Subject Name & Code:

PROGRAMMING FOR PROBLEM SOLVING- BE01R00121

LAB MANUAL PRACTICAL SOLUTION

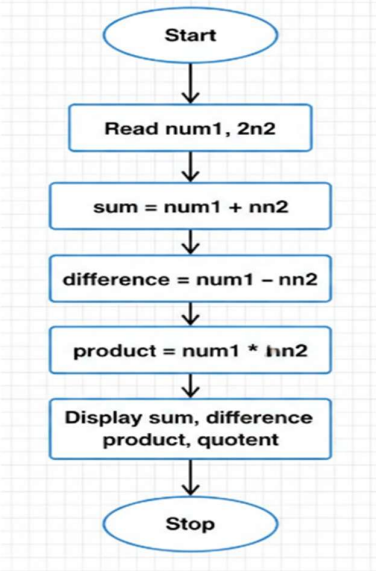
Practical - 1

Q-A: Accepting two numbers and performing addition, subtraction, division, and multiplication on them.

Answer:

Algorithm:

1. **Start**
2. **Read** two numbers, num1 and num2.
3. **Compute** sum = num1 + num2.
4. **Compute** difference = num1 - num2.
5. **Compute** product = num1 * num2.
6. **Compute** quotient = num1 / num2. (Assume num2 != 0)
7. **Print** sum, difference, product, and quotient.
8. **Stop**

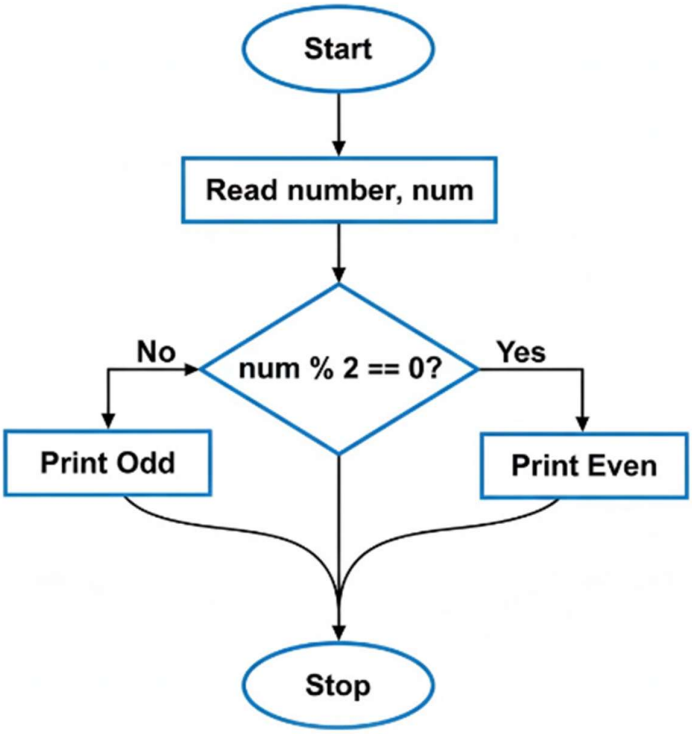


Q-B: To check whether a given number is even or odd.

Answer:

Algorithm:

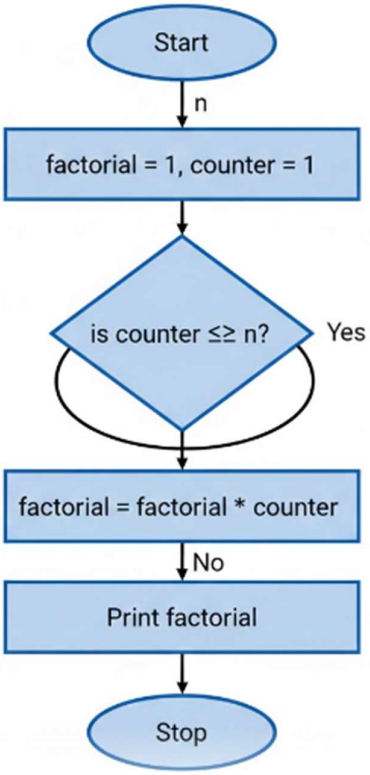
1. **Start**
2. **Read** a number, num.
3. **If** num % 2 == 0 **Then**
4. **Print** "The number is Even."
5. **Else**
6. **Print** "The number is Odd."
7. **End If**
8. **Stop**



Q-C: To find out the factorial of a given number "n".

Answer:

1. **Start**
2. **Read** a number, n.
3. **Initialize** factorial = 1 and counter = 1.
4. **Repeat** Steps 5 and 6 while counter <= n.
5. **Compute** factorial = factorial * counter.
6. **Increment** counter = counter + 1.
7. **Print** factorial.
8. **Stop**


Start
n
factorial = 1, counter = 1
is counter ≤≥ n?
Yes
factorial = factorial * counter
No
Print factorial
Stop

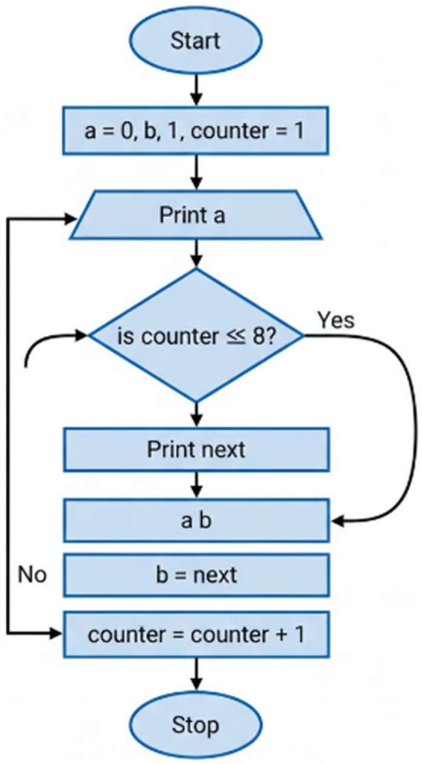
Practical - 2

Q-A: To print first 10 Fibonacci numbers.

Answer:

Algorithm:

1. **Start**
2. **Initialize** a = 0, b = 1, counter = 1.
3. **Print** a.
4. **Print** b.
5. **Repeat** Steps 6 to 9 while counter <= 8. (Because we print 2 numbers already)
6. **Compute** next = a + b.
7. **Print** next.
8. **Set** a = b.
9. **Set** b = next.
10. **Increment** counter = counter + 1.
11. **Stop**

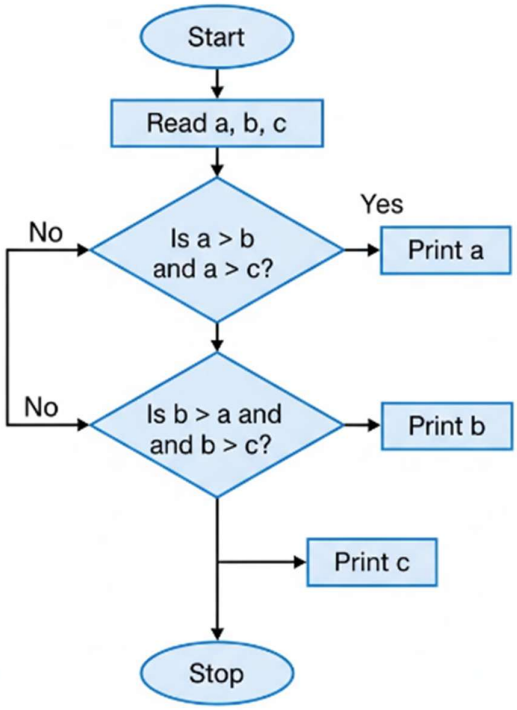


Q-B: To find out the largest number from three numbers.

Answer:

Algorithm:

1. **Start**
2. **Read** three numbers, a, b, and c.
3. **If** a > b **And** a > c **Then**
4. **Print** "Largest number is", a.
5. **Else If** b > a **And** b > c **Then**
6. **Print** "Largest number is", b.
7. **Else**
8. **Print** "Largest number is", c.
9. **End If**
10. **Stop**



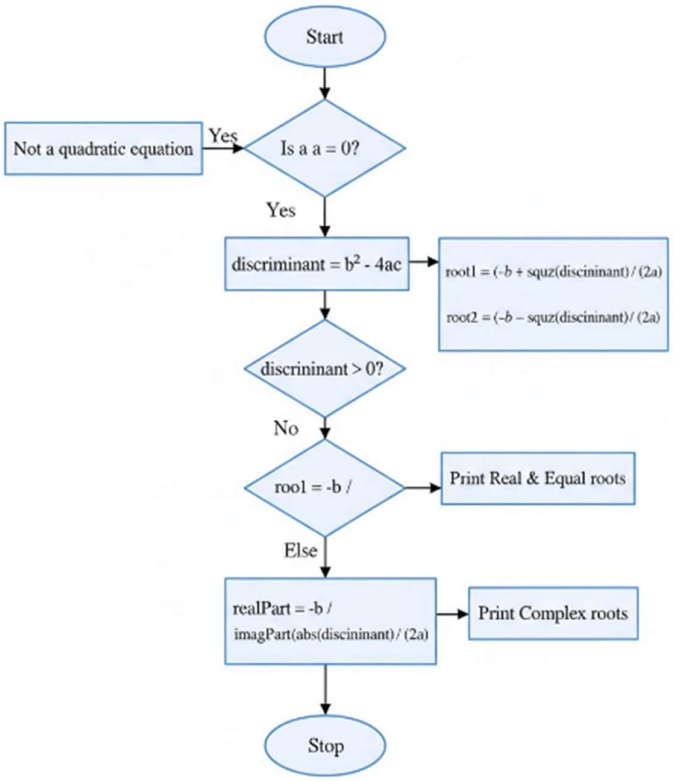
Q-C: To find out roots of a quadratic equation ax^2 + bx + c.

Answer:

Algorithm:

1. **Start**
2. **Read** coefficients a, b, and c.
3. **If** a == 0 **Then**
4. **Print** "Not a quadratic equation."
5. **Stop**
6. **End If**
7. **Compute** discriminant = b*b - 4*a*c.
8. **If** discriminant > 0 **Then**
9. **Compute** root1 = (-b + sqrt(discriminant)) / (2*a).
10. **Compute** root2 = (-b - sqrt(discriminant)) / (2*a).
11. **Print** "Real and Distinct Roots: root1, root2".
12. **Else If** discriminant == 0 **Then**

13. **Compute** root1 = -b / (2*a).
14. **Print** "Real and Equal Roots: root1, root1".
15. **Else**
16. **Compute** realPart = -b / (2*a).
17. **Compute** imagPart = sqrt(-discriminant) / (2*a).
18. **Print** "Complex Roots: realPart ± i*imagPart".
19. **End If**
20. **Stop**



Practical - 3

Q-A: Write a program that performs as a calculator (addition, multiplication, division, subtraction).

Answer:

```
#include <stdio.h>

int main() {
    float num1, num2;
    char operator;

    // Read input from user
    printf("Enter first number: ");
    scanf("%f", &num1);
    printf("Enter an operator (+, -, *, /): ");
    scanf(" %c", &operator); // Note the space before %c to consume any newline character
    printf("Enter second number: ");
    scanf("%f", &num2);

    // Perform calculation based on the operator
    switch(operator) {
        case '+':
            printf("%.2f + %.2f = %.2f\n", num1, num2, num1 + num2);
            break;
        case '-':
```

```
            printf("%.2f - %.2f = %.2f\n", num1, num2, num1 - num2);
            break;
        case '*':
            printf("%.2f * %.2f = %.2f\n", num1, num2, num1 * num2);
            break;
        case '/':
            if(num2 != 0) {
                printf("%.2f / %.2f = %.2f\n", num1, num2, num1 / num2);
            } else {
                printf("Error! Division by zero is not allowed.\n");
            }
            break;
        default:
            printf("Error! Invalid operator entered.\n");
    }

    return 0;
}
```

Q-B: Write a program to find the area of a triangle (a = h * b * 0.5).

Answer:

```
#include <stdio.h>

int main() {
    float height, base, area;
```

```
    // Read height and base from user
    printf("Enter the height of the triangle: ");
    scanf("%f", &height);
    printf("Enter the base of the triangle: ");
    scanf("%f", &base);

    // Calculate area
    area = 0.5 * height * base;

    // Display the result
    printf("The area of the triangle with height %.2f and base %.2f is: %.2f\n",
height, base, area);

    return 0;
}
```

Q-C: Write a C program to compute simple interest using the formula Si = p * r * n / 100.

Answer:

```
#include <stdio.h>

int main() {
    float principal, rate, time, simple_interest;

    // Read principal amount, rate of interest, and time period from user
    printf("Enter the principal amount: ");
```

```
    scanf("%f", &principal);
    printf("Enter the rate of interest (%% per year): ");
    scanf("%f", &rate);
    printf("Enter the time period (in years): ");
    scanf("%f", &time);

    // Calculate simple interest
    simple_interest = (principal * rate * time) / 100;

    // Display the result
    printf("Simple Interest = %.2f\n", simple_interest);

    return 0;
}
```

Practical - 4

Q-A: Write a C program to interchange two numbers.

Answer:

```
#include <stdio.h>

int main() {
    int a, b, temp;

    // Read two numbers from user
    printf("Enter value for a: ");
    scanf("%d", &a);
    printf("Enter value for b: ");
    scanf("%d", &b);

    // Display numbers before swapping
    printf("\nBefore swapping:\n");
    printf("a = %d, b = %d\n", a, b);

    // Swapping logic
    temp = a; // Step 1: Store value of 'a' in 'temp'
    a = b;    // Step 2: Assign value of 'b' to 'a'
    b = temp; // Step 3: Assign original value of 'a' (from 'temp') to 'b'

    // Display numbers after swapping
    printf("\nAfter swapping:\n");
    printf("a = %d, b = %d\n", a, b);
```

```
    return 0;
}
```

Q-B: Write a program to compute Fahrenheit from centigrade (f = 1.8 * c + 32).

Answer:

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit;

    // Read temperature in Celsius from user
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);

    // Convert Celsius to Fahrenheit
    fahrenheit = (1.8 * celsius) + 32;

    // Display the result
    printf("%.2f degrees Celsius is equal to %.2f degrees Fahrenheit.\n",
celsius, fahrenheit);

    return 0;
}
```

Q-C: Write a C program to calculate years, months, and days passed for a given number of days.

Answer:

```
#include <stdio.h>

int main() {
    int total_days, years, months, remaining_days;

    // Read total number of days from user
    printf("Enter the total number of days: ");
    scanf("%d", &total_days);

    // Calculate years
    years = total_days / 365;        // Step 1: Find total years
    remaining_days = total_days % 365; // Step 2: Find remaining days after years

    // Calculate months from the remaining days
    months = remaining_days / 30;    // Step 3: Find total months from remaining days
    remaining_days = remaining_days % 30; // Step 4: Find final remaining days

    // Display the result
    printf("\nBreakdown for %d days:\n", total_days);
    printf("Years: %d\n", years);
    printf("Months: %d\n", months);
    printf("Days: %d\n", remaining_days);
```

```
    return 0;
}
```

Practical - 5

Q-A: Write a program to read marks of a student from keyboard whether the student is pass or fail (using if else).

Answer:

```
#include <stdio.h>

int main() {
   float marks;

   // Read marks from user
   printf("Enter the student's marks: ");
   scanf("%f", &marks);

   // Check pass or fail condition
   if(marks >= 40) {
      printf("Result: PASS\n");
   } else {
      printf("Result: FAIL\n");
   }

   return 0;
}
```

Q-B: Write a program to read three numbers from keyboard and find out maximum out of these three. (nested if else)

Answer:

```
#include <stdio.h>

int main() {
    float a, b, c;

    // Read three numbers from user
    printf("Enter three numbers: ");
    scanf("%f %f %f", &a, &b, &c);

    // Find the maximum using nested if-else
    if (a >= b) {
        if (a >= c) {
            printf("The largest number is %.2f\n", a);
        } else {
            printf("The largest number is %.2f\n", c);
        }
    } else {
        if (b >= c) {
            printf("The largest number is %.2f\n", b);
        } else {
            printf("The largest number is %.2f\n", c);
        }
    }

    return 0;
}
```

Q-C: Write a C program that uses else-if ladder to check whether entered character is uppercase, lowercase, digit or special character.

Answer:

```
#include <stdio.h>
#include <ctype.h> // For isupper(), islower(), isdigit() functions

int main() {
    char ch;

    // Read a character from user
    printf("Enter a character: ");
    scanf("%c", &ch);

    // Check the type of character using else-if ladder
    if (isupper(ch)) {
        printf("'%c' is an Uppercase letter.\n", ch);
    } else if (islower(ch)) {
        printf("'%c' is a Lowercase letter.\n", ch);
    } else if (isdigit(ch)) {
        printf("'%c' is a Digit.\n", ch);
    } else {
        printf("'%c' is a Special character.\n", ch);
    }

    return 0;
}
```

Practical - 6

Q-A: Write a C program to read no 1 to 7 and print relatively day Sunday to Saturday. (Switch statement)

Answer:

```
#include <stdio.h>

int main() {
    int dayNumber;

    // Read day number from user
    printf("Enter a number (1-7): ");
    scanf("%d", &dayNumber);

    // Map number to day using switch
    switch(dayNumber) {
        case 1:
            printf("Sunday\n");
            break;
        case 2:
            printf("Monday\n");
            break;
        case 3:
            printf("Tuesday\n");
            break;
        case 4:
            printf("Wednesday\n");
```

```
            break;
        case 5:
            printf("Thursday\n");
            break;
        case 6:
            printf("Friday\n");
            break;
        case 7:
            printf("Saturday\n");
            break;
        default:
            printf("Invalid input! Please enter a number between 1 and 7.\n");
    }

    return 0;
}
```

Q-B: Write a C program to find factorial of a given number. (Using Iteration)

Answer:

```
#include <stdio.h>

int main() {
    int n, i;
    unsigned long long factorial = 1; // Use long long for large numbers

    // Read the number from user
    printf("Enter a positive integer: ");
```

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

    // If the user enters a negative number, show error
    if (n < 0) {
        printf("Error! Factorial is not defined for negative numbers.\n");
    } else {
        // Calculate factorial using a for loop (Iteration)
        for (i = 1; i <= n; ++i) {
            factorial *= i; // factorial = factorial * i;
        }
        printf("Factorial of %d = %llu\n", n, factorial);
    }

    return 0;
}
```

Q-C: Write a program to print the following patterns:

Answer:

Solution for (i):

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

```
#include <stdio.h>

int main() {
```

```
    int i, j, rows = 5;

    for (i = 1; i <= rows; ++i) {       // Outer loop for each row
        for (j = 1; j <= i; ++j) {      // Inner loop for stars in each row
            printf("*");
        }
        printf("\n"); // Move to the next line after each row
    }
    return 0;
}
```

Solution for (ii):

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

```
#include <stdio.h>

int main() {
    int i, j, rows = 5;

    for (i = 1; i <= rows; ++i) {
        // Loop to print leading spaces
        for (j = 1; j <= rows - i; ++j) {
            printf(" ");
        }
        // Loop to print stars
```

```
        for (j = 1; j <= i; ++j) {
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Solution for (iii):

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

int main() {
    int i, j, rows = 5;

    for (i = rows; i >= 1; --i) {      // Outer loop decrements from rows to 1
        for (j = 1; j <= i; ++j) {     // Inner loop prints i stars
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

Q-D: Write a program that prints following patterns:

Answer:

Solution for i):

A

AB

ABC

```
#include <stdio.h>

int main() {
   int i, j;
   char input = 'C'; // Change this to print more/less rows
   int rows = input - 'A' + 1; // Calculate number of rows based on last character

   for (i = 0; i < rows; ++i) {
      for (j = 0; j <= i; ++j) {
         printf("%c", 'A' + j); // Print characters starting from 'A'
      }
      printf("\n");
   }
   return 0;
}
```

Solution for ii):

1

2 3

4 5 6

```
#include <stdio.h>

int main() {
    int i, j, rows = 3, number = 1;

    for (i = 1; i <= rows; ++i) {
        for (j = 1; j <= i; ++j) {
            printf("%d ", number);
            ++number; // Increment the number to be printed
        }
        printf("\n");
    }
    return 0;
}
```

Practical - 7

Q-A: Write a program that defines a function to add first "n" numbers.

Answer:

```
#include <stdio.h>

// Function declaration (prototype)
int sumOfFirstN(int n);

int main() {
    int n, sum;

    // Read the value of n from user
    printf("Enter a positive integer n: ");
    scanf("%d", &n);

    // Call the function and store the result
    sum = sumOfFirstN(n);

    // Display the result
    printf("The sum of the first %d numbers is: %d\n", n, sum);

    return 0;
}

// Function definition
// This function calculates the sum of first n natural numbers
```

```
int sumOfFirstN(int n) {
    int i, sum = 0;
    for (i = 1; i <= n; i++) {
        sum += i; // Add each number to sum
    }
    return sum; // Return the calculated sum to main()
}
```

Q-B: Write a program using function to find maximum number from two numbers.

Answer:

```
#include <stdio.h>

// Function declaration
int findMax(int a, int b);

int main() {
    int num1, num2, max;

    // Read two numbers from user
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    // Call the function to find the maximum
    max = findMax(num1, num2);

    // Display the result
```

```
    printf("The maximum number between %d and %d is: %d\n", num1,
num2, max);

    return 0;
}

// Function definition to find the maximum of two numbers
int findMax(int a, int b) {
    if (a > b) {
        return a;
    } else {
        return b;
    }
}
```

Q-C: Write a function in the program to return 1 if number is prime otherwise return 0.

Answer:

```
#include <stdio.h>
#include <math.h> // For sqrt() function

// Function declaration
int isPrime(int num);

int main() {
    int num, result;
```

```
    // Read a number from user
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    // Call the function to check if the number is prime
    result = isPrime(num);

    // Display the result based on the return value
    if (result == 1) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}

// Function definition to check prime number
// Returns 1 if prime, 0 if not prime
int isPrime(int num) {
    int i;

    // Check for numbers less than 2
    if (num < 2) {
        return 0;
    }
```

```
    // Check for divisibility from 2 to sqrt(num)
    for (i = 2; i <= sqrt(num); i++) {
        if (num % i == 0) {
            return 0; // Not a prime number
        }
    }
    return 1; // Prime number
}
```

Practical - 8

Q-A: Write a program to exchange two numbers using function. (Using Call by Reference)

Answer:

```
#include <stdio.h>

// Function declaration using pointers
void swap(int *a, int *b);

int main() {
    int x, y;

    // Read two numbers from user
    printf("Enter value for x: ");
    scanf("%d", &x);
    printf("Enter value for y: ");
    scanf("%d", &y);

    printf("\nBefore swapping in main:\n");
    printf("x = %d, y = %d\n", x, y);

    // Call the swap function, passing the addresses of x and y
    swap(&x, &y);

    printf("\nAfter swapping in main:\n");
    printf("x = %d, y = %d\n", x, y);
```

```
    return 0;
}

// Function definition to swap two numbers
// This function uses pointers to modify the original variables
void swap(int *a, int *b) {
    int temp;
    temp = *a;  // Step 1: Store the value at address a in temp
    *a = *b;    // Step 2: Put the value at address b into address a
    *b = temp;  // Step 3: Put the value from temp into address b
}
```

Q-B: Write a program to reverse the number using function. (Using Call by Value)

Answer:

```
#include <stdio.h>

// Function declaration
int reverseNumber(int num);

int main() {
    int num, reversed;

    // Read a number from user
    printf("Enter an integer: ");
    scanf("%d", &num);
```

```
    // Call the function to reverse the number
    reversed = reverseNumber(num);

    // Display the result
    printf("The reverse of %d is %d.\n", num, reversed);

    return 0;
}

// Function definition to reverse a number
int reverseNumber(int num) {
    int reversed = 0, remainder;

    // Handle negative numbers by making them positive first
    int isNegative = 0;
    if (num < 0) {
        isNegative = 1;
        num = -num;
    }

    // Loop to reverse the digits
    while (num != 0) {
        remainder = num % 10;         // Get the last digit
        reversed = reversed * 10 + remainder; // Build the reversed number
        num = num / 10;               // Remove the last digit
    }
```

```
    // Convert back to negative if original number was negative
    if (isNegative) {
        reversed = -reversed;
    }

    return reversed;
}
```

Q-C: Write a program to generate first "n" number of Fibonacci series using recursion.

Answer:

```
#include <stdio.h>

// Recursive function declaration
int fibonacci(int n);

int main() {
    int n, i;

    // Read the value of n from user
    printf("Enter the number of terms: ");
    scanf("%d", &n);

    printf("Fibonacci Series: ");
    for (i = 0; i < n; i++) {
        printf("%d ", fibonacci(i)); // Call recursive function for each term
    }
```

```
    printf("\n");

    return 0;
}

// Recursive function to find the nth Fibonacci number
int fibonacci(int n) {
    // Base cases: fib(0) = 0, fib(1) = 1
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    } else {
        // Recursive case: fib(n) = fib(n-1) + fib(n-2)
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

Practical - 9

Q-A: Write a C program to find out the Maximum and Minimum number from given 10 numbers in an array.

Answer:

```
#include <stdio.h>

int main() {
    int arr[10]; // Step 1: Declare an array of size 10
    int i, max, min;

    // Step 2: Read 10 numbers from user and store in array
    printf("Enter 10 numbers:\n");
    for(i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }

    // Step 3: Initialize max and min with first element
    max = arr[0];
    min = arr[0];

    // Step 4: Traverse the array to find max and min
    for(i = 1; i < 10; i++) {
        if(arr[i] > max) {
            max = arr[i]; // Update max if current element is larger
        }
        if(arr[i] < min) {
```

```
            min = arr[i]; // Update min if current element is smaller
        }
    }

    // Step 5: Display the results
    printf("Maximum number: %d\n", max);
    printf("Minimum number: %d\n", min);

    return 0;
}
```

Q-B: Write a program to sort the elements of an array in ascending order. (Using Bubble Sort)

Answer:

```
#include <stdio.h>

int main() {
    int arr[10];
    int i, j, temp, n = 10;

    // Read 10 numbers
    printf("Enter 10 numbers:\n");
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Bubble Sort Algorithm
```

```
    for(i = 0; i < n-1; i++) { // Step 1: Outer loop for passes
        for(j = 0; j < n-i-1; j++) { // Step 2: Inner loop for comparisons
            if(arr[j] > arr[j+1]) { // Step 3: Compare adjacent elements
                // Step 4: Swap if they are in wrong order
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

    // Display sorted array
    printf("Sorted array in ascending order:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

Q-C: Write a C program to read 10 numbers from user and store them in an array. Display Sum, Minimum and Average of the numbers.

Answer:

```
#include <stdio.h>

int main() {
```

```
int arr[10];
int i, sum = 0, min;
float average;

// Read 10 numbers and calculate sum
printf("Enter 10 numbers:\n");
for(i = 0; i < 10; i++) {
    scanf("%d", &arr[i]);
    sum += arr[i]; // Calculate running total
}

// Find minimum
min = arr[0];
for(i = 1; i < 10; i++) {
    if(arr[i] < min) {
        min = arr[i];
    }
}

// Calculate average
average = (float)sum / 10; // Type casting for decimal result

// Display results
printf("Sum: %d\n", sum);
printf("Minimum: %d\n", min);
printf("Average: %.2f\n", average);
```

```
    return 0;
}
```

Practical - 10

Q-A: Write a program to access elements using pointer.

Answer:

```
#include <stdio.h>

int main() {
   int arr[5] = {10, 20, 30, 40, 50};
   int *ptr; // Pointer declaration
   int i;

   // Method 1: Pointer points to first element of array
   ptr = arr; // Equivalent to ptr = &arr[0]

   printf("Accessing array elements using pointer:\n");
   for(i = 0; i < 5; i++) {
      printf("Element %d: %d\n", i, *(ptr + i)); // Pointer arithmetic
   }

   // Method 2: Using pointer to traverse array
   printf("\nTraversing array using pointer increment:\n");
   ptr = arr; // Reset pointer to start of array
   for(i = 0; i < 5; i++) {
      printf("Element %d: %d\n", i, *ptr);
      ptr++; // Move pointer to next element
   }
```

```
    return 0;
}
```

Q-B: Design a structure student_record to contain name, branch and total marks obtained. Develop a program to read data for 10 students in a class and print them.

Answer:

```
#include <stdio.h>

// Define structure
struct student_record {
    char name[50];
    char branch[50];
    int total_marks;
};

int main() {
    struct student_record students[10]; // Array of structures
    int i;

    // Read data for 10 students
    printf("Enter details of 10 students:\n");
    for(i = 0; i < 10; i++) {
        printf("\nStudent %d:\n", i+1);
        printf("Enter name: ");
        scanf(" %[^\n]", students[i].name); // Space before % to skip newline
        printf("Enter branch: ");
        scanf(" %[^\n]", students[i].branch);
```

```
        printf("Enter total marks: ");
        scanf("%d", &students[i].total_marks);
    }

    // Display all student records
    printf("\nStudent Records:\n");
    printf("-----------------------------------------------\n");
    for(i = 0; i < 10; i++) {
        printf("Student %d:\n", i+1);
        printf("Name: %s\n", students[i].name);
        printf("Branch: %s\n", students[i].branch);
        printf("Total Marks: %d\n", students[i].total_marks);
        printf("-----------------------------------------------\n");
    }

    return 0;
}
```

Q-C: A file named data contains series of integer numbers. Write a c program to read all numbers from file and then write all odd numbers into file named "odd" and write all even numbers into file named "even". Display all the contents of these file on screen.

Answer:

```
#include <stdio.h>

int main() {
    FILE *fdata, *fodd, *feven;
    int number;
```

```
// Step 1: Open files
fdata = fopen("data.txt", "r");
fodd = fopen("odd.txt", "w");
feven = fopen("even.txt", "w");

// Check if files opened successfully
if(fdata == NULL || fodd == NULL || feven == NULL) {
    printf("Error opening files!\n");
    return 1;
}

// Step 2: Read from data.txt and separate odd/even numbers
while(fscanf(fdata, "%d", &number) != EOF) {
    if(number % 2 == 0) {
        fprintf(feven, "%d\n", number); // Write even numbers
    } else {
        fprintf(fodd, "%d\n", number); // Write odd numbers
    }
}

// Step 3: Close all files after writing
fclose(fdata);
fclose(fodd);
fclose(feven);

// Step 4: Display contents of odd.txt
```

```
    printf("Contents of odd.txt:\n");
    fodd = fopen("odd.txt", "r");
    while(fscanf(fodd, "%d", &number) != EOF) {
        printf("%d\n", number);
    }
    fclose(fodd);

    // Step 5: Display contents of even.txt
    printf("\nContents of even.txt:\n");
    feven = fopen("even.txt", "r");
    while(fscanf(feven, "%d", &number) != EOF) {
        printf("%d\n", number);
    }
    fclose(feven);

    return 0;
}
```

Q-D: Define a structure "personal" that would contain person name, date of joining and salary. Using this structure read information of 5 people and print the same on screen. Also display sum of salary of all 5 people.

Answer:

```
#include <stdio.h>

// Define structure
struct personal {
    char name[50];
```

```
    char doj[20]; // Date of joining as string for simplicity
    float salary;
};

int main() {
    struct personal people[5];
    int i;
    float total_salary = 0.0;

    // Read data for 5 people
    printf("Enter details of 5 people:\n");
    for(i = 0; i < 5; i++) {
        printf("\nPerson %d:\n", i+1);
        printf("Enter name: ");
        scanf(" %[^\n]", people[i].name);
        printf("Enter date of joining (dd/mm/yyyy): ");
        scanf(" %[^\n]", people[i].doj);
        printf("Enter salary: ");
        scanf("%f", &people[i].salary);

        total_salary += people[i].salary; // Calculate running total
    }

    // Display all records and total salary
    printf("\nPersonal Records:\n");
    printf("-------------------------------------------------\n");
    for(i = 0; i < 5; i++) {
```

```
        printf("Person %d:\n", i+1);
        printf("Name: %s\n", people[i].name);
        printf("Date of Joining: %s\n", people[i].doj);
        printf("Salary: %.2f\n", people[i].salary);
        printf("-----------------------------------------------\n");
    }

    printf("Total Salary of all 5 people: %.2f\n", total_salary);

    return 0;
}
```

Practical - 11

Q-A: (1) Original Code:

c

```
int main()
{
   int i=10;
   do {
      printf("%d",i);
      i++;
   } while(i<10);
   return 0;
}
```

Answer:

Output:

10

Error Analysis:

1. **Missing Header File:** The code uses printf but doesn't include stdio.h.
2. **Incorrect Data Type:** Int should be int (C is case-sensitive).
3. **Logic Error:** The loop condition while(i<10) is false in the first check since i is initialized to 10. However, in a do-while loop, the body executes at least once before the condition is checked.

Corrected Code:

```
#include <stdio.h>  // Added header file


int main() {        // Corrected 'Int' to 'int'
   int i = 10;
   do {
```

```
        printf("%d ", i); // Added space for better output formatting
        i++;
    } while(i < 10);
    return 0;
}
```

Step-by-Step Execution of Corrected Code:

1. **Initialization:** i = 10
2. **First Iteration (executes without condition check):**
    - printf("%d ", 10); prints "10"
    - i++ increments i to 11
3. **Condition Check:** while(11 < 10) → **False**
4. **Loop Terminates**
5. **Output:** 10

Q-B: Original Code:

```
intfunct1(intn);
int main()
{
   int n=10;
   printf("%d",funct1(n));
   return0;
}

intfunct1(intx)//function definition
{
   if(x>0)
   return x + funct1(x-1);
}
```

Answer:

Output:
This code will cause a **compilation error** and if compiled with warnings, may show **undefined behavior**.

Error Analysis:

1. **Missing Header File:** The code uses printf but doesn't include stdio.h.
2. **Syntax Errors:**
    - intfunct1(intn); → Missing space between int and funct1, and between int and n
    - return0; → Missing space between return and 0
    - intfunct1(intx) → Missing spaces in function definition
3. **Logical Error (Major):** The recursive function funct1 lacks a base case for when x <= 0. When x becomes 0, the function doesn't return any value, leading to undefined behavior.

Corrected Code:

```
#include <stdio.h>  // Added header file


int funct1(int n);  // Added spaces

int main()

{

   int n = 10;

   printf("%d", funct1(n));

   return 0;       // Added space

}


int funct1(int x)   // Added spaces

{

   if (x > 0) {
```

```
        return x + funct1(x - 1);
    } else {
        return 0;   // Added base case
    }
}
```

Step-by-Step Execution of Corrected Code:
(This calculates the sum of first n natural numbers)

Recursive Calls:

- funct1(10) = 10 + funct1(9)
- funct1(9) = 9 + funct1(8)
- funct1(8) = 8 + funct1(7)
- ...
- funct1(1) = 1 + funct1(0)
- funct1(0) = 0 **(Base case)**

Now the returns unwind:

- funct1(1) = 1 + 0 = 1
- funct1(2) = 2 + 1 = 3
- funct1(3) = 3 + 3 = 6
- ...
- funct1(10) = 10 + 45 = 55

Final Output: 55

Summary of Corrections:

1. Added #include <stdio.h>
2. Added proper spacing in function declarations and definitions
3. Added proper spacing in return 0;
4. Added base case return 0; for the recursive function
5. Added curly braces for better code structure