with a backslash (\) followed by a specific character. Here are some common escape sequences:

- **
- *n: Newline*
- *\t: Horizontal tab*
- *\\: Backslash*
- *\": Double quote*
- *\': Single quote*
- *\r: Carriage return*
- *\b: Backslash*

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
   printf("Name\tAge\n");
   printf("Alice\t30\n");
   printf("Bob\t25\n");
   return 0;
}
```

2. *Write a note on format specifier?*

A format specifier in programming languages like C is a placeholder within a string that specifies how an argument should be formatted. It is used in functions like `printf()` and `scanf()` to control the display of variables. Here are some common format specifiers in C: -

- *`%d` or `%i`: Integer*
- *`%f`: Floating-point number*
- *`%c`: Character*
- *`%s`: String*
- *`%x`: Hexadecimal integer*

Structure of Format Specifier according to data types?

Data Type	***Format Specifier***
int	*%d or %i*
float	*%f*
char	*%c*

Coding Example: -

```
#include <conio.h>
using namespace std;
int main() {
   char ch;
   printf("Press any key to continue...\n");
   ch = getch();
   printf("You pressed: %c\n", ch);
   return 0;
}
```

3. *Write a note on Operator?*

In programming and mathematics, an operator is a symbol or a character that performs a specific operation or action on one or more operands (values or variables).

Operators are used to perform various operations, such as calculations, comparisons, and assignments, in programming languages and mathematical expressions.

Types of operators include:

1. ***Arithmetic operators:***

   An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. Arithmetic operators are used to perform basic mathematical operations, such as:

   *+ (addition), - (subtraction), * (multiplication), / (division), %(modulus)*

2. ***Relational/Comparison operators:***

Relational operators are used to compare two values.

***Examples**:*

a. *== (Equal to): Checks if two values are equal.*

b. *!= (Not equal to): Checks if two values are not equal.*
c. *> (Greater than): Checks if the left value is greater than the right value.*
d. *< (Less than): Checks if the left value is less than the right value.*
e. *>= (Greater than or equal to): Checks if the left value is greater than or equal to the right value.*
f. *<= (Less than or equal to): Checks if the left value is less than or equal to the right value.*

***Usage**: Commonly used in conditions to control program flow.*

3. ***Logical operators**:*
   Logical operators are used to perform logical operations, typically with Boolean values..
   ***Purpose**: To evaluate and combine conditions.*
   ***Example**: if (a > 5 && b < 10)*
   Categories of operators:
   && (and), || (or), ! (not)
4. ***Assignment operators:***
   Used to assign a value to a variable.
   It takes the value on the right-hand side and stores it in the variable on the left-hand side.
   Example: int a = 5; assigns the value 5 to the variable a.
   Categories of operators:
   = (assignment), += (addition assignment), -= (subtraction assignment)
5. ***Bitwise operators**: & (bitwise and), | (bitwise or), ^ (bitwise xor)*
6. ***Binary Operator:***

Binary operators operate on two operands. They include arithmetic, logical, and relational operators. such as:

- ***Addition (+)**: Adds two operands.*
- ***Subtraction (-)**: Subtracts the second operand from the first.*
- ***Multiplication (*)**: Multiplies two operands.*
- ***Division (/)**: Divides the numerator by the denominator.*
- ***Usage:** Used for operations involving two values.*

7. ***Unary operator***
   Unary operators operate on a single operand and perform various operations like negation, incrementing, and decrementing.
   ***Examples:** - (negation), ! (logical NOT), ++ (increment), -- (decrement)*
   ***Example:** -a, ++a*
   ***Usage:** Used for operations involving a single value, such as changing its sign or incrementing/decrementing it.*
8. ***Equality operator***
   - *Used to compare two values or expressions.*
   - *It evaluates to true if the values are equal and false otherwise.*
   - *Example: if (a == 5) checks if the variable a is equal to 5.*
9. ***Ternary Operator***
   The ternary operator is a shorthand for the if-else statement. It takes three operands and is often used as a compact form of an if-else statement to assign a value to a variable based on a condition.
   The syntax is: -
   condition? expression_if_true: expression_if_false;

4. ***Write a note on arithmetic operator?***

An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them. Arithmetic operators are used to perform basic mathematical operations, such as:

TYPES:

1. ***+ (addition)***

   The addition operator (+) adds two operands.

   ***Example**: 5 + 3 results in 8.*

   ***Usage**: Used to sum values, like adding numbers or concatenating strings in some languages*

2. ***-(subtraction)***

   The subtraction operator (-) adds two operands.

   ***Example**: 5 - 3 results in 2.*

3. ****(multiplication)***

   The Multiplication operator (*) adds two operands.

   ***Example**: 5 * 3 results in 15.*

4. ***/ (division)***

   Divides the numerator by the denominator and returns the quotient.

   Example: `10 / 3` returns `3` in integer division

   %(modulus)

- *Divides the numerator by the denominator and returns the remainder.*
- *Example: `10 % 3` returns `1`.*

Coding Example: -

```
#include <stdio.h>
using namespace std;
int main() {
        int a = 10, b = 3;
        printf("Addition: %d\n", a + b);       // Output: 13
        printf("Subtraction: %d\n", a - b);    // Output: 7
        printf("Multiplication: %d\n", a * b); // Output: 30
        printf("Division: %d\n", a / b);       // Output: 3
        printf("Modulus: %d\n", a % b);        // Output: 1
        return 0;
}
```

5. ***Write a note on logical operator?***

   Logical operators are used to perform logical operations, typically with Boolean values..

   ***Purpose**: To evaluate and combine conditions.*

   ***Example**: if (a > 5 && b < 10)*

Categories of operators:

1. *&& (and),*

   Logical operators are used to perform logical operations, typically with Boolean values..

   ***Purpose**: To evaluate and combine conditions.*

   ***Example**: if (a > 5 && b < 10)*

Expression	Result
False && False	False
False && True	False
True && False	False
True && True	True

2. *|| (or),*
   Evaluates to true if at least one of the operands is true.
   ***Example**: (a > 5 || b < 10) is true if either a > 5 or b < 10 is true.*
   ***Usage**: Used when at least one condition needs to be true.*

Expression	Result
False || False	False
False || True	True
True || False	True
True || True	True

3. *! (not)*
   The NOT operator (!) is a logical operator that inverts the truth value of its operand. If the operand is true, it returns false; if the operand is false, it returns true.

A	***~A***
F	*T*
T	*F*

6. What is the common mistake of while writing arithmetic statement in C level language?

Common mistakes include:

- ***Integer Division**: Assuming division of two integers will yield a float result.*
  Example: 5 / 2 results in 2, not 2.5.
- ***Operator Precedence**: Not using parentheses to control the order of operations.*
  *Example: a + b * c evaluates as a + (b * c).*
- ***Incorrect Use of Modulus Operator**: Using the modulus operator with non-integer types.*
- ***Division by Zero**: Forgetting to check if the denominator is zero before performing division, which leads to runtime errors.*

7. Write a note on relational operator?

Relational operators are used to compare two values.

***Examples**:*

a. *== (Equal to): Checks if two values are equal.*
b. *!= (Not equal to): Checks if two values are not equal.*
c. *> (Greater than): Checks if the left value is greater than the right value.*
d. *< (Less than): Checks if the left value is less than the right value.*
e. *>= (Greater than or equal to): Checks if the left value is greater than or equal to the right value.*
f. *<= (Less than or equal to): Checks if the left value is less than or equal to the right value.*

***Usage**: Commonly used in conditions to control program flow.*

LONG CODING QUESTIONS

1. ***Write a program that takes original piece of a shirt and discount percentage from user. Program should display the original price of shirt, discount on piece and after discount.***

   Code: -

```
#include <stdio.h>
using namespace std;
int main() {
   float originalPrice, discountPercent, discount, finalPrice;
   printf("Enter the original price of the shirt: ");
   scanf("%f", &originalPrice);
   printf("Enter the discount percentage: ");
   scanf("%f", &discountPercent);
   discount = (discountPercent / 100) * originalPrice;
   finalPrice = originalPrice - discount;
   printf("Original Price: %.2f\n", originalPrice);
   printf("Discount: %.2f\n", discount);
   printf("Price after Discount: %.2f\n", finalPrice);
   return 0;
}
```

2. ***Write a program that takes the input of number of balls in Jar A and number of balls in Jar B. The program calculates and display the number of balls.***

   Code: -

```
#include <stdio.h>
using namespace std;
int main() {
   int jarA, jarB, totalBalls;
   printf("Enter the number of balls in Jar A: ");
   scanf("%d", &jarA);
   printf("Enter the number of balls in Jar B: ");
   scanf("%d", &jarB);
   totalBalls = jarA + jarB;
   printf("Total number of balls: %d\n", totalBalls);
   return 0;
}
```

3. ***Write a program that takes as input the length of one side of a square and calculate the area of a square.***

   Code: -

```
#include <stdio.h>
using namespace std;
int main() {
   float side, area;
   printf("Enter the length of one side of the square: ");
   scanf("%f", &side);
   area = side * side;
   printf("Area of the square: %.2f\n", area);
   return 0;
}
```

4. ***Write a program that takes roll number, percentage of marks and grades, from user input. Program should display the formatted output the following: -***

   Roll no: input value
   Percentage: input value %
   Grade: input value

Code: -

```
#include <stdio.h>
using namespace std;
int main() {
   int rollNo;
   float percentage;
   char grade;
   printf("Enter roll number: ");
   scanf("%d", &rollNo);
   printf("Enter percentage: ");
   scanf("%f", &percentage);
   printf("Enter grade: ");
   scanf(" %c", &grade);
   printf("Roll no: %d\n", rollNo);
   printf("Percentage: %.2f%%\n", percentage);
   printf("Grade: %c\n", grade);
   return 0;
}
```

5. ***Write a program that takes radius of a circle as an input. The program should be calculating and display the area of a circle.***

Code: -

```
#include <stdio.h>
#define PI 3.14159
using namespace std;
int main() {
   float radius, area;
   printf("Enter the radius of the circle: ");
   scanf("%f", &radius);
   area = 2*PI * radius * radius;
   printf("Area of the circle: %.2f\n", area);
   return 0;
}
```

6. ***Write a program that takes monthly income and monthly expense of the user like electricity bill, gas bill and food expense. Program should be calculating the following: -***

   Total monthly expense: yearly saving
   Total yearly expense: average saving per month
   Monthly saving: average expense per month

Code: -

```
#include <stdio.h>
using namespace std;
int main() {
   float income, electricity, gas, food, totalMonthlyExpense, monthlySaving, totalYearlyExpense, yearlySaving;
   printf("Enter monthly income: ");
   scanf("%f", &income);
   printf("Enter monthly electricity bill: ");
   scanf("%f", &electricity);
   printf("Enter monthly gas bill: ");
   scanf("%f", &gas);
```

```
    printf("Enter monthly food expense: ");
    scanf("%f", &food);
    totalMonthlyExpense = electricity + gas + food;
    monthlySaving = income - totalMonthlyExpense;
    totalYearlyExpense = totalMonthlyExpense * 12;
    yearlySaving = monthlySaving * 12;
    printf("Total monthly expense: %.2f\n", totalMonthlyExpense);
    printf("Monthly saving: %.2f\n", monthlySaving);
    printf("Total yearly expense: %.2f\n", totalYearlyExpense);
    printf("Yearly saving: %.2f\n", yearlySaving);
    return 0;
}
```

7. ***Write a program that should take working hour and overtime hour of employee as input. The program should calculate and display the total salary: -***

   Basic Salary = pay rate per hour * working hour of employee
   Overtime Salary = overtime pay rate * overtime working hour
   Total Salary = Basic Salary + Overtime Salary

Code: -

```
#include <stdio.h>
using namespace std;
int main() {
    float payRate, overtimeRate, workingHours, overtimeHours, basicSalary, overtimeSalary, totalSalary;
    printf("Enter pay rate per hour: ");
    scanf("%f", &payRate);
    printf("Enter working hours: ");
    scanf("%f", &workingHours);
    printf("Enter overtime pay rate: ");
    scanf("%f", &overtimeRate);
    printf("Enter overtime hours: ");
    scanf("%f", &overtimeHours);
    basicSalary = payRate * workingHours;
    overtimeSalary = overtimeRate * overtimeHours;
    totalSalary = basicSalary + overtimeSalary;
    printf("Total Salary: %.2f\n", totalSalary);
    return 0;
}
```

8. ***Write a program that take Celsius input as a temperature and convert into the Fahrenheit and show the output. According to the formula***

   F=[{(9*C)/5} + 32]

Code: -

```
#include <stdio.h>
using namespace std;
int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = ((9 * celsius) / 5) + 32;
    printf("Temperature in Fahrenheit: %.2f\n", fahrenheit);
    return 0;
}
```

ERRORS AND OUTPUT OF A PROGRAMING

Q.5 Write down output of the following code segments.

a)

```
#include<stdio.h>
  void main()
  {
        int x = 2, y = 3, z = 6;
        int ans1, ans2, ans3;
        ans1 = x / z * y;
        ans2 = y + z / y * 2;
        ans3 = z / x + x * y;
        printf("%d %d %d", ans1, ans2, ans3);
  }
```

Output:
0, 7, 9

b)

```
#include<stdio.h>
        void main()
        {
              printf("nn \n\n nnn \nn \n t  \t");
              printf("nn /n/n nn/n\n");
        }
```

Output:
nn

nnn
n
t nn /n/n nn/n

---

c)

```
#include<stdio.h>
 void main()
        {
              int a = 4, b;
              float c = 2.3;
              b = c * a;
              printf("%d", b);
        }
```

Output:
9

d)

```
#include<stdio.h>
void main()
        {
              int a = 4 * 3 / ( 5 + 1 ) + 7 % 4;
              printf("%d", a);
        }
```

Output:
5

e)

```
#include<stdio.h>
void main()
        {
              printf("%d", (((5 > 3) && (4 > 6)) || (7 > 3)));
        }
```

Output:
1

Q.6 Identify errors in the following code segments.

a)

```
#include<stdio.h>
void main()
        {
                int a, b = 13;
                b = a % 2;
                printf("Value of b is: %d, b);
        }
```

Error:
Value of variable a is not initialized.
Inverted comma is missing after %d

b)

```
#include<stdio.h>
void main()
        {
                int a, b, c,
                printf("Enter First Number:");
                scanf("%d", &a);
                printf("Enter second number:");
                scanf("%d", &b);
                a + b = c;
        }
```

Error:
Semicolon is missing after int variable. Invalid syntax of a + b = c;
The correct syntax is c=a+b;

c)

```
#include<stdio.h>
void main()
        {
                int num;
                printf(Enter Number:");
                scanf(%d, &num);
        };
```

Error:
Double quotes missing in printf statement and } has extra ;

---

d)

```
#include<stdio.h>
void main()
{
        float f;
        printf["Enter value:"];
        scanf("%c", &f);
}
```

Error:
Square bracket is used instead of parentheses. In printf statement float variable is used with %f in scanf instead of %c.

MCQ'S (MULTIPLE CHOICE QUESTIONS)

MCQs

Sr. No	MCQs	A	B	C	D
1	A .....is a device that takes data as input, process that data and generates the output.	Computer	Printer	Mobile	All of these
2	Each programming language has its.....or ..... functions for I/O operations.	Keywords	standard library	Language	both a and b
3	C language offers printf function to display the:	Program	Input	Output	All of these
4	.....function is used to get input from user.	getch	scanf	printf	print
5	The name of printf comes from ..... that is used to print the formatted output on screen.	"escape sequence"	"unprinted formatted"	"printed sequence"	"print formatted"
6	The format specifier % f show the data type	int	float	char	All of these
7	The format specifier % c show the data type	int	float	char	Grouped
8	When we use %f to display a float value, it display ..... digits after the decimal point.	5	6	7	8
9	scanf is a built-in function in C language that takes ..... from user into the variables	Input	Output	Instruction	Guideline
10	The expected input data type in scanf function can be specified with the help of _____.	scanf	Escape sequence	format specifier	logical operator
11	The main parts of scanf function _____	2	3	4	5
12	The first part of scanf function is_____	inside the single quotes	inside the double quotes	inside the curly bracket	inside the triple quotes
13	Without & sign in scanf function, the program gets ..... but does not behave as expected.	Instruction	Complied	Executed	Input
14	The common mistake in scanf function _____ sign.	&	%	*	$
15	_____function is used to read a character from user.	printf	getch()	scanf	scan

16	To use getch() function, there is a need to include the library _____in the header section of program.	stdio.h	void main()	conio.h	include.h
17	When we read character through scanf, it requires us to .....for further execution.	Press enter	Press esc	Press shift	Press ctrl
18	A _____is identifier for compiler which identifies end of a line.	Format specifier	Escape sequence	Statement terminator	Logical operator
19	In C language _____is used as statement terminator.	!	;	%	,
20	_____are used in printf function inside the "and".	Terminator	Statement terminator	Escape terminator	Format specifier
21	printf("My name is \"Ali\""); In the above statement \" is an_____	Escape sequence	Format specifier	Format	Operator
22	_____consist of two characters.	Format specifier	Escape sequence	Statement terminator	All of these
23	What is the purpose of escape sequence \\?	Display Back slash	Generates an alert sound	Removes previous char	Display Single Quote
24	What is the purpose of escape sequence \a?	Display Single Quote	Removes previous char	Generates an alert sound	Display Back slash
25	After escape character _____ specifies movement of the cursor to start of the next line.	\t	\n	\b	\a
26	Which escape sequence is used to print the output on multiple lines?	\a	\b	\c	\n
27	Escape sequence_____ specifier the I/O function of moving to the next tab stop horizontally.	\m	\n	\t	\\
28	The list of some basic operator types is _____.	Assignment operator	Arithmetic operator	Relational operator	All of these
29	_____ is used to assign a value to a variable, or assign a value of variable to another variable.	Arithmetic operator	Assignment operator	Logical operator	Relational operator

30	____ is used as assignment operator in C.	=	$	!	%
31	____ are used to perform arithmetic operations on data	Arithmetic operator	Assignment operator	Logical operator	Relational operator
32	What is the name of / operator?	Multiplication operator	Subtraction operator	Division operator	Addition operator
33	___ divides the value of left operand by the value of right operand.	Division operator	Multiplication operator	Addition operator	Modulus operator
34	If both the operands are of type int, then result of division is also type of ____.	char	var	int	All of these
35	_____is a binary operator which performs the product of two numbers.	Logical operator	Arithmetic operator	Multiplication operator	Division operator
36	_____ calculates the sum of two operands.	Arithmetic operator	Multiplication operator	Division operator	Logical operator
37	The statement a--; or --a, used to decrease the value of a by _____.	1	1.5	1.8	-1
38	___ subtracts right operand from the left operand.	Arithmetic operator	Multiplication operator	Division operator	Subtraction Operator
39	What is the name of % operator _____.	Multiplication operator	Modulus operator	Division operator	Subtraction Operator
40	_____ performs divisions of left operand by the right operand and returns the remainder value after division.	Modulus operator	Division operator	Subtraction Operator	Arithmetic operator
41	_____compare two values to determine the relationship between values.	Arithmetic operator	Relational operator	Logical operator	Binary operator
42	C language allows us to perform relational operators on _____and _____data type.	int	Numeric	char	Both b and c
43	A true value of relational operator is represented by_____.	0	1	2	3
44	A false value relational operator is represented by_____.	0	1	3	2

45	In C language,_____ operator is used to check for equality of two expressions.	Assignment operator	Arithmetic operator	Logical operator	Relational operator
46	____operator assigns the result of expression on right side to the variable on left side.	==	=	*	%
47	_____ perform operations on Boolean expression and produce a Boolean expression as a result	Logical operator	Assignment operator	Arithmetic operator	Relational operator
48	Which one from the following is not a logical operator?	AND	OR	NAND	NOT
49	_____operator takes two Boolean expressions as operands and produces the result true if both of its operands are true.	AND	OR	NAND	NOT
50	_____accepts Boolean expression and returns true if at least one of the operands is true.	AND	OR	NAND	NOT
51	_____operator negates or reverses the value of Boolean expression. It makes it true, if it is false and false if it is true.	NOT	AND	OR	NAND
52	_____require two operands to perform the operation.	Relational operator	Unary operator	Binary Operator	Logical Operator
53	_____are applied over one operand only.	Unary operator	Binary Operator	Logical Operator	Relational operator
54	Which operator has high precedence?	&	^	( )	|
55	_____operator are applied on three operand.	Ternary	Unary	Binary	Logical
56	____function is used to clear the output screen of C language editor.	getch();	scanf	printf	clrscr();

Keys:

1	2	3	4	5	6	7	8	9	10
A	D	C	B	D	B	C	B	A	C
11	12	13	14	15	16	17	18	19	20
A	B	C	A	B	C	A	C	B	C
21	22	23	24	25	26	27	28	29	30
A	B	A	C	B	D	C	D	B	A
31	32	33	34	35	36	37	38	39	40
A	C	A	C	C	A	A	D	B	A
41	42	43	44	45	46	47	48	49	50
B	D	B	A	A	B	A	C	A	B
51	52	53	54	55					
A	C	A	C	A					