NAME = Harshit Kumar
BATCH = AU-1
ROLLNO = 31

Assignment

Q1. Write a C program for calculating the price of a product after adding the sales tax to its original price. Where rate of tax and price is inputted by user.

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

int main() {
    // Declare variables to store original price and tax rate
    float originalPrice, taxRate, finalPrice;

    // Prompt the user to enter the original price
    printf("Enter the original price of the product: $");
    scanf("%f", &originalPrice);

    // Prompt the user to enter the tax rate
    printf("Enter the tax rate (in percentage): ");
    scanf("%f", &taxRate);

    // Calculate the final price by adding tax
    finalPrice = originalPrice + (originalPrice * (taxRate / 100));

    // Display the final price
    printf("The final price after adding %.2f%% tax is: $%.2f\n", taxRate, finalPrice);

    return 0;
}
```

Q2. Write a C program to calculate the weekly wages of an employee. The pay depends on wages per hour and number of hours worked. Moreover, if the employee has worked for more than 30 hours, then he or she gets twice the wages per hour, for every extra hour that he or she has worked.

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

int main() {
    float hourlyWage, weeklyWages;
    int hoursWorked;

    // Prompt the user to enter the hourly wage
    printf("Enter the hourly wage: $");
    scanf("%f", &hourlyWage);

    // Prompt the user to enter the number of hours worked
    printf("Enter the number of hours worked in a week: ");
    scanf("%d", &hoursWorked);

    // Calculate weekly wages
    if (hoursWorked <= 30) {
        weeklyWages = hourlyWage * hoursWorked;
    } else {
```

```
        // Calculate regular wages for the first 30 hours
        weeklyWages = hourlyWage * 30;

        // Calculate extra wages for hours worked beyond 30 (double the hourly wage)
        weeklyWages += 2 * hourlyWage * (hoursWorked - 30);
    }

    // Display the weekly wages
    printf("Weekly wages: $%.2f\n", weeklyWages);

    return 0;
}
```

Q.3 Mr. X goes to market for buying some fruits and vegetables. He is having a currency of Rs 500 with him for marketing. From a shop, he purchases 2.0 kg Apple priced Rs. 50.0 per kg, 1.5 kg Mango priced Rs.35.0 per kg, 2.5 kg Potato priced Rs.10.0 per kg, and 1.0 kg Tomato priced Rs.15 per kg. He gives the currency of Rs. 500 to the shopkeeper. Find out the amount shopkeeper will return to X by writing a C program.

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

int main() {
    // Declare variables for the quantities and prices of items
    float applePricePerKg = 50.0;
    float mangoPricePerKg = 35.0;
    float potatoPricePerKg = 10.0;
    float tomatoPricePerKg = 15.0;

    float appleWeight = 2.0;
    float mangoWeight = 1.5;
    float potatoWeight = 2.5;
    float tomatoWeight = 1.0;

    // Calculate the total cost of each item
    float totalCostApple = appleWeight * applePricePerKg;
    float totalCostMango = mangoWeight * mangoPricePerKg;
    float totalCostPotato = potatoWeight * potatoPricePerKg;
    float totalCostTomato = tomatoWeight * tomatoPricePerKg;

    // Calculate the total cost of all items
    float totalCost = totalCostApple + totalCostMango + totalCostPotato + totalCostTomato;

    // Declare variables for the currency given and change to be returned
    float currencyGiven = 500.0;
    float change = currencyGiven - totalCost;

    // Check if Mr. X has enough money to cover the total cost
    if (change >= 0) {
        printf("Mr. X will get Rs. %.2f in return.\n", change);
    } else {
        printf("Mr. X does not have enough money to cover the total cost.\n");
    }

    return 0;
}
```

Q4.Write a C program to print your name, date of birth and mobile number in 3 different lines.

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

int main() {
    // Declare variables to store your information
    char name[] = "Your Name";
    char dob[] = "Your Date of Birth";
    char mobileNumber[] = "Your Mobile Number";

    // Print your name
    printf("Name: %s\n", name);

    // Print your date of birth
    printf("Date of Birth: %s\n", dob);

    // Print your mobile number
    printf("Mobile Number: %s\n", mobileNumber);

    return 0;
}
```

Q5.Write a program to read an integer, a character and a float value from keyboard and display the same in different lines on the screen.

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

int main() {
    int integerNumber;
    char character;
    float floatNumber;

    // Prompt the user to enter an integer
    printf("Enter an integer: ");
    scanf("%d", &integerNumber);

    // Prompt the user to enter a character
    printf("Enter a character: ");
    scanf(" %c", &character); // Note the space before %c to consume any leading whitespace.

    // Prompt the user to enter a floating-point number
    printf("Enter a float: ");
    scanf("%f", &floatNumber);

    // Display the entered values on separate lines
    printf("Integer: %d\n", integerNumber);
    printf("Character: %c\n", character);
    printf("Float: %.2f\n", floatNumber);

    return 0;
}
```

Q6.Write a program to print the following line (Assume the total value is contained in a variable named cost)
The sales total is : $ 172.53

Ans. #include <stdio.h>

```
int main() {
    // Assume the total value is contained in a variable named cost
    float cost = 172.53;

    // Print the line with the sales total
    printf("The sales total is: $ %.2f\n", cost);

    return 0;
}
```

Q7.Raju got 6 and half apples from each of Raghu, Sheenu and Akash. He wants to know how many apples he has in total without adding them. Write a program which could help Raju in doing this.

Ans. #include <stdio.h>

```
int main() {
    // Number of apples Raju got from each person
    float applesFromRaghu = 6.5;
    float applesFromSheenu = 6.5;
    float applesFromAkash = 6.5;

    // Calculate the total number of apples without adding manually
    float totalApples = applesFromRaghu + applesFromSheenu + applesFromAkash;

    // Display the total number of apples
    printf("Raju has %.1f apples in total.\n", totalApples);

    return 0;
}
```

Q8.Write a program that prints the floating point value in exponential format correct to two decimal places.

Ans. #include <stdio.h>

```
int main() {
    // Declare a floating-point value
    double floatValue = 12345.6789;

    // Print the floating-point value in exponential format with two decimal places
    printf("Floating-point value in exponential format: %.2e\n", floatValue);

    return 0;
}
```

Q9.Write a program to input and print your mobile number (i.e. of 10 digits).

Ans. #include <stdio.h>

```
int main() {
    long long int mobileNumber; // Assuming a 10-digit mobile number as a long long int
```

```
    // Prompt the user to enter their 10-digit mobile number
    printf("Enter your 10-digit mobile number: ");
    scanf("%lld", &mobileNumber);

    // Check if the entered number has exactly 10 digits
    if (mobileNumber >= 1000000000LL && mobileNumber <= 9999999999LL) {
        // Print the entered mobile number
        printf("You entered the mobile number: %lld\n", mobileNumber);
    } else {
        printf("Invalid input. Please enter a 10-digit mobile number.\n");
    }

    return 0;
}
```

Q10.The population of a city is 30000. It increases by 20 % during first year and 30% during the second year. Write a program to find the population after two years?

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

int main() {
    // Initialize the initial population
    int initialPopulation = 30000;

    // Calculate the population after the first year (increased by 20%)
    int populationAfterFirstYear = initialPopulation + (0.20 * initialPopulation);

    // Calculate the population after the second year (increased by 30%)
    int populationAfterSecondYear = populationAfterFirstYear + (0.30 * populationAfterFirstYear);

    // Print the population after two years
    printf("Population after two years: %d\n", populationAfterSecondYear);

    return 0;
}
```

Q11. Write a program to find the ASCII value of a character.

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

int main() {
    char character;

    // Prompt the user to enter a character
    printf("Enter a character: ");
    scanf("%c", &character);

    // Find and print the ASCII value of the entered character
    printf("ASCII value of '%c' is %d\n", character, (int)character);

    return 0;
}
```

Q12. Write a program to calculate salary of an employee, given his basic pay (entered by user), HRA=15% of the basic pay and TA=20% of the basic pay.

Ans. #include <stdio.h>

```
int main() {
    float basicPay, HRA, TA, salary;

    // Prompt the user to enter the basic pay
    printf("Enter the basic pay: $");
    scanf("%f", &basicPay);

    // Calculate HRA (15% of basic pay)
    HRA = 0.15 * basicPay;

    // Calculate TA (20% of basic pay)
    TA = 0.20 * basicPay;

    // Calculate the total salary (basic pay + HRA + TA)
    salary = basicPay + HRA + TA;

    // Display the calculated salary
    printf("Salary of the employee is: $%.2f\n", salary);

    return 0;
}
```

Q13. Write a program to find the slope of a line and angle of inclination that passes through two points P and Q with coordinates (xp, yp) and (xq, yq) respectively.

Ans. #include <stdio.h>
#include <math.h>

```
int main() {
    // Declare variables to store the coordinates of points P and Q
    double xp, yp, xq, yq;

    // Prompt the user to enter the coordinates of point P
    printf("Enter the coordinates of point P (xp, yp): ");
    scanf("%lf %lf", &xp, &yp);

    // Prompt the user to enter the coordinates of point Q
    printf("Enter the coordinates of point Q (xq, yq): ");
    scanf("%lf %lf", &xq, &yq);

    // Calculate the slope of the line
    double slope = (yq - yp) / (xq - xp);

    // Calculate the angle of inclination in degrees
    double angle = atan(slope) * 180.0 / M_PI;

    // Display the slope and angle of inclination
    printf("Slope of the line: %.2lf\n", slope);
    printf("Angle of inclination: %.2lf degrees\n", angle);

    return 0;
}
```

Q14. The SPI (Semester Performance Index) is a weighted average of the grade points earned by a student in all the courses he registered for in a semester. If the grade points associated with the letter grades awarded to a student are g1, g2, g3,鈥﹀€�.gk etc. and the corresponding credits are c1, c2, c3,.鈥..ck, the SPI is given by:

Where, k is the number of courses for which the candidate remains registered for during the semester/ trimester. Write a program in C to calculate SPI for k =5.

Ans. #include <stdio.h>

```
int main() {
    // Initialize the grade points and credits for 5 courses
    float g1 = 3.5, g2 = 4.0, g3 = 3.0, g4 = 3.7, g5 = 4.0;
    int c1 = 3, c2 = 4, c3 = 3, c4 = 2, c5 = 3;

    // Calculate the SPI
    float spi = (g1 * c1 + g2 * c2 + g3 * c3 + g4 * c4 + g5 * c5) / (c1 + c2 + c3 + c4 + c5);

    // Print the SPI
    printf("SPI for 5 courses = %.2f\n", spi);

    return 0;
}
```

Q 15. Write a program to calculate the frequency (f) of a given wave with wavelength (位) and speed (c), where c=位*f.

Ans. #include <stdio.h>

```
int main() {
    // Declare variables to store wavelength and speed
    double wavelength, speed, frequency;

    // Prompt the user to enter the wavelength and speed
    printf("Enter the wavelength (in meters): ");
    scanf("%lf", &wavelength);

    printf("Enter the speed of the wave (in meters per second): ");
    scanf("%lf", &speed);

    // Calculate the frequency using the formula c = 位 * f (f = c / 位)
    frequency = speed / wavelength;

    // Display the calculated frequency
    printf("The frequency of the wave is %.2lf Hz.\n", frequency);

    return 0;
}
```

Q 16. A car travelling at 30 m/s accelerates steadily at 5 m/s2 for a distance of 70 m. What is the final velocity of the car? [Hint: v2 = u2 + 2as]

Ans. #include <stdio.h>
#include <math.h>

```
int main() {
	double initialVelocity = 30.0; // m/s
	double acceleration = 5.0; // m/s^2
	double distance = 70.0; // m
	double finalVelocity;

	finalVelocity = sqrt(pow(initialVelocity, 2) + 2 * acceleration * distance);

	printf("The final velocity of the car is %.2lf m/s\n", finalVelocity);

	return 0;
}
```

When you run this program, it will calculate and display the final velocity of the car, which is approximately 38.14 m/s (rounded to two decimal places).

Q 17.A horse accelerates steadily from rest at 4 m/s2 for 3s. (a) What is its final velocity? (b) How far has it travelled? [Hint: (a) v = u + at (b) s = ut + 陆 at2]

Ans.

```
#include <stdio.h>

int main() {
	// Given values
	double initial_velocity = 0.0;	// Initial velocity in m/s
	double acceleration = 4.0;		// Acceleration in m/s^2
	double time = 3.0;			// Time in seconds

	// (a) Calculate final velocity using v = u + at
	double final_velocity = initial_velocity + acceleration * time;

	// (b) Calculate distance traveled using s = ut + 1/2 * at^2
	double distance_traveled = initial_velocity * time + 0.5 * acceleration * time * time;

	// Output the results
	printf("(a) Final velocity: %.2lf m/s\n", final_velocity);
	printf("(b) Distance traveled: %.2lf meters\n", distance_traveled);

	return 0;
}
```

Q 18. Write a program to find the sum of your four last digit of your university roll number .

Ans.

```
#include <stdio.h>

int main() {
	// Replace '12345678' with your actual university roll number
	int roll_number = 12345678;

	// Extract the last four digits
	int last_four_digits = roll_number % 10000;

	// Calculate the sum of the last four digits
	int sum = 0;
	while (last_four_digits > 0) {
		sum += last_four_digits % 10;
		last_four_digits /= 10;
	}
```

```
    printf("Sum of the last four digits of the roll number: %d\n", sum);

    return 0;
}
```
In this program, replace 12345678 with your actual university roll number. When you run the program, it will calculate and print the sum of the last four digits of your university roll number.

Q19. Write a program to initialize your height and weight in cm. and kgs respectively demonstrating compile time initialization and convert them in feets and pounds respectively. Note :- 1 cm = 0.393701inch , 1 Kg = 2.20462

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

int main() {
    // Compile-time initialization of height in centimeters and weight in kilograms
    double height_cm = 175.0;    // Replace with your actual height in cm
    double weight_kg = 70.0;     // Replace with your actual weight in kg

    // Constants for conversion
    const double CM_TO_INCH = 0.393701;
    const double KG_TO_POUND = 2.20462;

    // Conversion
    double height_inch = height_cm * CM_TO_INCH;
    double weight_pound = weight_kg * KG_TO_POUND;

    // Output the results
    printf("Height in feet: %.2lf ft\n", height_inch / 12.0);
    printf("Weight in pounds: %.2lf lbs\n", weight_pound);

    return 0;
}
```
In this program, replace 175.0 with your actual height in centimeters and 70.0 with your actual weight in kilograms. When you run the program, it will calculate and print your height in feet and your weight in pounds based on the provided conversion factors.

Q 20 . Code the variable declarations for each of following:
a) A character variable named option.
b) An integer variable sum initialized to 0
c) A floating point variable, product, initialized to 1

Ans.
a) A character variable named option:
```
char option;
```

b) An integer variable sum initialized to 0
```
int sum = 0;
```

c) A floating-point variable product initialized to 1:
```
float product = 1.0;
```

Q21. Write a program that reads nine integers. Display these numbers by printing three numbers in a line separated by commas.

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

```
int main() {
    int numbers[9];    // Array to store the nine integers

    // Input nine integers
    printf("Enter nine integers, one at a time:\n");
    for (int i = 0; i < 9; i++) {
        scanf("%d", &numbers[i]);
    }

    // Display the numbers in groups of three
    printf("Numbers in groups of three:\n");
    for (int i = 0; i < 9; i++) {
        printf("%d", numbers[i]);
        if ((i + 1) % 3 == 0) {
            printf("\n");    // Start a new line after every third number
        } else {
            printf(", ");    // Add a comma and space between numbers
        }
    }

    return 0;
}
```

Q22. What are header files and what are its uses in C programming?

Ans. Header files in C programming are files that contain declarations of functions, variables, constants, and other constructs that are used in a program but are defined in other source files. Header files typically have a .h extension and are included in C source code files using #include directives. Header files serve several important purposes in C programming:

Modularization and Code Organization: Header files help organize code into modular units. By separating the declarations of functions and variables from the actual implementation, code becomes more maintainable and easier to read. Each header file typically corresponds to a module or component of a program.

Encapsulation: Header files provide a way to encapsulate the interface of a module. They hide the implementation details and only expose what's necessary for other parts of the program to use that module. This is crucial for building complex software systems where different parts interact with each other.

Reusability: Header files promote code reusability. Once you have a well-defined interface for a module in a header file, you can reuse it in multiple parts of your program or even in different programs altogether.

Avoiding Redundancy: By including the same header file in multiple source files, you can ensure that the same declarations are used consistently throughout your program. This helps prevent errors and inconsistencies.

Library Integration: Header files are used extensively when working with libraries in C. When you want to use functions or data types from a library, you include the corresponding header file to access the library's interface.

Compile-Time Type Checking: Header files provide information about the types of functions and variables, which allows the compiler to perform type checking. This helps catch type-related errors at compile time, reducing the likelihood of runtime errors.

Documentation: Header files often include comments and documentation that describe how to use the functions and variables declared within them. This serves as a reference for programmers using the module.

Common C header files include <stdio.h> (for input and output functions), <stdlib.h> (for standard library functions), <math.h> (for mathematical functions), and many others. Programmers can also create their own header files to encapsulate the functionality of custom modules or libraries.

In summary, header files in C are essential for organizing, modularizing, and maintaining code, as well as for defining interfaces between different parts of a program or between a program and external libraries. They play a crucial role in making C code more readable, maintainable, and reusable.

Q23. What will be the output of following program?

Ans. #include<stdio.h>
int main()
{ int num=070;
printf(鈥%d\t%o\t%x 鈥,num,num,num);
}

The output of the provided C program will be:

56 70 38
Here's an explanation of the output:

1. num is initialized with the value 070. In C, when you prefix a number with 0, it is considered an octal (base-8) literal. Therefore, 070 is interpreted as an octal number, not a decimal number.

2. %d in the printf statement prints the decimal representation of num, which is 56.

3. %o prints the octal representation of num, which is 70.

4. %x prints the hexadecimal representation of num, which is 38.

So, the program prints 56 (decimal), 70 (octal), and 38 (hexadecimal) separated by tabs.
}

Q 24. What will be the output of following program?
Ans.
#include <stdio.h>
void main()
{
int x = printf("GLA UNIVERSITY");
printf("%d", x);
}

The output of the provided C program will be:

GLA UNIVERSITY15
Here's an explanation of the output:

1. printf("GLA UNIVERSITY") is a printf statement that prints the string "GLA UNIVERSITY" to the console. The printf function returns the number of characters printed. In this case, there are 15 characters in the string "GLA UNIVERSITY."

2. int x = printf("GLA UNIVERSITY") assigns the return value of printf, which is 15, to the variable x.

3. printf("%d", x) then prints the value of x, which is 15, to the console.

So, the program first prints "GLA UNIVERSITY" and then prints the value 15, resulting in the output "GLA UNIVERSITY15."

Q25. What are library functions? List any four library functions.

Ans. Library functions, also known as standard library functions or built-in functions, are pre-written functions provided by the C programming language or other programming languages. These functions are part of a standard library and can be used by programmers to perform common tasks without having to write the code for those tasks from scratch. Library functions are typically organized into header files, and you include these headers in your C programs to access the functions they provide.

Here are four commonly used library functions in C:

1. printf: The printf function is used to format and print data to the standard output (usually the console). It allows you to display text and data with various formatting options

printf("Hello, World!\n");

2. scanf: The scanf function is used for reading formatted input from the standard input (usually the keyboard). It allows you to receive and store user input into variables.

```
int age;
scanf("%d", &age);
```

3. strlen: The strlen function is used to find the length (number of characters) of a string. It's part of the string manipulation library functions.

```
#include <string.h>
// ...
char str[] = "Hello";
int length = strlen(str); // length will be 5
```

4. rand: The rand function generates pseudo-random numbers. To use rand, you typically include the <stdlib.h> header.

```
#include <stdlib.h>
// ...
int random_number = rand();
```

These are just a few examples of the many library functions available in C. Library functions help simplify programming tasks and make it easier to develop efficient and maintainable code by providing pre-defined functionality for common operations.

Q26. What will be the output of following program?

Ans.
```
#include <stdio.h>
void main()
{
```

```
    int x = printf("C is placement oriented Language") 鈥 printf(鈥溹 i 鈥);
    printf("%d %o %x", x,x,x);
}
```

There is an issue with the provided code. It uses a strange character in the printf statement between "Hi" and printf. The character used is not the standard minus sign -, but rather an en-dash 鈥. This character is not recognized as an operator in C, so it will result in a compilation error.

Here's the corrected code with a standard minus sign:

```
#include <stdio.h>
void main()
{
    int x = printf("C is placement oriented Language") - printf("Hi");
    printf("%d %o %x", x, x, x);
}
```

With this corrected code, the output will be:

C is placement oriented LanguageHi27 33 1b
Here's an explanation of the corrected output:

1. The first printf statement prints "C is placement oriented Language" and returns the number of characters printed, which is 32.

2. The second printf statement prints "Hi" and returns the number of characters printed, which is 2.

3.The value of x is calculated as 32 - 2, which is 30.

4. The final printf statement prints the value of x in decimal, octal, and hexadecimal formats, which are 30, 36, and 1B in those respective bases.

Q27. What is the meaning of following statement? printf(鈥%d 鈥,scanf(鈥%d%d 鈥,&a,&b));

Ans. The statement printf("%d", scanf("%d%d", &a, &b)); is a bit unconventional and may not produce the expected results.

Here's what happens step by step:

1. scanf("%d%d", &a, &b) is used to read two integer values from the standard input (usually the keyboard) and store them in variables a and b. The scanf function returns the number of successfully read items, which in this case should be 2 if both integers are successfully read.

2. The result of scanf("%d%d", &a, &b) is then passed as an argument to printf("%d", ...). In this context, you are trying to print the result of scanf.

However, there are a couple of issues with this code:

. The printf function is used to format and print data, but scanf is used for input. Mixing these two functions in this way doesn't produce meaningful output because scanf returns the number of items successfully read, which may not be useful to print.

. The return value of scanf is meant for error checking, not for display. It is used to determine if the input was successfully read as expected.

If you want to check for errors or print the values read from the user, you should separate the scanf and printf statements and use the variables a and b to display the values. Here's an example of how you can do it:

```
#include <stdio.h>

int main() {
	int a, b;

	printf("Enter two integers: ");
	int num_read = scanf("%d%d", &a, &b);

	if (num_read == 2) {
		printf("You entered: a=%d, b=%d\n", a, b);
	} else {
		printf("Input error: Please enter two integers.\n");
	}

	return 0;
}
```

This code first reads two integers using scanf and checks if both integers were successfully read. If successful, it prints the values; otherwise, it displays an error message.

Q28. What will be the output of following program?

Ans.
```
#include <stdio.h>
void main()
{
   printf(" \"C %% FOR %% PLACEMENT\"");
}
```

The output of the provided C program will be:

```
 "C % FOR % PLACEMENT"
```

Here's an explanation of the output:

The printf statement contains a string with double quotation marks. Inside this string, the %% sequence is used to represent a literal % character. When the program is executed, it will print the string exactly as it appears in the printf statement, including the quotation marks. So, the output includes the double quotes and the escaped % characters, resulting in:

```
 "C % FOR % PLACEMENT"
```

The backslashes (\) before the double quotes are escape characters that indicate that the double quotes should be treated as literal characters to be printed, rather than as string delimiters.

Q29. Suppose distance between GLA University and Delhi is m km (to be entered by user), by BUS you can reach Delhi in 4 hours. Develop a 鈥楢鈥 program to calculate speed of bus.

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

int main() {
	double distance;    // Distance in kilometers
	double time = 4.0;    // Time in hours
```

```
    // Input distance from the user
    printf("Enter the distance between GLA University and Delhi (in kilometers): ");
    scanf("%lf", &distance);

    // Calculate the speed
    double speed = distance / time;

    // Output the speed
    printf("The speed of the bus is %.2lf km/h\n", speed);

    return 0;
}
```

Q30. In an exam Satyam got 50 marks, Suman got 70 marks and Shyam got 80 marks, Write a 鈥榗鈥 program to find average marks of these three participants.

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

int main() {
    int satyam_marks = 50;
    int suman_marks = 70;
    int shyam_marks = 80;

    // Calculate the total marks
    int total_marks = satyam_marks + suman_marks + shyam_marks;

    // Calculate the average marks
    int num_participants = 3;
    double average_marks = (double)total_marks / num_participants;

    // Output the average marks
    printf("Average Marks: %.2lf\n", average_marks);

    return 0;
}
```

Q31. One day, Mohan called Saurav and Sajal and gave some money to them, later he realized that money that was given to Saurav should be given to Sajal and vice-versa. Develop a 鈥榗鈥 program to help Mohan so that he can rectify his mistake.

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

int main() {
    double saurav_money, sajal_money, temp;

    // Input the amounts initially given to Saurav and Sajal
    printf("Enter the amount given to Saurav: ");
    scanf("%lf", &saurav_money);

    printf("Enter the amount given to Sajal: ");
    scanf("%lf", &sajal_money);

    // Swap the amounts
    temp = saurav_money;
    saurav_money = sajal_money;
```

```
    sajal_money = temp;

    // Output the corrected amounts
    printf("After rectifying the mistake:\n");
    printf("Amount given to Saurav: %.2lf\n", saurav_money);
    printf("Amount given to Sajal: %.2lf\n", sajal_money);

    return 0;
}
```

Q32. One day when I was going for a lunch, suddenly rain started, I was very hungry so started running with speed of 4km/h and it took 3 min to reach mess. Help me to develop a 鈥榗鈥 program to calculate distance travelled by me.

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

int main() {
    // Given speed in km/h and time in minutes
    double speed_kmph = 4.0;
    double time_minutes = 3.0;

    // Convert speed to meters per minute
    double speed_mpm = (speed_kmph * 1000) / 60;

    // Calculate the distance
    double distance = speed_mpm * time_minutes;

    // Output the distance
    printf("Distance traveled: %.2lf meters\n", distance);

    return 0;
}
```

Q33. Can two or more escape sequences such as \n and \t be combined in a single line of program code?

Ans. Yes, you can combine multiple escape sequences in a single line of program code in C or many other programming languages that support escape sequences. Escape sequences are special character sequences that are used to represent characters that are difficult to include directly in a string. For example:

\n represents a newline character.
\t represents a tab character.

You can use multiple escape sequences together to format your output or create specific patterns. For instance:

```
printf("Hello, World!\n\tThis is a tabbed line.\n");
```

In the code above, \n is used to create a newline, and \t is used to insert a tab character. When you run this code, it will produce the following output:

```
Hello, World!
        This is a tabbed line.
```

So, yes, you can combine multiple escape sequences in a single line to control the formatting and layout of your text or to represent special characters in your strings.

Q34. What are comments and how do you insert it in a C program?

Ans. Comments in a C program are non-executable statements that provide human-readable explanations or annotations within the code. Comments are ignored by the compiler and do not affect the program's functionality; they are used solely for documentation and clarification purposes. Comments help programmers and other readers understand the code's logic, purpose, and important details.

There are two common types of comments in C:

1. Single-Line Comments: Single-line comments start with // and continue until the end of the line. They are often used for brief explanations or comments on a single line of code.

```
// This is a single-line comment
int x = 5; // Assign 5 to variable x
```

2. Multi-Line Comments: Multi-line comments, also known as block comments, start with /* and end with */. They can span multiple lines and are suitable for longer explanations or comments that cover multiple lines of code.

```
/* This is a multi-line comment.
    It can span multiple lines.
    Use it for longer explanations or comments. */
int y = 10;
```

To insert comments in a C program:

. For single-line comments, simply start a line with // and write your comment after it.

. For multi-line comments, use /* to start the comment block and */ to end it. Everything between /* and */ is treated as a comment.

Comments should be used regularly in your code to provide context, explain complex logic, document functions and variables, and make your code more understandable to both yourself and others who may read it.

Q35. What is wrong in this statement? scanf(鈥%d 鈥,number);

Ans. The issue with the statement scanf("%d", number); is that the scanf function expects the address of the variable where it should store the input value, but it's not provided correctly.

To fix this, you need to use the & operator before the variable name number to provide the address of the variable to scanf. Here's the corrected statement:

```
scanf("%d", &number);
```

In C, when you read input using scanf for variables that have a specific type, you need to use & to provide the memory location where scanf should store the value. & is the address-of operator, and it is used to get the memory address of a variable.

Q36. What will be the output?

Ans.
```
#include <stdio.h>
int main()
```

```
{
    if (sizeof(int) > -1)
        printf("Yes");
    else
        printf("No");
    return 0;
}
```

The output of the provided C program will be:

Yes

Here's an explanation of why "Yes" is printed:

The sizeof operator in C is used to determine the size (in bytes) of a data type or an expression. In this case, sizeof(int) is used to determine the size of the int data type.

The if condition checks whether the size of an int is greater than -1.

In C, the sizeof operator always returns a non-negative value. It represents the size of a data type in bytes, so it cannot be negative.

Since the condition sizeof(int) > -1 is always true, the printf("Yes") statement is executed, and "Yes" is printed to the console.

Q37. Point out which of the following variable names are invalid:
gross-salary INTEREST , salary of emp , avg. , thereisbookinmysoup

Ans. Here are the evaluations of the provided variable names:

1. gross-salary: This variable name is invalid. In C and many other programming languages, variable names cannot contain hyphens (-). You can use underscores (_) instead to create a valid variable name, like gross_salary.

2. INTEREST: This variable name is valid. Variable names are case-sensitive in C, so INTEREST and interest are treated as distinct variables.

3. salary of emp: This variable name is invalid. Variable names cannot contain spaces. You can use underscores or combine the words without spaces, like salary_of_emp or salaryOfEmp.

4. avg.: This variable name is invalid. Variable names cannot contain periods (dots) except for the special use of the dot operator in some contexts. You should avoid using periods in variable names. You could use underscores instead, like avg_value.

5. thereisbookinmysoup: This variable name is valid. It doesn't contain any special characters or spaces, and it adheres to the rules for variable names in C.

Q38. Tom works at an aquarium shop on Saturdays. One Saturday, when Tom gets to work, he is asked to clean a 175-gallon reef tank. His first job is to drain the tank. He puts a hose into the tank and starts a siphon. Tom wonders if the tank will finish draining before he leaves work. He measures the amount of water that is draining out and finds that 12.5 gallons drain out in 30 minutes. So, he figures that the rate is 25 gallons per hour. Develop a 鈥棈鈥 program to help Tom to calculate time required to completely clean tank.

Ans. #include <stdio.h>

```
int main() {
    // Define the total gallons in the tank and the draining rate
    int totalGallons = 175;
    float drainingRatePerHour = 25.0; // Use a floating-point value for accuracy

    // Calculate the time required in hours to drain the tank completely
    float timeRequired = totalGallons / drainingRatePerHour;

    // Display the result
    printf("It will take %.2f hours to completely drain the tank.\n", timeRequired);

    return 0;
}
```

Q39. The percent y (in decimal form) of battery power remaining x hours after you turn on a laptop computer is y = 鉴0.2 x + 1. Develop a 欽榱欽 program to calculate after how many hours the battery power is at 75%?

Ans.

```
#include <stdio.h>

int main() {
    // Define the desired battery power (75% in decimal form)
    float desiredBatteryPower = 0.75;

    // Calculate the number of hours (x) when battery power is at 75%
    float x = (1 - desiredBatteryPower) / -0.2;

    // Display the result
    printf("The battery power will be at 75%% after %.2f hours.\n", x);

    return 0;
}
```

Q40.Which of the following is used to convert the high level language in machine language in a single go?

a. Compiler b.Interpreter
c. Linker d.Assembler

Ans. The correct answer is:

a. Compiler

Q 41. What is the format specifier for an Octal Number?

a.%0 b.%d
c. %o d. %e

Ans. The correct format specifier for an octal number in C is:

c. %o

Q 42. Which format specifier is used to print the exponent value upto 2 decimal places.

a. %e b.%.2f c. %f d.%.2e

v. The correct format specifier to print the exponent value up to 2 decimal places is:

d. %.2e

Q 43. Which of the following is not a basic data type?
a. char
b. array
c. float
d. int

Ans. b. array

Q 44. What is the output of following code?

```
#include<stdio.h>
void main()
{
   int x=0;
   x= printf("\"hello\b\"");
   printf(鈥%d 鈥,x);
}
```

a. hello7 b. 鈥涉 ello 鈥7 c. 鈥涉 ell 鈥8 d. hell8

Ans. The output of the given code will be:

c. "hell"8

Q 45. What is the output of following code?

```
#include<stdio.h>
void main()
{
   int b,c=5 ;
   int(鈥%d , %d 鈥, b,c);
}
```

a. 5, 5 b. 5, 5.000000

Ans.

```
#include<stdio.h>
int main()
{
   int b, c = 5;
   printf("%d , %d", b, c);
   return 0;
}
```

With the corrected code, the output will be:

a. 5, 5

Q46. Which of the following is an identifier?
a. &fact b. Basic_pay c. enum d. 1sum

Ans. the correct answer is:

b. Basic_pay

Q 47. What is the output of the following program?

```
#include<stdio.h>
void main()
```

```
{
    char x, a=鈥樷鈥;
    x=printf("%c",a);
    printf(鈥%d 鈥,x);
}
```

a. c1 b. cgarbage
c. 1 c. c

```
Ans.#include<stdio.h>
int main()
{
    char x, a='c';
    x = printf("%c", a);
    printf("%d", x);
    return 0;
}
```

With these corrections, the output will be:

a. c1

Q48. Perform the following conversion from Decimal to other number as directed-
a) (365.55)10 = (?)2
b) (453.65)10 = (?)8
c) (5164.12)10 = (?)16
d) (23.65)10 = (?)5
e) (772)10 = (?)7

Ans. a) (365.55)10 = (?)2 (Binary)
To convert the integer part, repeatedly divide by 2 and keep track of the remainders. For the fractional part, repeatedly multiply by 2 and keep track of the integer parts. Here's the conversion:

Integer part:
365 / 2 = 182 remainder 1
182 / 2 = 91 remainder 0
91 / 2 = 45 remainder 1
45 / 2 = 22 remainder 1
22 / 2 = 11 remainder 0
11 / 2 = 5 remainder 1
5 / 2 = 2 remainder 1
2 / 2 = 1 remainder 0
1 / 2 = 0 remainder 1

So, the integer part in binary is 101101101.

Fractional part:
0.55 * 2 = 1.10 (0)
0.10 * 2 = 0.20 (0)
0.20 * 2 = 0.40 (0)
0.40 * 2 = 0.80 (0)
0.80 * 2 = 1.60 (1)
0.60 * 2 = 1.20 (1)
0.20 * 2 = 0.40 (0)
0.40 * 2 = 0.80 (0)
0.80 * 2 = 1.60 (1)

So, the fractional part in binary is 10011001100110011001100110011...

Combining the two parts, we get (365.55)10 = (101101101.10011001100110011...)2.

b) (453.65)10 = (?)8 (Octal)
To convert to octal, you can follow a similar process, but divide by 8 for the integer part and multiply by 8 for the fractional part. The conversion is as follows:

Integer part:
453 / 8 = 56 remainder 5
56 / 8 = 7 remainder 0
7 / 8 = 0 remainder 7

So, the integer part in octal is 750.

Fractional part:
0.65 * 8 = 5.20 (5)
0.20 * 8 = 1.60 (1)
0.60 * 8 = 4.80 (4)
0.80 * 8 = 6.40 (6)
0.40 * 8 = 3.20 (3)
0.20 * 8 = 1.60 (1)

So, the fractional part in octal is 515461.

Combining the two parts, we get (453.65)10 = (750.515461)8.

c) (5164.12)10 = (?)16 (Hexadecimal)
To convert to hexadecimal, you can divide by 16 for the integer part and multiply by 16 for the fractional part. The conversion is as follows:

Integer part:
5164 / 16 = 322 remainder 12 (C)
322 / 16 = 20 remainder 2

So, the integer part in hexadecimal is 20C.

Fractional part:
0.12 * 16 = 1.92 (1)
0.92 * 16 = 14.72 (E)
0.72 * 16 = 11.52 (B)
0.52 * 16 = 8.32 (8)
0.32 * 16 = 5.12 (5)
0.12 * 16 = 1.92 (1)

So, the fractional part in hexadecimal is 1EB85...

Combining the two parts, we get (5164.12)10 = (20C.1EB851EB851EB851...)16.

d) (23.65)10 = (?)5 (Quinary)
To convert to quinary (base-5), follow a similar process as before:

Integer part:
23 / 5 = 4 remainder 3

So, the integer part in quinary is 43.

Fractional part:
0.65 * 5 = 3.25 (3)
0.25 * 5 = 1.25 (1)
0.25 * 5 = 1.25 (1)
0.25 * 5 = 1.25 (1)
0.25 * 5 = 1.25 (1)

So, the fractional part in quinary is 31111...

Combining the two parts, we get (23.65)10 = (43.31111...)5.

e) (772)10 = (?)7 (Septenary)
To convert to septenary (base-7), simply divide the number by 7:

772 / 7 = 110 remainder 2

So, the number in septenary is 2112.

Q49. Covert the following numbers to decimal number system-
a) (325.54)6 = (?)10
b) (1001010110101.1110101)2 = (?)10
c) (742.72)8 = (?)10
d) (AC94.C5)16 = (?)10

Ans. To convert numbers from different bases to the decimal number system, you can use the following methods:

a) (325.54)6 = (?)10 (Hexadecimal)
To convert a number from base-6 to decimal, use the following formula:

Decimal = d0 * 6^0 + d1 * 6^1 + d2 * 6^2 + ...
where d0, d1, d2, ... are the digits of the number from right to left.

So, for (325.54)6:

Decimal = 4 * 6^0 + 5 * 6^1 + 5 * 6^2 + 2 * 6^3 + 3 * 6^4 = 4 + 30 + 180 + 432 + 972 = 1618

b) (1001010110101.1110101)2 = (?)10 (Binary)
To convert a binary number to decimal, use the following formula:

Decimal = d0 * 2^0 + d1 * 2^1 + d2 * 2^2 + ...
where d0, d1, d2, ... are the binary digits (0 or 1) of the number from right to left.

So, for (1001010110101.1110101)2:

Decimal = 1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 0 * 2^3 + 1 * 2^4 + 0 * 2^5 + 1 * 2^6 + 1 * 2^7 + 0 * 2^8 + 1 * 2^9 + 0 * 2^10 + 1 * 2^11 + 1 * 2^12 + 1 * 2^(-1) + 1 * 2^(-2) + 0 * 2^(-3) + 1 * 2^(-4) + 0 * 2^(-5) + 1 * 2^(-6) = 5453.8125

c) (742.72)8 = (?)10 (Octal)
To convert an octal number to decimal, use the following formula:

Decimal = d0 * 8^0 + d1 * 8^1 + d2 * 8^2 + ...
where d0, d1, d2, ... are the octal digits (0 to 7) of the number from right to left.

So, for (742.72)8:

Decimal = 2 * 8^0 + 7 * 8^1 + 4 * 8^2 + 7 * 8^(-1) + 2 * 8^(-2) = 2 + 56 + 256 + 0.875 + 0.25 = 315.125

d) (AC94.C5)16 = (?)10 (Hexadecimal)
To convert a hexadecimal number to decimal, use the following formula:

Decimal = d0 * 16^0 + d1 * 16^1 + d2 * 16^2 + ...
where d0, d1, d2, ... are the hexadecimal digits (0 to 9 and A to F) of the number from right to left.

So, for (AC94.C5)16:

Decimal = 5 * 16^0 + C * 16^1 + 4 * 16^2 + 9 * 16^3 + A * 16^(-1) + C * 16^(-2) = 5 + 192 + 1024 + 2304 + 10 * 16^(-1) + 12 * 16^(-2) = 6145.7734375
These are the decimal equivalents for the given numbers in various base systems.

Q50. Perform the following conversion from Hexadecimal to other number as directed-
(DB56.CD4)16 = (?)2, (?)8, (?)4

Ans. To convert a hexadecimal number (DB56.CD4)16 to other number systems, follow these steps:

a) Convert to Binary (Base-2):
To convert a hexadecimal digit to binary, replace it with its 4-bit binary representation:

D = 1101
B = 1011
5 = 0101
6 = 0110
C = 1100
D = 1101
4 = 0100

Now, combine these binary representations together, including the binary point:

(DB56.CD4)16 = (1101101101101010.110011010100)2

b) Convert to Octal (Base-8):
Group the binary digits into sets of 3, starting from the binary point:

(DB56.CD4)16 = (001 101 101 101 101 010.110 011 010 100)2

Now, convert each group of 3 binary digits to octal:

001 = 1
101 = 5
101 = 5
101 = 5
010 = 2
110 = 6
011 = 3
010 = 2
100 = 4

Combine these octal digits together, including the octal point:

(DB56.CD4)16 = (15552.6324)8

c) Convert to Quaternary (Base-4):
Group the binary digits into sets of 2, starting from the binary point:

(DB56.CD4)16 = (00 11 01 10 11 01 10 11 01 01 00.11 00 11 01 01 00)2

Now, convert each group of 2 binary digits to quaternary:

00 = 0
11 = 3
01 = 1
10 = 2
11 = 3
01 = 1
10 = 2
11 = 3
01 = 1
01 = 1
00 = 0
11 = 3
00 = 0
11 = 3
01 = 1
01 = 1
00 = 0

Combine these quaternary digits together, including the quaternary point:

(DB56.CD4)16 = (03123120110300.31130011)4

So, the conversion of (DB56.CD4)16 to other number systems is as follows:

.In binary (Base-2): (1101101101101010.110011010100)2
.In octal (Base-8): (15552.6324)8
.In quaternary (Base-4): (03123120110300.31130011)4

Q51. Perform the following conversion from octal to other number as directed-
(473.42)8 = (?)2, (?)10, (?)16, (?)5
Q52. Find the value of A?
a) (23)10 = (17)A
b) (21)16 = (41)A
c) (32)8 = (101)A

Ans. Perform the following conversion from octal to other number systems:

a) To Binary (Base-2):
To convert an octal number to binary, you can convert each octal digit to its 3-bit binary equivalent. Here's the conversion for (473.42)8:

4 = 100
7 = 111
3 = 011
. (decimal point remains the same)

4 = 100
2 = 010

Combine the binary equivalents:

(473.42)8 = (100111011.100010)2

b) To Decimal (Base-10):
To convert an octal number to decimal, you can use the following formula:

Decimal = d0 * 8^0 + d1 * 8^1 + d2 * 8^2 + ...

Where d0, d1, d2, ... are the octal digits from right to left.

(473.42)8 = 2 * 8^0 + 4 * 8^1 + 1 * 8^2 + 0 * 8^(-1) + 2 * 8^(-2)
(473.42)8 = 2 + 32 + 64 + 0 + 0.25 + 0.015625
(473.42)8 = 98.265625

c) To Hexadecimal (Base-16):
First, convert the octal number to binary, and then group the binary digits into sets of 4 (starting from the binary point) and convert each group to hexadecimal.

(473.42)8 = (100111011.100010)2

Grouping into sets of 4 from the binary point:

1001 1101 . 1000 10

Now, convert each group to hexadecimal:

1001 = 9
1101 = D
1000 = 8
10 = 2

Combine the hexadecimal groups:

(473.42)8 = (9D.82)16

d) To Quinary (Base-5):
To convert an octal number to quinary, you can convert each octal digit to its 3-bit binary equivalent and then group the binary digits into sets of 2 (starting from the binary point) to convert to quinary.

4 = 100
7 = 111
3 = 011
. (decimal point remains the same)
4 = 100
2 = 010

Combine the binary equivalents and group into sets of 2:

(473.42)8 = (100111011.100010)2
Grouping: 10 01 11 01 11 . 10 00 10

Now, convert each group to quinary:

10 = 2
01 = 1
11 = 3
01 = 1
11 = 3
. (decimal point remains the same)
10 = 2
00 = 0
10 = 2

Combine the quinary groups:

(473.42)8 = (21313.202)5

Q52. Find the value of A in each equation:
a) (23)10 = (17)A
b) (21)16 = (41)A
c) (32)8 = (101)A

Ans. a) (23)10 = (17)A
To find the value of A, set up an equation and solve for A:

23 = 1A + 7
23 = A + 7
A = 23 - 7
A = 16

So, A = 16.

b) (21)16 = (41)A
To find the value of A, set up an equation and solve for A:

21 = 4A + 1
21 = 4A + 1
4A = 21 - 1
4A = 20
A = 20 / 4
A = 5

So, A = 5.

c) (32)8 = (101)A
To find the value of A, set up an equation and solve for A:

32 = 1A + 0
32 = A
A = 32

So, A = 32.

Q53: What will be the output of following program? Assume integer is of 2 bytes

```
void main(){
int    a=32770;
```

```
printf(鈥%d 鈥,a);
}
```

Ans. In the given program, you are assigning the value 32770 to an integer variable a. Since an integer typically uses 2 bytes of memory, its range is usually -32,768 to 32,767 for a signed integer.

The value 32770 exceeds the maximum value that can be stored in a 2-byte signed integer, and it will cause an overflow. When overflow occurs, the behavior is undefined in C, and the result may not be predictable.

However, in many compilers and systems, the value will "wrap around" to the minimum value for a 2-byte signed integer, which is -32,768. So, the output of the program is likely to be:

-32768

But please note that the behavior is compiler and platform-dependent, and in some cases, you may get different results or encounter undefined behavior. It's generally a good practice to ensure that values assigned to variables stay within their valid ranges to avoid such issues.

Q54:
```
#include <stdio.h>
 int main()
{
 float c = 5.0;
 printf ("Temperature in Fahrenheit is %.2f", (9/5)*c + 32);
 return 0;
}
```

The given C program calculates and prints the temperature in Fahrenheit based on a given temperature in Celsius (5.0 degrees Celsius in this case).

Here's how the program works:

1. float c = 5.0; declares a floating-point variable c and assigns it the value of 5.0 degrees Celsius.

2. printf("Temperature in Fahrenheit is %.2f", (9/5)*c + 32); calculates the temperature in Fahrenheit using the formula (9/5)*c + 32 and prints it with two decimal places.

3. return 0; indicates a successful execution of the program.

However, there is a potential issue with the code. In the expression (9/5)*c, the division is performed with integer values (9 and 5), and the result is an integer value (which is 1 in this case) before being multiplied by c. To ensure that the division is performed as floating-point division, you should use floating-point literals (9.0 and 5.0) like this:

```
printf("Temperature in Fahrenheit is %.2f", (9.0/5.0)*c + 32);
```

With this change, the program will correctly calculate and display the temperature in Fahrenheit.