



Problem Solving with C

LABORATORY MANUAL

Week 3

Semester: 2

Course Code: UE24CS151B

Lab Anchors: Dr. Mohan Kumar AV

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Dr. Mohan Kumar AV, Prof. Rajeshwari CN

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none"> • All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Monday - Sections: P12, C12 (8.45AM-11.00AM)

1. Write a program that takes a student's score as input and prints their corresponding letter grade according to the following scale: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59).

Solution:

```
#include <stdio.h>
int main()
{
    int score;
    scanf("%d",&score);
    if(score>=90)
        printf("%c",'A');
    else if(score>=80)
        printf("%c",'B');
    else if(score>=70)
        printf("%c",'C');
    else if(score>=60)
        printf("%c",'D');
    else
        printf("%c",'F');

    return 0;
}
```

- 2. Write a program that prints the number of days in the given month as input for the year 2025. Input should be the month's number. For example, for "January," the input is 1 and program should display 31.**

Solution:

```
#include <stdio.h>

int main()
{
    int month;
    printf("Enter the month number (1-12): ");
    scanf("%d", &month);

    switch (month)
    {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            printf("31 days\n");
            break;
        case 4: case 6: case 9: case 11:
            printf("30 days\n");
            break;
        case 2:
            printf("28 days\n"); // 2025 is not a leap year
            break;
        default: printf("Invalid month number!\n");
    }

    return 0;
}
```

- 3. Write a C program to check whether a number entered by the user is even or odd using the if-else statement.**

Solution:

```
#include <stdio.h>

int main()
{
    int num;

    // Taking user input
    printf("Enter an integer: ");
    scanf("%d", &num);

    // Checking even or odd using if-else
    if (num % 2 == 0)
    {
        printf("%d is even.\n", num);
    }
    else
    {
        printf("%d is odd.\n", num);
    }
}
```

- 4. Competitive Challenge:** Sam has given a task of listing Leap years from 1000 AD to till date. Write a C program that takes a year as input and prints if it is a Leap Year or Not.

Solution:

```
#include<stdio.h>
int main()
{
    int year;
    scanf("%d", &year);

    if ((year%400 == 0) || ((year%4 == 0) && (year%100 != 0)))
        printf("Leap Year.");
    else
        printf("Not a Leap Year.");

    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none"> • All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Monday - Sections: P3, C3 (11:30AM-1:45PM)

1. Write a program that takes a student's score as input and prints their corresponding letter grade according to the following scale: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59).

Solution:

```
#include <stdio.h>
int main()
{
    int score;
    scanf("%d",&score);
    if(score>=90)
        printf("%c",'A');
    else if(score>=80)
        printf("%c",'B');
    else if(score>=70)
        printf("%c",'C');
    else if(score>=60)
        printf("%c",'D');
    else
        printf("%c",'F');

    return 0;
}
```

- 2. Write a program that prints the number of days in the given month as input for the year 2024. Input should be the month's number. For example, for "January," the input is 1 and program should display 31.**

Solution:

```
#include <stdio.h>
int main()
{
    int month;
    scanf("%d",&month);
    switch(month)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12: printf("%d",31);
                   break;

        case 2: printf("%d",29);
                  break;
        case 4:
        case 6:
        case 9:
        case 11:printf("%d",30);
                  break;

        default: printf("Invalid Entry");
                  break;
    }
    return 0;
}
```

-
- 3. Write a C program to check whether a number entered by the user is even or odd using the if-else statement.**

Solution:

```
#include <stdio.h>

int main()
{
    int num;

    // Taking user input
    printf("Enter an integer: ");
    scanf("%d", &num);

    // Checking even or odd using if-else
    if (num % 2 == 0) {
        printf("%d is even.\n", num);
    } else {
        printf("%d is odd.\n", num);
    }

    return 0;
}
```

- 4. Competitive Challenge: A digital lock allows entry only if a 4-digit passcode meets these criteria:**

- a. **The sum of the digits is even.**
- b. **The first and last digits are equal.**
- c. **Write a C program that reads a 4-digit integer and checks if it meets the criteria.**

Solution:

```
#include <stdio.h>

int main() {
    int num, d1, d2, d3, d4, sum;

    printf("Enter a 4-digit number: ");
    scanf("%d", &num);

    if (num < 1000 || num > 9999) {
        printf("Invalid Input! Enter a 4-digit number.\n");
        return 1;
    }

    d1 = num / 1000;
    d2 = (num / 100) % 10;
    d3 = (num / 10) % 10;
    d4 = num % 10;
    sum = d1 + d2 + d3 + d4;

    if (sum % 2 == 0 && d1 == d4) {
        printf("Access Granted\n");
    } else {
        printf("Access Denied\n");
    }

    return 0;
}
```

To learn and solve	Programs on Control Structures <ul style="list-style-type: none"> • All Program are mandatory questions and should be submitted for evaluation.
--------------------	--

(Students should submit their codes through drive link)

Monday - Sections: P2, C2 (2:30PM-4:45PM)

1. Write a program that takes a student's score as input and prints their corresponding letter grade according to the following scale: A (90-100), B (80-89), C (70-79), D (60-69), F (0-59).

Solution:

```
#include <stdio.h>
int main()
{
    int score;
    scanf("%d",&score);
    if(score>=90)
        printf("%c",'A');
    else if(score>=80)
        printf("%c",'B');
    else if(score>=70)
        printf("%c",'C');
    else if(score>=60)
        printf("%c",'D');
    else
        printf("%c",'F');

    return 0;
}
```

- 2. Write a program that prints the number of days in the given month as input for the year 2025. Input should be the month's number. For example, for "January," the input is 1 and program should display 31.**

Solution:

```
#include <stdio.h>

int main()
{
    int month;
    printf("Enter the month number (1-12): ");
    scanf("%d", &month);

    switch (month)
    {
        case 1: case 3: case 5: case 7: case 8: case 10: case 12:
            printf("31 days\n");
            break;
        case 4: case 6: case 9: case 11:
            printf("30 days\n");
            break;
        case 2:
            printf("28 days\n"); // 2025 is not a leap year
            break;
        default: printf("Invalid month number!\n");
    }

    return 0;
}
```

-
- 3. Write a C program to check whether a number entered by the user is even or odd using the if-else statement.**

Solution:

```
#include <stdio.h>

int main() {
    int num;

    // Taking user input
    printf("Enter an integer: ");
    scanf("%d", &num);

    // Checking even or odd using if-else
    if (num % 2 == 0) {
        printf("%d is even.\n", num);
    } else {
        printf("%d is odd.\n", num);
    }

    return 0;
}
```

4. Competitive Challenge: Simulate a traffic light system where:

- a. "Red" lasts for 30 seconds.
- b. "Yellow" lasts for 5 seconds.
- c. "Green" lasts for 25 seconds.

Given the elapsed time (0-59 seconds), display the current light color.

Solution:

```
#include <stdio.h>

int main() {
    int time;
    printf("Enter elapsed time (0-59): ");
    scanf("%d", &time);

    if (time < 0 || time > 59) {
        printf("Invalid input! Enter time between 0-59.\n");
        return 1;
    }

    if (time < 30) {
        printf("Red Light\n");
    } else if (time < 35) {
        printf("Yellow Light\n");
    } else {
        printf("Green Light\n");
    }

    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none"> • All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Tuesday - Sections: P6, C6 (8.45AM-11.00AM)

1. Write a program that checks whether a given character is a vowel or a consonant using switch-case.

Solution:

```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter an alphabet: ");
    scanf(" %c", &ch);
    switch(ch) {
        case 'a': case 'e': case 'i': case 'o': case 'u':
        case 'A': case 'E': case 'I': case 'O': case 'U':
            printf("%c is a vowel.\n", ch);
            break;
        default:
            printf("%c is a consonant.\n", ch);
    }
    return 0;
}
```

2. Write a program to calculate the factorial of a given number using a loop.

Use if to check for negative input.

Solution:

```
#include <stdio.h>
int main() {
    int n;
    unsigned long long factorial = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    if(n < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        for(int i = 1; i <= n; i++){
            factorial *= i;
        }
        printf("Factorial of %d is %llu.\n", n, factorial);
    }
    return 0;
}
```

3. Write a program to check whether a given number is prime or not using loops and control structures.

Solution:

```
#include <stdio.h>
int main() {
    int num, flag = 1; // Assume the number is prime initially.

    printf("Enter an integer: ");
    scanf("%d", &num);

    // Numbers less than 2 are not prime.
    if (num < 2) {
        flag = 0;
    } else {
        // Check divisibility from 2 to num/2.
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                flag = 0; // Found a divisor, so number is not prime.
                break;
            }
        }
    }
}
```

4. **Competitive Challenge:** Write a program that repeatedly sums the digits of a number until a single-digit number (digital root) is obtained. Use nested loops and control structures.

Solution:

```
#include <stdio.h>
int main() {
    int num, sum, digit;
    printf("Enter a positive integer: ");
    scanf("%d", &num);

    while(num >= 10) { // Continue until num becomes a single digit
        sum = 0;
        while(num > 0) {
            digit = num % 10;
            sum += digit;
            num /= 10;
        }
        num = sum;
    }
    printf("The digital root is: %d\n", num);
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none"> • All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Tuesday - Sections: P1, C1 (11.30AM-1.45PM)

1. Write a program that checks whether a given character is a vowel or a consonant using switch-case.

Solution:

```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter an alphabet: ");
    scanf(" %c", &ch);
    switch(ch) {
        case 'a': case 'e': case 'i': case 'o': case 'u':
        case 'A': case 'E': case 'I': case 'O': case 'U':
            printf("%c is a vowel.\n", ch);
            break;
        default:
            printf("%c is a consonant.\n", ch);
    }
    return 0;
}
```

2. Write a program to calculate the factorial of a given number using a loop.

Use if to check for negative input.

Solution:

```
#include <stdio.h>
int main() {
    int n;
    unsigned long long factorial = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    if(n < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        for(int i = 1; i <= n; i++){
            factorial *= i;
        }
        printf("Factorial of %d is %llu.\n", n, factorial);
    }
    return 0;
}
```

3. Write a program to check whether a given number is prime or not using loops and control structures.

Solution:

```
#include <stdio.h>
int main() {
    int num, flag = 1; // Assume the number is prime initially.

    printf("Enter an integer: ");
    scanf("%d", &num);

    // Numbers less than 2 are not prime.
    if (num < 2) {
        flag = 0;
    } else {
        // Check divisibility from 2 to num/2.
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                flag = 0; // Found a divisor, so number is not prime.
                break;
            }
        }
    }
}
```

4. **Competitive Challenge:** Write a program to compute both the GCD and LCM of two numbers. Use the Euclidean algorithm for GCD and the relation: $\text{LCM} \times \text{GCD} = \text{product of the numbers}$.

Solution:

```
#include <stdio.h>
int main() {
    int a, b, num1, num2, gcd, lcm, temp;
    printf("Enter two positive integers: ");
    scanf("%d %d", &a, &b);
    num1 = a;
    num2 = b;

    // Euclidean algorithm for GCD
    while(b != 0) {
        temp = b;
        b = a % b;
        a = temp;
    }
    gcd = a;
    lcm = (num1 * num2) / gcd;

    printf("GCD: %d\n", gcd);
    printf("LCM: %d\n", lcm);
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Tuesday - Sections: P4, C4 (2.30PM-4.00PM)

1. Write a program that checks whether a given character is a vowel or a consonant using switch-case.

Solution:

```
#include <stdio.h>
int main()
{
    char ch;
    printf("Enter an alphabet: ");
    scanf(" %c", &ch);
    switch(ch) {
        case 'a': case 'e': case 'i': case 'o': case 'u':
        case 'A': case 'E': case 'I': case 'O': case 'U':
            printf("%c is a vowel.\n", ch);
            break;
        default:
            printf("%c is a consonant.\n", ch);
    }
    return 0;
}
```

2. Write a program to calculate the factorial of a given number using a loop.

Use if to check for negative input.

Solution:

```
#include <stdio.h>
int main() {
    int n;
    unsigned long long factorial = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    if(n < 0) {
        printf("Factorial is not defined for negative numbers.\n");
    } else {
        for(int i = 1; i <= n; i++){
            factorial *= i;
        }
        printf("Factorial of %d is %llu.\n", n, factorial);
    }
    return 0;
}
```

3. Write a program to check whether a given number is prime or not using loops and control structures.

Solution:

```
#include <stdio.h>
int main() {
    int num, flag = 1; // Assume the number is prime initially.

    printf("Enter an integer: ");
    scanf("%d", &num);

    // Numbers less than 2 are not prime.
    if (num < 2) {
        flag = 0;
    } else {
        // Check divisibility from 2 to num/2.
        for (int i = 2; i <= num / 2; i++) {
            if (num % i == 0) {
                flag = 0; // Found a divisor, so number is not prime.
                break;
            }
        }
    }
}
```

4. **Competitive Challenge:** Write a program to calculate the electricity bill from number of units based on following criteria: First 10 units: No Charge Next 15 units: Rs. 10/unit Above 25 units: Rs. 25/unit

Solution:

```
#include<stdio.h>
int main()
{
    int units;
    scanf("%d",&units);
    if(units<=10)
        printf("No Charge\n");
    else if(units<=25)
        printf("%f",(float)(units-10)*10);
    else
        printf("%f",(float)((15*10) + (units-25)*25));
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Wednesday - Sections: P13 (8.45AM-11.00AM)

1. Write a program that prints the multiplication table of a given number using a loop.

Solution:

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    for(int i = 1; i <= 10; i++){
        printf("%d x %d = %d\n", num, i, num * i);
    }
    return 0;
}
```

- 2. Write a program to calculate the sum of the first n natural numbers using a while loop.**

Solution:

```
#include <stdio.h>
int main() {
    int n, sum = 0, i = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    while(i <= n) {
        sum += i;
        i++;
    }
    printf("Sum of the first %d natural numbers is %d.\n", n, sum);
    return 0;
}
```

-
- 3. Write a program to find the Greatest Common Divisor (GCD) of two numbers using a while loop and control structures.**

Solution:

```
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter two positive integers: ");
    scanf("%d %d", &a, &b);

    while(b != 0) {
        temp = b;
        b = a % b;
        a = temp;
    }

    printf("GCD is %d.\n", a);
    return 0;
}
```

- 4. Competitive Challenge:** Write a C program that prints all numbers from 1 to n that are multiples of 2, without using the modulus (%) operator.

Solution:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter a number: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        // Check if the number is a multiple of 2 by subtracting 2 repeatedly
        int j = i;
        while (j > 2) {
            j -= 2; // Decrease by 2 until it becomes less than or equal to 2
        }
        if (j == 0) {
            printf("%d ", i); // Print if it's a multiple of 2
        }
    }
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Wednesday - Sections: P13 (11.30AM-1.45PM)

1. Write a program that prints the multiplication table of a given number using a loop.

Solution:

```
#include <stdio.h>
int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    for(int i = 1; i <= 10; i++){
        printf("%d x %d = %d\n", num, i, num * i);
    }
    return 0;
}
```

- 2. Write a program to calculate the sum of the first n natural numbers using a while loop.**

Solution:

```
#include <stdio.h>
int main() {
    int n, sum = 0, i = 1;
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    while(i <= n) {
        sum += i;
        i++;
    }
    printf("Sum of the first %d natural numbers is %d.\n", n, sum);
    return 0;
}
```

-
- 3. Write a program to find the Greatest Common Divisor (GCD) of two numbers using a while loop and control structures.**

Solution:

```
#include <stdio.h>
int main() {
    int a, b, temp;
    printf("Enter two positive integers: ");
    scanf("%d %d", &a, &b);

    while(b != 0) {
        temp = b;
        b = a % b;
        a = temp;
    }

    printf("GCD is %d.\n", a);
    return 0;
}
```

- 4. Competitive Challenge:** Write a program that prints all the numbers from 1 to n that are either multiples of 2 or multiples of 3

Solution:

```
#include<stdio.h>
int main()
{
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
        if(i%2==0 || i%3==0)
            printf("%d ",i);
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Thursday - Sections: P9, C9 (8.45AM-11.30AM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

- 1. Write a program to print the count of even numbers in a given range. The lower and upper value in the range will be given as input.**

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==0)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Write a program in C to calculate the electricity bill for a household based on the units consumed. The rate of electricity per unit varies as follows: For the first 100 units, the rate is Rs. 2.50 per unit. For the next 200 units, the rate is Rs. 4.00 per unit. For any additional units, the rate is Rs. 6.00 per unit. The program should read the number of units consumed. Then, calculate and display the total bill amount.**

Solution:

```
#include<stdio.h>
int main()
{
    int units;
    float bill = 0.0;
    scanf("%d", &units);
    if(units <= 100)
        bill = units * 2.50;
    else if (units <= 300)
        bill = 100 * 2.50 + (units - 100) * 4.00;
    else
        bill = 100 * 2.50 + 200 * 4.00 + (units - 300) * 6.00;

    printf("Rs.%2f", bill);
    return 0;
}
```

4. **Competitive Challenge:** A strong number is one in which the sum of the factorials of its digits equals the number itself (e.g., $145 = 1! + 4! + 5!$). Write a program to check if a given number is a strong number using nested loops.

Solution:

```
#include <stdio.h>
int main() {
    int num, original, digit, sum = 0, factorial;
    printf("Enter a positive integer: ");
    scanf("%d", &num);
    original = num;

    while(num > 0) {
        digit = num % 10;
        // Calculate factorial of the digit
        factorial = 1;
        for(int i = 1; i <= digit; i++){
            factorial *= i;
        }
        sum += factorial;
        num /= 10;
    }

    if(sum == original)
        printf("%d is a strong number.\n", original);
    else
        printf("%d is not a strong number.\n", original);

    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Thursday - Sections: P9, C9 (8.45AM-11.30AM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

- 1. Write a program to print the count of even numbers in a given range. The lower and upper value in the range will be given as input.**

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==0)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Write a program in C to calculate the electricity bill for a household based on the units consumed. The rate of electricity per unit varies as follows: For the first 100 units, the rate is Rs. 2.50 per unit. For the next 200 units, the rate is Rs. 4.00 per unit. For any additional units, the rate is Rs. 6.00 per unit. The program should read the number of units consumed. Then, calculate and display the total bill amount.**

Solution:

```
#include<stdio.h>
int main()
{
    int units;
    float bill = 0.0;
    scanf("%d", &units);
    if(units <= 100)
        bill = units * 2.50;
    else if (units <= 300)
        bill = 100 * 2.50 + (units - 100) * 4.00;
    else
        bill = 100 * 2.50 + 200 * 4.00 + (units - 300) * 6.00;

    printf("Rs.%2f", bill);
    return 0;
}
```

-
- 4. Competitive Challenge: Write a program to print a pyramid pattern of stars (*) for a given number of rows using nested loops.**

Solution:

```
#include <stdio.h>
int main() {
    int rows;
    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    for(int i = 1; i <= rows; i++){
        // Print leading spaces
        for(int j = 1; j <= rows - i; j++){
            printf(" ");
        }
        // Print stars
        for(int k = 1; k <= (2 * i - 1); k++){
            printf("*");
        }
        printf("\n");
    }
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Thursday - Sections: P9, C9 (8.45AM-11.30AM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

2. Write a program to print the count of even numbers in a given range.

The lower and upper value in the range will be given as input.

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==0)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Write a program in C to calculate the electricity bill for a household based on the units consumed. The rate of electricity per unit varies as follows: For the first 100 units, the rate is Rs. 2.50 per unit. For the next 200 units, the rate is Rs. 4.00 per unit. For any additional units, the rate is Rs. 6.00 per unit. The program should read the number of units consumed. Then, calculate and display the total bill amount.**

Solution:

```
#include<stdio.h>
int main()
{
    int units;
    float bill = 0.0;
    scanf("%d", &units);
    if(units <= 100)
        bill = units * 2.50;
    else if (units <= 300)
        bill = 100 * 2.50 + (units - 100) * 4.00;
    else
        bill = 100 * 2.50 + 200 * 4.00 + (units - 300) * 6.00;

    printf("Rs.%2f", bill);
    return 0;
}
```

- 4. Competitive Challenge:** An organization wants to compute the final salary for its employees based on their base salary, years of service, and performance rating. The bonus scheme is as follows:

Performance Bonus:

Rating A: 20% of base salary

Rating B: 10% of base salary

Rating C: 5% of base salary

Any other rating: 0% bonus

Loyalty Bonus:

If an employee has more than 5 years of service, they receive an additional 5% of the base salary as a loyalty bonus.

Write a program in C that:

Accepts the base salary (as a floating-point number), years of service (integer), and performance rating (character) from the user.

Calculates the performance bonus and loyalty bonus (if applicable).

Outputs the base salary, performance bonus, loyalty bonus, and the final salary (base salary + bonuses).

Solution:

```
#include <stdio.h>

int main() {
    float baseSalary, performanceBonus = 0.0, loyaltyBonus = 0.0, finalSalary;
    int yearsOfService;
    char rating;

    // Input section
    printf("Enter the base salary: ");
    scanf("%f", &baseSalary);

    printf("Enter the number of years of service: ");
    scanf("%d", &yearsOfService);

    printf("Enter the performance rating (A, B, C, or other): ");
    scanf(" %c", &rating); // Note the space before %c to consume any whitespace

    // Calculate performance bonus based on rating
    if (rating == 'A' || rating == 'a') {
        performanceBonus = baseSalary * 0.20;
    } else if (rating == 'B' || rating == 'b') {
        performanceBonus = baseSalary * 0.10;
    } else if (rating == 'C' || rating == 'c') {
        performanceBonus = baseSalary * 0.05;
    } else {
        performanceBonus = 0.0;
    }

    // Calculate loyalty bonus if years of service > 5
    if (yearsOfService > 5) {
        loyaltyBonus = baseSalary * 0.05;
    }

    // Calculate final salary
    finalSalary = baseSalary + performanceBonus + loyaltyBonus;

    // Output results
    printf("\n--- Salary Breakdown ---\n");
    printf("Base Salary : %.2f\n", baseSalary);
    printf("Performance Bonus : %.2f\n", performanceBonus);
    printf("Loyalty Bonus : %.2f\n", loyaltyBonus);
    printf("Final Salary : %.2f\n", finalSalary);

    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Friday - Sections: P10, C10 (8.45AM-11.30AM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

2. Write a program to print the count of odd numbers in a given range.

The lower and upper value in the range will be given as input.

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==1)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Design a program that calculates the area of a geometric shape based on the user's choice. The program should display a menu with three options:**

- **Circle**
- **Rectangle**
- **Triangle**

Depending on the user's choice.

Solution:

```
#include <stdio.h>
int main() {
    int choice;
    float area, radius, length, width, base, height;

    printf("Area Calculator for Geometric Shapes\n");
    printf("-----\n");
    printf("Choose a shape:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("3. Triangle\n");
    printf("Enter your choice (1-3): ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Enter the radius of the circle: ");
            scanf("%f", &radius);
            area = 3.14 * radius * radius;
            printf("Area of the circle = %.2f\n", area);
            break;
        case 2:
            printf("Enter the length and width of the rectangle: ");
            scanf("%f %f", &length, &width);
            area = length * width;
            printf("Area of the rectangle = %.2f\n", area);
            break;
        case 3:
            printf("Enter the base and height of the triangle: ");
            scanf("%f %f", &base, &height);
            area = 0.5 * base * height;
            printf("Area of the triangle = %.2f\n", area);
            break;
        default:
            printf("Invalid choice. Please run the program again and select a
number between 1 and 3.\n");
    }
}
```

4. Competitive Challenge: A bank wants to implement a basic ATM system where a user can withdraw money. The program should:

1. Ask the user to enter their account balance.

2. Ask the user to enter the withdrawal amount.

3. Check the following conditions using control structures:

- If the withdrawal amount is greater than the account balance, display "Insufficient balance!"**
- If the withdrawal amount is 0 or negative, display "Invalid amount entered!"**
- Otherwise, deduct the amount and display the remaining balance.**

Constraints:

Use if-else statements to check conditions.

Solution:

```
#include <stdio.h>

int main() {
    float balance, withdrawAmount;

    // Input section
    printf("Enter your account balance: ");
    scanf("%f", &balance);

    printf("Enter withdrawal amount: ");
    scanf("%f", &withdrawAmount);

    // Checking conditions using if-else
    if (withdrawAmount > balance) {
        printf("Insufficient balance!\n");
    } else if (withdrawAmount <= 0) {
        printf("Invalid amount entered!\n");
    } else {
        balance -= withdrawAmount; // Deducting the withdrawal amount
        printf("Withdrawal successful!\n");
        printf("Remaining balance: %.2f\n", balance);
    }
    return 0;
}
```

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Friday - Sections: P11, C11 (11.30AM-1.45PM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

2. Write a program to print the count of odd numbers in a given range.

The lower and upper value in the range will be given as input.

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==1)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Design a program that calculates the area of a geometric shape based on the user's choice. The program should display a menu with three options:**

- **Circle**
- **Rectangle**
- **Triangle**

Depending on the user's choice.

```
#include <stdio.h>
int main() {
    int choice;
    float area, radius, length, width, base, height;

    printf("Area Calculator for Geometric Shapes\n");
    printf("-----\n");
    printf("Choose a shape:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("3. Triangle\n");
    printf("Enter your choice (1-3): ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Enter the radius of the circle: ");
            scanf("%f", &radius);
            area = 3.14 * radius * radius;
            printf("Area of the circle = %.2f\n", area);
            break;
        case 2:
            printf("Enter the length and width of the rectangle: ");
            scanf("%f %f", &length, &width);
            area = length * width;
            printf("Area of the rectangle = %.2f\n", area);
            break;
        case 3:
            printf("Enter the base and height of the triangle: ");
            scanf("%f %f", &base, &height);
            area = 0.5 * base * height;
            printf("Area of the triangle = %.2f\n", area);
            break;
        default:
            printf("Invalid choice. Please run the program again and select a
number between 1 and 3.\n");
    }
}
```

Solution:

- 4. Competitive Challenge: A store offers discounts based on the total purchase amount. Write a C program that:**

Asks the user to enter the total purchase amount.

Calculates the discount based on the following conditions:

If the purchase amount is below ₹500, no discount is applied.

If the purchase amount is between ₹500 and ₹1000, a 10% discount is applied.

If the purchase amount is above ₹1000, a 20% discount is applied.

Displays the original price, discount applied, and final price after discount.

If the entered amount is negative or zero, print "Invalid amount entered!".

```
#include <stdio.h>

int main() {
    float amount, discount = 0.0, finalPrice;

    // Input section
    printf("Enter total purchase amount: ");
    scanf("%f", &amount);

    // Validate input
    if (amount <= 0) {
        printf("Invalid amount entered!\n");
    } else {
        // Calculate discount
        if (amount >= 500 && amount <= 1000) {
            discount = amount * 0.10; // 10% discount
        } else if (amount > 1000) {
            discount = amount * 0.20; // 20% discount
        }

        // Calculate final price
        finalPrice = amount - discount;

        // Output results
        if (discount == 0) {
            printf("No discount applied.\n");
        } else {
            printf("Discount applied: ₹%.2f\n", discount);
        }
        printf("Final price: ₹%.2f\n", finalPrice);
    }
}
```

Solution:

To learn and solve	Programs on Control Structures
	<ul style="list-style-type: none">• All Program are mandatory questions and should be submitted for evaluation.

(Students should submit their codes through drive link)

Friday - Sections: P8, C8 (2.30PM-4.45PM)

1. Write a program to print the count of digits in a given positive number.

Solution:

```
#include<stdio.h>
int main()
{
    int n,count;
    scanf("%d",&n);
    count = 0;
    while(n>0)
    {
        count++;
        n = n/10;
    }
    printf("Count=%d",count);
    return 0;
}
```

2. Write a program to print the count of odd numbers in a given range.

The lower and upper value in the range will be given as input.

Solution:

```
#include<stdio.h>
int main()
{
    int lower,upper,count=0;
    scanf("%d%d",&lower,&upper);
    for(int i=lower;i<=upper;i++)
        if(i%2==1)
            count+=1;
    printf("%d",count);
    return 0;
}
```

- 3. Design a program that calculates the area of a geometric shape based on the user's choice. The program should display a menu with three options:**

- **Circle**
- **Rectangle**
- **Triangle**

Depending on the user's choice.

Solution:

```
#include <stdio.h>
int main() {
    int choice;
    float area, radius, length, width, base, height;

    printf("Area Calculator for Geometric Shapes\n");
    printf("-----\n");
    printf("Choose a shape:\n");
    printf("1. Circle\n");
    printf("2. Rectangle\n");
    printf("3. Triangle\n");
    printf("Enter your choice (1-3): ");
    scanf("%d", &choice);

    switch(choice) {
        case 1:
            printf("Enter the radius of the circle: ");
            scanf("%f", &radius);
            area = 3.14 * radius * radius;
            printf("Area of the circle = %.2f\n", area);
            break;
        case 2:
            printf("Enter the length and width of the rectangle: ");
            scanf("%f %f", &length, &width);
            area = length * width;
            printf("Area of the rectangle = %.2f\n", area);
            break;
        case 3:
            printf("Enter the base and height of the triangle: ");
            scanf("%f %f", &base, &height);
            area = 0.5 * base * height;
            printf("Area of the triangle = %.2f\n", area);
            break;
        default:
            printf("Invalid choice. Please run the program again and select a
number between 1 and 3.\n");
    }
}
```

Competitive Challenge: A government wants to implement an income tax calculator for citizens based on their annual salary. Write a C program that:

- Asks the user to enter their annual income.
- Calculates the income tax based on the following conditions:
 - Income \leq ₹2,50,000: No tax
 - ₹2,50,001 - ₹5,00,000: 5% tax on the amount exceeding ₹2,50,000
 - ₹5,00,001 - ₹10,00,000: 20% tax on the amount exceeding ₹5,00,000, plus 5% tax on the previous ₹2,50,000
 - Above ₹10,00,000: 30% tax on the amount exceeding ₹10,00,000, plus 20% tax on the previous ₹5,00,000 and 5% tax on ₹2,50,000

Displays the tax amount and net income after tax.

If the income entered is negative, print "Invalid income entered!".

Solution:

```
#include <stdio.h>

int main() {
    float income, tax = 0.0, netIncome;

    // Input section
    printf("Enter annual income: ");
    scanf("%f", &income);

    // Validate input
    if (income < 0) {
        printf("Invalid income entered!\n");
    } else {
        // Calculate tax based on income slabs
        if (income > 1000000) {
            tax += (income - 1000000) * 0.30;
            income = 1000000;
        }
        if (income > 500000) {
            tax += (income - 500000) * 0.20;
            income = 500000;
        }
        if (income > 250000) {
            tax += (income - 250000) * 0.05;
        }

        // Calculate net income after tax
        netIncome = income - tax;

        // Display output
        if (tax == 0) {
            printf("No tax.\n");
        } else {
            printf("Income tax: ₹%.2f\n", tax);
        }
        printf("Net income after tax: ₹%.2f\n", netIncome);
    }

    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

- 1. Write a C program that takes two numbers as input from the user, calculates their sum and average, and then displays the results. The program should:**
- 2. Prompt the user to enter two numbers.**
- 3. Compute the sum of the two numbers.**
- 4. Compute the average of the two numbers.**
- 5. Display the sum and average with appropriate formatting.**

Solution:

```
#include <stdio.h>

int main() {
    float num1, num2, sum, average;

    printf("Enter two numbers: ");
    scanf("%f %f", &num1, &num2);

    sum = num1 + num2;
    average = sum / 2;

    printf("Sum = %.2f\n", sum);
    printf("Average = %.2f\n", average);

    return 0;
}
```

- 1) Write a C program that takes two numbers as input from the user and calculates the absolute difference between them. The program should:**

- 1. Prompt the user to enter two numbers.**
- 2. Compute the absolute difference between the two numbers.**
- 3. Display the absolute difference.**

Solution:

```
#include <stdio.h>
#include <stdlib.h> // For abs() function

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

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

    difference = abs(num1 - num2);

    printf("Absolute Difference = %d\n", difference);

    return 0;
}
```

- 2) Write a C program that converts a given distance in kilometers to meters and centimeters. The program should:**

- 1. Prompt the user to enter the distance in kilometers.**
- 2. Convert the distance into meters and centimeters.**
- 3. Display the results with appropriate units.**

Solution:

```
#include <stdio.h>

int main() {

    float kilometers, meters, centimeters;

    printf("Enter distance in kilometers: ");
    scanf("%f", &kilometers);

    meters = kilometers * 1000;
    centimeters = kilometers * 100000;

    printf("Distance in meters: %.2f m\n", meters);
    printf("Distance in centimeters: %.2f cm\n", centimeters);

    return 0;
}
```

- 3) Write a C program that takes a number as input from the user and calculates its square and cube. The program should:**

- 1. Prompt the user to enter a number.**
- 2. Compute the square and cube of the given number.**
- 3. Display the results.**

Solution:

```
#include <stdio.h>

int main() {
    int num, square, cube;

    printf("Enter a number: ");
    scanf("%d", &num);

    square = num * num;
    cube = num * num * num;

    printf("Square of %d = %d\n", num, square);
    printf("Cube of %d = %d\n", num, cube);

    return 0;
}
```

4) Write a C program that calculates the volume of a cylinder using the formula:

$$V = \pi r^2 h$$

where:

- **r** is the radius of the base,
- **h** is the height,
- **π** is approximately **3.14159**.

The program should:

1. Prompt the user to enter the radius and height of the cylinder.
2. Compute the volume using the given formula.
3. Display the result with appropriate formatting.

Solution:

```
#include <stdio.h>
#define PI 3.14159 // Define the value of π

int main() {
    float radius, height, volume;

    printf("Enter the radius of the cylinder: ");
    scanf("%f", &radius);
    printf("Enter the height of the cylinder: ");
    scanf("%f", &height);

    volume = PI * radius * radius * height;

    printf("Volume of the cylinder = %.2f cubic units\n", volume);

    return 0;
}
```

5) Write a C program that takes two integers as input from the user: a dividend and a divisor (where the divisor is not zero). The program should:

- 1. Prompt the user to enter the dividend and divisor.**
- 2. Compute the quotient and remainder using integer division.**
- 3. Display the results.**

Hint:

- Quotient using integer division (/).**
- Remainder using the modulus operator (%).**

Solution:

```
#include <stdio.h>

int main() {
    int dividend, divisor, quotient, remainder;

    printf("Enter dividend: ");
    scanf("%d", &dividend);

    printf("Enter divisor (non-zero): ");
    scanf("%d", &divisor);

    quotient = dividend / divisor;
    remainder = dividend % divisor;

    printf("Quotient = %d\n", quotient);
    printf("Remainder = %d\n", remainder);

    return 0;
}
```

6) Write a C program that takes an integer input from the user and finds its last digit. The program should:

- 1. Prompt the user to enter a number.**
- 2. Compute the last digit using the modulus (%) operator.**
- 3. Display the last digit as output.**

Solution:

```
#include <stdio.h>

int main() {
    int num, last_digit;

    printf("Enter a number: ");
    scanf("%d", &num);

    last_digit = num % 10;

    printf("The last digit of %d is %d\n", num, last_digit);

    return 0;
}
```

7) Write a C program to calculate simple interest using the formula:

Where:

- **P = Principal amount**
- **R = Annual interest rate (in percentage)**
- **T = Time in years**

$$\text{Simple Interest} = \frac{P \times R \times T}{100}$$

The program should:

1. Take user input for principal, rate, and time.
2. Compute the simple interest using the given formula.
3. Display the simple interest as output.

Solution:

```
#include <stdio.h>

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

    printf("Enter principal amount: ");
    scanf("%lf", &principal);
    printf("Enter annual interest rate (in percentage): ");
    scanf("%lf", &rate);
    printf("Enter time (in years): ");
    scanf("%lf", &time);

    // Calculate simple interest using the formula SI = (P * R * T) / 100
    simple_interest = (principal * rate * time) / 100;

    printf("Simple Interest = %.2lf\n", simple_interest);

    return 0;
}
```

- 8) A scientist is studying a bacteria colony that doubles every hour. Given the initial number of bacteria and the number of hours, write a C program to calculate the total bacteria count after the given hours using the pow() function.**

Solution:

$$\text{Total Bacteria} = \text{Initial Count} \times 2^{\text{hours}}$$

```
#include <stdio.h>
#include <math.h> // Required for pow()

int main() {
    int initial_bacteria, hours;
    double total_bacteria;

    printf("Enter the initial bacteria count: ");
    scanf("%d", &initial_bacteria);
    printf("Enter the number of hours: ");
    scanf("%d", &hours);

    // Calculating total bacteria using pow()
    total_bacteria = initial_bacteria * pow(2, hours);

    printf("After %d hours, the total bacteria count is: %.0lf\n", hours, total_bacteria);

    return 0;
}
```

9) Write a C program that calculates compound interest using the formula:

$$A = P \times \left(1 + \frac{r}{n}\right)^{(n \times t)}$$

Where:

- **P = Principal amount**
- **r = Annual interest rate (in decimal form)**
- **t = Time in years**
- **n = Number of times interest is compounded per year**
- **A = Final amount after interest is applied**

The compound interest earned is the final amount minus the initial principal:

$$\text{Compound Interest} = A - P$$

The program should:

1. **Prompt the user to enter the principal amount, interest rate, time in years, and compounding frequency (times per year).**
2. **Compute the compound interest using the given formula.**
3. **Display the compound interest as output.**

Solution:

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

int main() {
    double principal, rate, time, compound_interest;
    int n;

    printf("Enter principal amount: ");
    scanf("%lf", &principal);
    printf("Enter annual interest rate (in percentage): ");
    scanf("%lf", &rate);
    printf("Enter time (in years): ");
    scanf("%lf", &time);
    printf("Enter number of times interest is compounded per year: ");
    scanf("%d", &n);

    // Convert rate from percentage to decimal
    rate = rate / 100;

    // Calculate compound interest using the formula A = P * (1 + r/n)^(n*t)
    compound_interest = principal * pow((1 + rate / n), (n * time)) - principal;

    printf("Compound Interest = %.2lf\n", compound_interest);

    return 0;
}
```

- 10) Write a C program that converts a given temperature from Celsius to Fahrenheit using the formula:**

The program should:

1. Prompt the user to enter the temperature in Celsius.
2. Convert the Celsius temperature to Fahrenheit.
3. Display the equivalent temperature in Fahrenheit.

$$F = \frac{9}{5}C + 32$$

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit;

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

    fahrenheit = (9.0 / 5.0) * celsius + 32;

    printf("Temperature in Fahrenheit: %.2f°F\n", fahrenheit);

    return 0;
}
```

11) Write a C program that takes a character as input and prints its ASCII value.

The program should:

- 1. Prompt the user to enter a single character.**
- 2. Determine the ASCII value of the entered character.**
- 3. Display the character and its corresponding ASCII value.**

Solution:

```
#include <stdio.h>

int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    printf("The ASCII value of '%c' is %d\n", ch, ch);

    return 0;
}
```

12) Write a C program that takes three numbers as input. The program should:

- 1. Prompt the user to enter three integers.**
- 2. Print them in reverse order, separated by a space ().**

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, num3;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    printf("Numbers in reverse order: %d %d %d\n", num3, num2, num1);
    return 0;
}
```




Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

- 1) Write a C program that takes two numbers as input and swaps their values using a third (temporary) variable. The program should:**
- Prompt the user to enter two integers.**
 - Use a third variable to swap their values.**
 - Print the swapped values.**

Solution:

```
#include <stdio.h>

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

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

    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

- 2) Write a C program that takes two numbers as input and swaps their values without using a third (temporary) variable. The program should:**

- 1. Prompt the user to enter two integers.**
- 2. Swap their values without using an extra variable.**
- 3. Display the swapped values.**

Solution:

```
#include <stdio.h>

int main() {
    int a, b;

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

    a = a + b;
    b = a - b;
    a = a - b;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

3) Write a C program that takes two numbers as input and swaps their values without using a third variable or arithmetic operations (+ or -), but instead using the bitwise XOR operator (^). The program should:

- 1. Prompt the user to enter two integers.**
- 2. Swap their values using XOR bitwise operations.**
- 3. Display the swapped values.**

Hint:

XOR (^) is a bitwise operator that flips bits where the two numbers differ.

1. First, $a = a \wedge b$ stores combined bitwise differences in a.
2. Next, $b = a \wedge b$ cancels out b's original bits, leaving b with a's original value.
3. Then, $a = a \wedge b$ cancels out a's modified bits, leaving a with b's original value.

This method swaps values without using extra space or arithmetic operations.

Example:

$a = 5$ (Binary: 0101)

$b = 9$ (Binary: 1001)

Step 1: $a = a \wedge b \rightarrow 0101 \wedge 1001 = 1100$ (a becomes 12)

Step 2: $b = a \wedge b \rightarrow 1100 \wedge 1001 = 0101$ (b becomes 5)

Step 3: $a = a \wedge b \rightarrow 1100 \wedge 0101 = 1001$ (a becomes 9)

Solution:

```
#include <stdio.h>

int main() {
    int a, b;

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

    a = a ^ b;
    b = a ^ b;
    a = a ^ b;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

- 4) Write a C program that takes an integer as input. The program should extract and print the last digit separately, followed by the remaining digits of the number. If the number is a single-digit number, the remaining digits should be displayed as 0.**

Solution:

```
#include <stdio.h>

int main() {
    int num, last, rest;

    printf("Enter a number: ");
    scanf("%d", &num);

    // Get the last digit
    last = num % 10;

    // Get the remaining digits
    rest = num / 10;

    printf("Last digit: %d\n", last);
    printf("Remaining digits: %d\n", rest);

    return 0;
}
```

- 5) Write a C program that takes a two-digit number as input. The program should reverse the digits and print the reversed number. Ensure that the input is always a two-digit number (10 to 99).**

Solution:

```
#include <stdio.h>

int main() {
    int num, first, last, reversed;

    printf("Enter a two-digit number: ");
    scanf("%d", &num);

    first = num / 10; // Get the first digit
    last = num % 10; // Get the last digit

    // Reverse the number
    reversed = (last * 10) + first;

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

-
- 6) Write a C program that takes a three-digit number as input and prints its reversed form. Ensure that the input is always a three-digit number (100 to 999) and does not contain leading zeros.**

Solution:

```
#include <stdio.h>

int main() {

    int num, first, middle, last, reversed;

    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    first = num / 100;           // Get the first digit
    middle = (num / 10) % 10;   // Get the middle digit
    last = num % 10;            // Get the last digit

    // Reverse the number
    reversed = (last * 100) + (middle * 10) + first;

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

- 7) Write a C program that takes the number of days as input and converts it into years, weeks, and remaining days. Assume that a year has 365 days and a week has 7 days.**

Solution:

```
#include <stdio.h>

int main() {
    int days, years, weeks, remaining_days;

    printf("Enter the number of days: ");
    scanf("%d", &days);

    years = days / 365;                      // Calculate number of years
    weeks = (days % 365) / 7;                  // Calculate remaining weeks
    remaining_days = (days % 365) % 7;         // Calculate remaining days

    printf("Years: %d\n", years);
    printf("Weeks: %d\n", weeks);
    printf("Days: %d\n", remaining_days);

    return 0;
}
```

- 8) Write a C program to determine and display the size (in bytes) of the fundamental data types: int, float, double, and char. Use the sizeof() operator to find the memory size of each type and print the results.**

Fun Fact: %lu is used in printf() because sizeof() returns a value of type **size_t**, which is unsigned long.

size_t is an unsigned integer type in C used to represent sizes and memory-related values. It is defined in stddef.h and stdlib.h.

Solution:

```
#include <stdio.h>

int main() {
    // Displaying the size of each data type using sizeof()
    printf("Size of int: %lu bytes\n", sizeof(int));
    printf("Size of float: %lu bytes\n", sizeof(float));
    printf("Size of double: %lu bytes\n", sizeof(double));
    printf("Size of char: %lu bytes\n", sizeof(char));

    return 0;
}
```

- 9) Write a C program to print "Hello, World!" without directly using the printf() function inside the main() function. Instead, define a macro that handles the printing, and use it in the main() function.**

Solution:

```
#include <stdio.h>
#define PRINT printf("Hello, World!\n")
int main() {
    PRINT;
    return 0;
}
```

10) Write a C program that takes three numbers as input. The program should:

- 1. Prompt the user to enter three integers.**
- 2. Double each number.**
- 3. Print the results separated by a colon (:).**

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, num3;

    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    printf("Doubled numbers: %d:%d:%d\n", num1 * 2, num2 * 2, num3 * 2);

    return 0;
}
```

11) Write a C program that takes a three-digit number as input and finds the difference between its first and last digit.

Solution:

```
#include <stdio.h>

int main() {
    int num, first, last, difference;

    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    first = num / 100; // Get the first digit
    last = num % 10; // Get the last digit

    difference = first - last;

    printf("Difference between first and last digit: %d\n", difference);

    return 0;
}
```

12) Write a C program that doubles the first digit, halves the last digit (integer division), keeps the middle digit unchanged, and prints the new number.

Solution:

```
#include <stdio.h>

int main() {
    int num, first, middle, last, modified_num;

    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    first = num / 100;           // First digit
    middle = (num / 10) % 10;    // Middle digit
    last = num % 10;            // Last digit

    first *= 2;     // Double the first digit
    last /= 2;      // Halve the last digit (integer division)

    modified_num = (first * 100) + (middle * 10) + last;

    printf("Modified number: %d\n", modified_num);

    return 0;
}
```

-
- 13) Write a C program that takes a three-digit number as input. The program should replace the middle digit with 0 while keeping the first and last digits unchanged, then print the modified number.**

Solution:

```
#include <stdio.h>

int main() {
    int num, first, middle, last, modifiedNum;

    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    first = num / 100;           // First digit
    middle = (num / 10) % 10;    // Middle digit
    last = num % 10;            // Last digit

    // Replace the middle digit with 0
    modifiedNum = (first * 100) + (0 * 10) + last;

    printf("Modified number: %d\n", modifiedNum);

    return 0;
}
```




Problem Solving with C
LABORATORY MANUAL
Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



**Department of CSE, PES
University**
UE24CS151B – Problem Solving with

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

1. Write a program that takes three numbers as input and calculates their average. Then, print the calculated average as output.

Solution:

```
#include <stdio.h>

int main() {
    float num1, num2, num3, avg;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &num1, &num2, &num3);

    avg = (num1 + num2 + num3) / 3;
    printf("Average: %.2f\n", avg);
    return 0;
}
```

2. Write a program that takes length and width as input and calculates the perimeter of the rectangle. Then, print the calculated perimeter as output.

Solution:

```
#include <stdio.h>

int main() {
    float length, width, perimeter;
    printf("Enter length and width: ");
    scanf("%f %f", &length, &width);
    perimeter = 2 * (length + width);
    printf("Perimeter of rectangle: %.2f\n", perimeter);
    return 0;
}
```

3. Write a C program to calculate the volume of a sphere using the mathematical formula:

$$\text{Volume} = \frac{4}{3} \pi r^3$$

where r is the radius of the sphere. Use the `math.h` for π (`M_PI`) and `pow()`. Print the volume as output.

Solution:

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

```

#include <math.h> // Including math library for pow() and
M_PI

int main() {
    double radius, volume;

    // Taking input
    printf("Enter the radius of the sphere: ");
    scanf("%lf", &radius);

    // Using the formula with pow() function
    volume = (4.0 / 3.0) * M_PI * pow(radius, 3);

    printf("Volume of the sphere: %.2lf cubic units\n",
volume);

    return 0;
}

```

4. Write a C program to find and print the first five multiples of a given number.

Print each multiple on a new line.

Solution:

```

#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Printing the first five multiples without using loops
    printf("First five multiples of %d:\n %d \n %d \n %d \n
%d \n %d\n", num, num * 2, num * 3, num * 4, num * 5);

    return 0;
}

```

5. Write a C program that takes a two-digit number as input. The program should reverse the digits and print the reversed number. Ensure that the input is always a two-digit number (10 to 99).

Solution:

```
#include <stdio.h>

int main() {
    int num, first, last, reversed;

    printf("Enter a two-digit number: ");
    scanf("%d", &num);

    first = num / 10; // Get the first digit
    last = num % 10; // Get the last digit

    // Reverse the number
    reversed = (last * 10) + first;

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

6. Write a program to Calculate the Remainder of two numbers. Print the value of the quotient and remainder as output.

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, quotient, remainder;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    quotient = num1 / num2;
    remainder = num1 % num2;

    printf("Quotient: %d\n", quotient);
```

```

    printf("Remainder: %d\n", remainder);
    return 0;
}

```

7. Write a C program that calculates the compound interest (CI) using the

$$CI = P \times \left(1 + \frac{R}{100}\right)^T - P$$

formula:

where:

- P = Principal amount
- R = Annual interest rate (%)
- T = Time in years

The program should take P (Principal), T (Time in years), and R (Rate of interest) as double inputs from the user and display the output in the same data type.

Solution:

```

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

int main() {
    double principal, rate, time, CI;
    printf("Enter principal, rate of interest, and time: ");
    scanf("%lf %lf %lf", &principal, &rate, &time);

    CI = principal * pow((1 + rate / 100), time) - principal;
    printf("Compound Interest: %.2lf\n", CI);
    return 0;
}

```

8. Write a C program that takes a time duration in seconds as input and converts it into hours, minutes, and remaining seconds.

Solution:

```

#include <stdio.h>

int main() {
    int seconds, hours, minutes, remaining_seconds;
    printf("Enter time in seconds: ");
    scanf("%d", &seconds);

    hours = seconds / 3600;

```

```

minutes = (seconds % 3600) / 60;
remaining_seconds = seconds % 60;

printf("%d seconds = %d hours %d minutes %d seconds\n",
seconds, hours, minutes, remaining_seconds);
return 0;
}

```

9. Write a C program that swaps two numbers using a temporary variable.

Solution:

```

#include <stdio.h>

int main() {
    int a, b, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}

```

10. Reverse a Two-Digit Number: Write a C program that takes a two-digit integer as input and prints its reverse.

example:

input = 38

output = 83

Solution:

```

#include <stdio.h>

int main() {
    int num, reversed;
    printf("Enter a two-digit number: ");
    scanf("%d", &num);

    reversed = (num % 10) * 10 + (num / 10);
    printf("Reversed number: %d\n", reversed);
    return 0;
}

```

```
}
```

11. Write a program that reads a date DD/MM/YYYY and print it in YYYY-MM-DD Format

```
#include <stdio.h>

int main() {
    int day, month, year;
    printf("Enter a date (DD/MM/YYYY) : ");
    scanf("%d/%d/%d", &day, &month, &year);

    printf("Formatted date: %d-%02d-%02d\n", year, month,
day);
    return 0;
}
```

12. Write a C Program to Multiply a Number by 8 Without * Operator.

Hint:

- **Left shifting (<<)** a number **by n positions** is equivalent to multiplying it by 2^n .
- Since $8 = 2^3$, shifting left by 3 (num << 3) **multiplies the number by 8**.

Solution:

```
#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Multiplying by 8 using bitwise shift
    printf("Result: %d\n", num << 3);

    return 0;
}
```

13. Write C program to find the sum of first n natural numbers. Print the sum as the output.

Hint: $(n * (n + 1)) / 2$

Solution:

```
#include <stdio.h>

int main() {
    int n, sum;

    // Taking input
    printf("Enter a positive integer N: ");
    scanf("%d", &n);

    // Using formula to find the sum
    sum = (n * (n + 1)) / 2;

    printf("Sum of first %d natural numbers: %d\n", n, sum);

    return 0;
}
```



Problem Solving with C
LABORATORY MANUAL
Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



**Department of CSE, PES
University**
UE24CS151B – Problem Solving with

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

1. Write a program that takes the base and height of a triangle as input and calculates its area.

Solution:

```
#include <stdio.h>

int main() {
    float base, height, area;
    printf("Enter base and height of the triangle: ");
    scanf("%f %f", &base, &height);

    area = 0.5 * base * height;
    printf("Area of Triangle: %.2f\n", area);
    return 0;
}
```

2. Write a program that takes three integer values as input from the user. The program should find the squares of the numbers and print them separated by %.

Solution:

```
#include <stdio.h>
int main() {
    int a, b, c;
    printf("Enter three integers\n");
    scanf("%d%d%d", &a, &b, &c);

    a = a*a;
    b = b*b;
    c = c*c;

    printf("%d%%d%%d%%d\n", a, b, c);
    return 0;
}
```

3. Write a program that takes the radius and height of a cylinder as input and calculates its volume. Formula : volume = pi* r^2 *h

Use the math.h for π (M_PI) and pow(). Print the volume as output.

Solution:

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

int main() {
```

```

double radius, height, volume;
printf("Enter radius and height of the cylinder: ");
scanf("%lf %lf", &radius, &height);

volume = M_PI * pow(radius, 2) * height;
printf("Volume of Cylinder: %.2lf cubic units\n",
volume);
return 0;
}

```

4. Write a program that takes an integer as input and prints its last digit.

Solution:

```

#include <stdio.h>

int main() {
    int num, lastDigit;
    printf("Enter an integer: ");
    scanf("%d", &num);

    lastDigit = num % 10;
    printf("Last digit: %d\n", lastDigit);
    return 0;
}

```

5. Find the ASCII Value of a Character

Problem Statement:

Write a C program that takes a **character** as input and prints its **ASCII value**.

Solution:

```

#include <stdio.h>

int main() {
    char ch;

    // Taking input
    printf("Enter a character: ");
    scanf(" %c", &ch);

    printf("ASCII value of '%c' is %d\n", ch, ch);
    return 0;
}

```

6. A pirate found a treasure chest but needs a secret code to unlock it. The code is the product of two given numbers. Write a C program that helps the pirate find the code.

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, code;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    code = num1 * num2;
    printf("The treasure code is %d.\n", code);
    return 0;
}
```

7. Write a program that takes a date in YYYY/MM/DD format as input and prints it as MM-DD-YYYY.

Solution:

```
#include <stdio.h>

int main() {
    int year, month, day;
    printf("Enter a date (YYYY/MM/DD) : ");
    scanf("%d/%d/%d", &year, &month, &day);

    printf("Formatted date: %02d-%02d-%d\n", month, day,
year);
    return 0;
}
```

8. Write a program that takes a three-digit number as input and prints its hundreds, tens, and ones digit, separated by a tab space.

Solution:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    printf("%d\t%d\t%d\n", num / 100, (num / 10) % 10, num %
10);
```

```
    return 0;  
}
```

9. Write a C program that takes the first term (a), common difference (d), and number of terms (n) as input and calculates the sum of the AP using the

$$S_n = \frac{n}{2} \times (2a + (n - 1) \times d)$$

formula.

Solution:

```
#include <stdio.h>  
  
int main() {  
    int a, d, n, sum;  
  
    // Taking input  
    printf("Enter the first term (a): ");  
    scanf("%d", &a);  
  
    printf("Enter the common difference (d): ");  
    scanf("%d", &d);  
  
    printf("Enter the number of terms (n): ");  
    scanf("%d", &n);  
  
    // Using the formula to calculate sum of AP  
    sum = (n * (2 * a + (n - 1) * d)) / 2;  
  
    // Display the result  
    printf("Sum of the Arithmetic Progression: %d\n", sum);  
  
    return 0;  
}
```

10. Write a C program that takes an uppercase letter as input and converts it to lowercase.

Solution:

```
#include <stdio.h>  
  
int main() {  
    char ch;  
  
    // Taking input
```

```

printf("Enter a lowercase letter: ");
scanf(" %c", &ch);

// Conversion formula
ch = ch + ('a' - 'A');

printf("Uppercase: %c\n", ch);
return 0;
}

```

11. Write a program that multiplies an integer by 16 using bitwise operators. (Bitwise left shift (<<)). Hint: num << 4 is equivalent to num * 16.

Solution:

```

#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Multiplying by 16 using bitwise shift
    printf("Result: %d\n", num << 4);

    return 0;
}

```

12. Write a C program that converts a given length from feet to meters. The program should take the length in feet as input and display the equivalent length in meters, formatted to 4 decimal places.

Solution:

```

#include <stdio.h>

int main() {
    float feet, meters;

    // Taking input
    printf("Enter length in feet: ");
    scanf("%f", &feet);

    // Conversion formula (1 foot = 0.3048 meters)
    meters = feet * 0.3048;

    printf("Length in meters: %.4f\n", meters);
}

```

```
    return 0;  
}
```

13. Write a program to swap two numbers using arithmetic operations instead of a temporary variable.

Solution:

```
#include <stdio.h>  
  
int main() {  
    int a, b;  
    printf("Enter two numbers: ");  
    scanf("%d %d", &a, &b);  
  
    a = a + b;  
    b = a - b;  
    a = a - b;  
  
    printf("After swapping: a = %d, b = %d\n", a, b);  
    return 0;  
}
```



Problem Solving with C
LABORATORY MANUAL
Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



**Department of CSE, PES
University**
UE24CS151B – Problem Solving with

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

1. Write a program that takes three numbers as input and calculates their average. Then, print the calculated average as output.

Solution:

```
#include <stdio.h>

int main() {
    float num1, num2, num3, avg;
    printf("Enter three numbers: ");
    scanf("%f %f %f", &num1, &num2, &num3);

    avg = (num1 + num2 + num3) / 3;
    printf("Average: %.2f\n", avg);
    return 0;
}
```

2. Write a program that takes length and width as input and calculates the perimeter of the rectangle. Then, print the calculated perimeter as output.

Solution:

```
#include <stdio.h>

int main() {
    float length, width, perimeter;
    printf("Enter length and width: ");
    scanf("%f %f", &length, &width);
    perimeter = 2 * (length + width);
    printf("Perimeter of rectangle: %.2f\n", perimeter);
    return 0;
}
```

3. Write a C program to calculate the volume of a sphere using the mathematical formula:

$$\text{Volume} = \frac{4}{3} \pi r^3$$

where r is the radius of the sphere. Use the `math.h` for π (`M_PI`) and `pow()`. Print the volume as output.

Solution:

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

```

#include <math.h> // Including math library for pow() and
M_PI

int main() {
    double radius, volume;

    // Taking input
    printf("Enter the radius of the sphere: ");
    scanf("%lf", &radius);

    // Using the formula with pow() function
    volume = (4.0 / 3.0) * M_PI * pow(radius, 3);

    printf("Volume of the sphere: %.2lf cubic units\n",
volume);

    return 0;
}

```

4. Write a C program to find and print the first five multiples of a given number.

Print each multiple on a new line.

Solution:

```

#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Printing the first five multiples without using loops
    printf("First five multiples of %d:\n %d \n %d \n %d \n
%d \n %d\n", num, num * 2, num * 3, num * 4, num * 5);

    return 0;
}

```

5. Write a C program that takes a two-digit number as input. The program should reverse the digits and print the reversed number. Ensure that the input is always a two-digit number (10 to 99).

Solution:

```
#include <stdio.h>

int main() {
    int num, first, last, reversed;

    printf("Enter a two-digit number: ");
    scanf("%d", &num);

    first = num / 10; // Get the first digit
    last = num % 10; // Get the last digit

    // Reverse the number
    reversed = (last * 10) + first;

    printf("Reversed number: %d\n", reversed);

    return 0;
}
```

6. Write a program to Calculate the Remainder of two numbers. Print the value of the quotient and remainder as output.

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, quotient, remainder;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);

    quotient = num1 / num2;
    remainder = num1 % num2;

    printf("Quotient: %d\n", quotient);
```

```

    printf("Remainder: %d\n", remainder);
    return 0;
}

```

7. Write a C program that calculates the compound interest (CI) using the

$$CI = P \times \left(1 + \frac{R}{100}\right)^T - P$$

formula:

where:

- P = Principal amount
- R = Annual interest rate (%)
- T = Time in years

The program should take P (Principal), T (Time in years), and R (Rate of interest) as double inputs from the user and display the output in the same data type.

Solution:

```

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

int main() {
    double principal, rate, time, CI;
    printf("Enter principal, rate of interest, and time: ");
    scanf("%lf %lf %lf", &principal, &rate, &time);

    CI = principal * pow((1 + rate / 100), time) - principal;
    printf("Compound Interest: %.2lf\n", CI);
    return 0;
}

```

8. Write a C program that takes a time duration in seconds as input and converts it into hours, minutes, and remaining seconds.

Solution:

```

#include <stdio.h>

int main() {
    int seconds, hours, minutes, remaining_seconds;
    printf("Enter time in seconds: ");
    scanf("%d", &seconds);

    hours = seconds / 3600;

```

```

minutes = (seconds % 3600) / 60;
remaining_seconds = seconds % 60;

printf("%d seconds = %d hours %d minutes %d seconds\n",
seconds, hours, minutes, remaining_seconds);
return 0;
}

```

9. Write a C program that swaps two numbers using a temporary variable.

Solution:

```

#include <stdio.h>

int main() {
    int a, b, temp;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d\n", a, b);
    return 0;
}

```

10. Reverse a Two-Digit Number: Write a C program that takes a two-digit integer as input and prints its reverse.

example:

input = 38

output = 83

Solution:

```

#include <stdio.h>

int main() {
    int num, reversed;
    printf("Enter a two-digit number: ");
    scanf("%d", &num);

    reversed = (num % 10) * 10 + (num / 10);
    printf("Reversed number: %d\n", reversed);
    return 0;
}

```

```
}
```

11. Write a program that reads a date DD/MM/YYYY and print it in YYYY-MM-DD Format

```
#include <stdio.h>

int main() {
    int day, month, year;
    printf("Enter a date (DD/MM/YYYY) : ");
    scanf("%d/%d/%d", &day, &month, &year);

    printf("Formatted date: %d-%02d-%02d\n", year, month,
day);
    return 0;
}
```

12. Write a C Program to Multiply a Number by 8 Without * Operator.

Hint:

- **Left shifting (<<)** a number **by n positions** is equivalent to multiplying it by 2^n .
- Since $8 = 2^3$, shifting left by 3 (num << 3) **multiplies the number by 8**.

Solution:

```
#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Multiplying by 8 using bitwise shift
    printf("Result: %d\n", num << 3);

    return 0;
}
```

13. Write C program to find the sum of first n natural numbers. Print the sum as the output.

Hint: $(n * (n + 1)) / 2$

Solution:

```
#include <stdio.h>

int main() {
    int n, sum;

    // Taking input
    printf("Enter a positive integer N: ");
    scanf("%d", &n);

    // Using formula to find the sum
    sum = (n * (n + 1)) / 2;

    printf("Sum of first %d natural numbers: %d\n", n, sum);

    return 0;
}
```



Problem Solving with C
LABORATORY MANUAL
Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



**Department of CSE, PES
University**
UE24CS151B – Problem Solving with

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

1. Write a program that takes the base and height of a triangle as input and calculates its area.

Solution:

```
#include <stdio.h>

int main() {
    float base, height, area;
    printf("Enter base and height of the triangle: ");
    scanf("%f %f", &base, &height);

    area = 0.5 * base * height;
    printf("Area of Triangle: %.2f\n", area);
    return 0;
}
```

2. Write a program that takes three integer values as input from the user. The program should find the squares of the numbers and print them separated by %.

Solution:

```
#include <stdio.h>
int main() {
    int a, b, c;
    printf("Enter three integers\n");
    scanf("%d%d%d", &a, &b, &c);

    a = a*a;
    b = b*b;
    c = c*c;

    printf("%d%%d%%d%%d\n", a, b, c);
    return 0;
}
```

3. Write a program that takes the radius and height of a cylinder as input and calculates its volume. Formula : volume = pi* r^2 *h

Use the math.h for π (M_PI) and pow(). Print the volume as output.

Solution:

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

int main() {
```

```

double radius, height, volume;
printf("Enter radius and height of the cylinder: ");
scanf("%lf %lf", &radius, &height);

volume = M_PI * pow(radius, 2) * height;
printf("Volume of Cylinder: %.2lf cubic units\n",
volume);
return 0;
}

```

4. Write a program that takes an integer as input and prints its last digit.

Solution:

```

#include <stdio.h>

int main() {
    int num, lastDigit;
    printf("Enter an integer: ");
    scanf("%d", &num);

    lastDigit = num % 10;
    printf("Last digit: %d\n", lastDigit);
    return 0;
}

```

5. Find the ASCII Value of a Character

Problem Statement:

Write a C program that takes a **character** as input and prints its **ASCII value**.

Solution:

```

#include <stdio.h>

int main() {
    char ch;

    // Taking input
    printf("Enter a character: ");
    scanf(" %c", &ch);

    printf("ASCII value of '%c' is %d\n", ch, ch);
    return 0;
}

```

6. A pirate found a treasure chest but needs a secret code to unlock it. The code is the product of two given numbers. Write a C program that helps the pirate find the code.

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, code;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    code = num1 * num2;
    printf("The treasure code is %d.\n", code);
    return 0;
}
```

7. Write a program that takes a date in YYYY/MM/DD format as input and prints it as MM-DD-YYYY.

Solution:

```
#include <stdio.h>

int main() {
    int year, month, day;
    printf("Enter a date (YYYY/MM/DD) : ");
    scanf("%d/%d/%d", &year, &month, &day);

    printf("Formatted date: %02d-%02d-%d\n", month, day,
year);
    return 0;
}
```

8. Write a program that takes a three-digit number as input and prints its hundreds, tens, and ones digit, separated by a tab space.

Solution:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a three-digit number: ");
    scanf("%d", &num);

    printf("%d\t%d\t%d\n", num / 100, (num / 10) % 10, num %
10);
```

```
    return 0;  
}
```

9. Write a C program that takes the first term (a), common difference (d), and number of terms (n) as input and calculates the sum of the AP using the

$$S_n = \frac{n}{2} \times (2a + (n - 1) \times d)$$

formula.

Solution:

```
#include <stdio.h>  
  
int main() {  
    int a, d, n, sum;  
  
    // Taking input  
    printf("Enter the first term (a): ");  
    scanf("%d", &a);  
  
    printf("Enter the common difference (d): ");  
    scanf("%d", &d);  
  
    printf("Enter the number of terms (n): ");  
    scanf("%d", &n);  
  
    // Using the formula to calculate sum of AP  
    sum = (n * (2 * a + (n - 1) * d)) / 2;  
  
    // Display the result  
    printf("Sum of the Arithmetic Progression: %d\n", sum);  
  
    return 0;  
}
```

10. Write a C program that takes an uppercase letter as input and converts it to lowercase.

Solution:

```
#include <stdio.h>  
  
int main() {  
    char ch;  
  
    // Taking input
```

```

printf("Enter a lowercase letter: ");
scanf(" %c", &ch);

// Conversion formula
ch = ch + ('a' - 'A');

printf("Uppercase: %c\n", ch);
return 0;
}

```

11. Write a program that multiplies an integer by 16 using bitwise operators. (Bitwise left shift (<<)). Hint: num << 4 is equivalent to num * 16.

Solution:

```

#include <stdio.h>

int main() {
    int num;

    // Taking input
    printf("Enter a number: ");
    scanf("%d", &num);

    // Multiplying by 16 using bitwise shift
    printf("Result: %d\n", num << 4);

    return 0;
}

```

12. Write a C program that converts a given length from feet to meters. The program should take the length in feet as input and display the equivalent length in meters, formatted to 4 decimal places.

Solution:

```

#include <stdio.h>

int main() {
    float feet, meters;

    // Taking input
    printf("Enter length in feet: ");
    scanf("%f", &feet);

    // Conversion formula (1 foot = 0.3048 meters)
    meters = feet * 0.3048;

    printf("Length in meters: %.4f\n", meters);
}

```

```
    return 0;  
}
```

13. Write a program to swap two numbers using arithmetic operations instead of a temporary variable.

Solution:

```
#include <stdio.h>  
  
int main() {  
    int a, b;  
    printf("Enter two numbers: ");  
    scanf("%d %d", &a, &b);  
  
    a = a + b;  
    b = a - b;  
    a = a - b;  
  
    printf("After swapping: a = %d, b = %d\n", a, b);  
    return 0;  
}
```



**Problem Solving with C
LABORATORY MANUAL
Week 2**

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

- 1. Write a program that takes two integers as input and calculates their sum. Print the result.**

Solution:

```
#include <stdio.h>

int main() {
    int a, b, sum;
    printf("Enter two integers: ");
    scanf("%d %d", &a, &b);
    sum = a + b;
    printf("Sum = %d\n", sum);
    return 0;
}
```

2. Take a float (say X) as input and perform the following operations:

- **Multiply X by the maximum value of FLOAT allowed in 'C'.**
- **Divide X by the minimum positive value of FLOAT in 'C'.**

Solution:

```
#include <stdio.h>

#include <float.h>

int main() {

    float X;

    printf("Enter a float value: ");

    scanf("%f", &X);

    printf("X * FLOAT_MAX = %e\n", X * FLT_MAX);

    printf("X / FLT_MIN = %e\n", X / FLT_MIN);

    return 0;

}
```

- 3. Write a program to calculate the compound interest (CI) based on the given principal (P), rate of interest (R), time period (T), and number of times interest is compounded per year (N). Use the formula:**

$$CI = P * \left(1 + \frac{R}{N * 100}\right)^{N*T}$$

Solution:

```
#include <stdio.h>

#include <math.h>

int main() {

    float P, R, T, N, CI;

    printf("Enter Principal, Rate, Time, and Number of compounds per year: ");

    scanf("%f %f %f %f", &P, &R, &T, &N);

    CI = P * pow(1 + (R / (N * 100)), N * T);

    printf("Compound Interest =%.2f\n", CI);

    return 0;

}
```

4. Write a program that takes four integer values as input and doubles each value.

Print the results separated by a semicolon (;).

Solution:

```
#include <stdio.h>

int main() {

    int a, b, c, d;

    printf("Enter four integers: ");

    scanf("%d %d %d %d", &a, &b, &c, &d);

    printf("%d; %d; %d; %d\n", 2 * a, 2 * b, 2 * c, 2 * d);

    return 0;

}
```

- 5. Create a program that takes the circumference of a circle as input and calculates its radius using the formula: $r = C / 2 * \pi$. Use math.h for π (M_PI).**

Solution:

```
#include <stdio.h>

#include <math.h>

int main() {

    float C, r;

    printf("Enter circumference: ");

    scanf("%f", &C);

    r = C / (2 * M_PI);

    printf("Radius = %.2f\n", r);

    return 0;

}
```

- 6. Write a program to take the cost price and selling price of a product as input and calculate the profit or loss.**

Solution:

```
#include <stdio.h>

int main() {

    float cp, sp;

    printf("Enter cost price and selling price: ");

    scanf("%f %f", &cp, &sp);

    float result = sp - cp;

    if (result > 0)

        printf("Profit = %.2f\n", result);

    else if (result < 0)

        printf("Loss = %.2f\n", -result);

    else

        printf("No Profit No Loss\n");

    return 0;

}
```

- 7. Write a C program to read and print a character. Also, print the ASCII value of the character.**

Solution:

```
#include <stdio.h>

int main() {

    char ch;

    printf("Enter a character: ");

    scanf(" %c", &ch);

    printf("Character: %c, ASCII: %d\n", ch, ch);

    return 0;

}
```

- 8. Write a C program to read an uppercase letter from the user and convert it to lowercase without using formatted I/O functions.**

Solution:

```
#include <stdio.h>

int main() {

    char ch;

    printf("Enter an uppercase letter: ");

    scanf(" %c", &ch);

    ch = ch + 32;

    printf("Lowercase: %c\n", ch);

    return 0;

}
```

- 9. Write a program to read an integer and display its binary and octal representation.**

Solution:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);
    printf("Binary: %#b (not directly supported in C)\n");
    printf("Octal: %#o\n", num);
    return 0;
}
```

10. Write a program that takes a floating-point number as input and prints it in fixed-point notation.

Solution:

```
#include <stdio.h>

int main() {

    float num;

    printf("Enter a floating number: ");

    scanf("%f", &num);

    printf("Fixed notation: %.2f\n", num);

    return 0;

}
```

11. Write a program to read an integer and display:

- **The number raised to the power of 3.**
- **The square root of the number (use math.h).**

Solution:

```
#include <stdio.h>

#include <math.h>

int main() {

    int num;

    printf("Enter an integer: ");

    scanf("%d", &num);

    printf("Cube: %d\n", num * num * num);

    printf("Square Root: %.2f\n", sqrt(num));

    return 0;

}
```

12. Write a program that takes a number of hours as input and converts it into equivalent days and weeks.

Solution:

```
#include <stdio.h>

int main() {

    int hours;

    printf("Enter number of hours: ");

    scanf("%d", &hours);

    printf("Days: %d, Weeks: %d\n", hours / 24, hours / 168);

    return 0;

}
```

13. Write a program that takes four integers as input and increases each by 2, then prints them separated by a forward slash (/).

Solution:

```
#include <stdio.h>

int main() {

    int a, b, c, d;

    printf("Enter four integers: ");

    scanf("%d %d %d %d", &a, &b, &c, &d);

    printf("%d/%d/%d/%d\n", a + 2, b + 2, c + 2, d + 2);

    return 0;

}
```



**Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 2**



**Problem Solving with C
LABORATORY MANUAL
Week 2**

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

- 1. Write a program that takes three integer inputs and calculates their average. Print the result.**

Solution:

```
#include <stdio.h>

int main() {
    int a, b, c;
    scanf("%d %d %d", &a, &b, &c);
    printf("Average: %.2f\n", (a + b + c) / 3.0);
    return 0;
}
```

2. Take an integer and a float as input and perform:

- **Multiply the integer by the float.**
- **Divide the float by the integer.**

Solution:

```
#include <stdio.h>

int main() {

    int x;

    float y;

    scanf("%d %f", &x, &y);

    printf("Multiply: %.2f\nDivide: %.2f\n", x * y, y / x);

    return 0;

}
```

3. Write a program to calculate the Body Mass Index (BMI) given weight (W) in kg and height (H) in meters using the formula:

$$BMI = \frac{W}{H^2}$$

Solution:

```
#include <stdio.h>

int main() {
    float w, h;
    scanf("%f %f", &w, &h);
    printf("BMI: %.2f\n", w / (h * h));
    return 0;
}
```

- 4. Write a program that takes three integer values as input and swaps their values in a cyclic order.**

Solution:

```
#include <stdio.h>

int main() {

    int a, b, c, temp;

    scanf("%d %d %d", &a, &b, &c);

    temp = a;

    a = b;

    b = c;

    c = temp;

    printf("%d %d %d\n", a, b, c);

    return 0;

}
```

- 5. Create a program that calculates the perimeter of a rectangle given its length and width. Use the formula:**

$$P = 2 \times (L + W)$$

Solution:

```
#include <stdio.h>

int main() {
    int l, w;
    scanf("%d %d", &l, &w);
    printf("Perimeter: %d\n", 2 * (l + w));
    return 0;
}
```

- 6. Write a program to take the base and height of a triangle and compute its area.**

Solution:

```
#include <stdio.h>

int main() {

    float b, h;

    scanf("%f %f", &b, &h);

    printf("Area: %.2f\n", 0.5 * b * h);

    return 0;

}
```

- 7. Write a program to take the cost price and selling price of a product as input and calculate the profit or loss.**

Solution:

```
#include <stdio.h>

int main() {
    float cost_price, selling_price, result;

    printf("Enter Cost Price: ");
    scanf("%f", &cost_price);

    printf("Enter Selling Price: ");
    scanf("%f", &selling_price);

    result = selling_price - cost_price;

    printf("Profit/Loss: %.2f", result);

    return 0;
}
```

- 8. Write a C program to read a lowercase letter and convert it to uppercase using ASCII arithmetic.**

Solution:

```
#include <stdio.h>

int main() {
    char ch;
    scanf(" %c", &ch);
    printf("Uppercase: %c\n", ch - 32);
    return 0;
}
```

- 9. Write a program to read a floating-point number and display its equivalent hexadecimal representation.**

Solution:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a decimal number: ");
    scanf("%d", &num);
    printf("Hexadecimal: %#X\n", num);
    return 0;
}
```

- 10. Write a program that takes a floating-point number as input and prints its value rounded to two decimal places.**

Solution:

```
#include <stdio.h>

int main() {

    float num;

    printf("Enter a floating-point number: ");

    scanf("%f", &num);

    printf("Fixed notation: %.2f\n", num);

    return 0;

}
```

11. Write a program that reads an integer and prints:

- Its square.
- Its cube.
- Its square root (use math.h).

Solution:

```
#include <stdio.h>

#include <math.h>

int main() {

    int num;

    printf("Enter an integer: ");

    scanf("%d", &num);

    printf("Square: %d\n", num * num);

    printf("Cube: %d\n", num * num * num);

    printf("Square Root: %.2f\n", sqrt(num));

    return 0;

}
```

- 12. Write a program that takes the number of seconds as input and converts it into hours, minutes, and seconds.**

Solution:

```
#include <stdio.h>

int main() {

    int seconds, hours, minutes;

    printf("Enter time in seconds: ");

    scanf("%d", &seconds);

    hours = seconds / 3600;

    minutes = (seconds % 3600) / 60;

    seconds = seconds % 60;

    printf("%d hours %d minutes %d seconds\n", hours, minutes, seconds);

    return 0;

}
```

- 13. Write a program that takes three integers as input, decreases each by 2, and prints them separated by a pipe symbol (|).**

Solution:

```
#include <stdio.h>

int main() {

    int a, b, c;

    printf("Enter three integers: ");

    scanf("%d %d %d", &a, &b, &c);

    printf("%d|%d|%d\n", a - 2, b - 2, c - 2);

    return 0;

}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Jan 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information

Commands to execute Programs using gcc

1. Editing

\$gedit filename.c

2. Compile

\$ gcc -c filename.c

3. Linking

\$ gcc filename.o

4. Loading - Execute

\$./a.out (Linux Operating System)

\$ a.exe (Windows Operating System)

SET 1 SOLUTIONS

Solution 1:

```
#include <stdio.h>
int main() {
    int base, height;
    float area;

    printf("Enter base and height: ");
    scanf("%d %d", &base, &height);

    area = 0.5 * base * height;

    printf("Area of triangle: %.2f\n", area);
    return 0;
}
```

Solution 2:

```
#include <stdio.h>
#include <limits.h>
int main() {
    int X;
    long long Y;

    printf("Enter integer X and long long Y: ");
    scanf("%d %lld", &X, &Y);

    printf("INT_MIN + X = %d\n", INT_MIN + X);
    printf("LLONG_MIN + Y = %lld\n", LLONG_MIN + Y);
    return 0;
}
```

Solution 3:

```
#include <stdio.h>
#include <math.h>
int main() {
    float principal, rate, time, CI;

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

    CI = principal * (pow((1 + rate/100), time)) - principal;

    printf("Compound Interest: %.2f\n", CI);
    return 0;
}
```

Solution 4:

```
#include <stdio.h>
int main() {
    int num1, num2, num3;

    printf("Enter three integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    num1 += 2;
    num2 += 2;
    num3 += 2;

    printf("%d$%d$%d\n", num1, num2, num3);
    return 0;
}
```

Solution 5:

```
#include <stdio.h>
#include <math.h>
int main() {
    float side, area, perimeter;

    printf("Enter side length: ");
    scanf("%f", &side);

    area = side * side;
    perimeter = 4 * side;

    printf("Area: %.2f\n", area);
    printf("Perimeter: %.2f\n", perimeter);
    return 0;
}
```

Solution 6:

```
#include <stdio.h>
int main() {
    float food_cost, tip_percent, total_bill;

    printf("Enter food cost: ");
    scanf("%f", &food_cost);
    printf("Enter tip percentage: ");
    scanf("%f", &tip_percent);

    total_bill = food_cost + (food_cost * tip_percent / 100);

    printf("Total bill: $%.2f\n", total_bill);
    return 0;
}
```

Solution 7:

```
#include<stdio.h>
Int main()
{
    # Get user input
    num = int(input("Enter a three-digit number: "))

    # Extract digits
    hundreds = num // 100 # Get the hundreds place
    tens = (num % 100) // 10 # Get the tens place
    ones = num % 10 # Get the ones place

    # Print the digits separated by a tab space
    print(hundreds, tens, ones, sep='\t')
}
```

Solution 8:

```
#include <stdio.h>
int main() {
    int num1, num2, num3;

    printf("Enter three integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    num1 -= 5;
    num2 -= 5;
    num3 -= 5;

    printf("%d@%d@%d\n", num1, num2, num3);
    return 0;
}
```

Solution 9:

```
#include <stdio.h>
int main() {
    float mass, velocity, kinetic_energy;

    printf("Enter mass and velocity: ");
    scanf("%f %f", &mass, &velocity);

    kinetic_energy = 0.5 * mass * velocity * velocity;

    printf("Kinetic Energy: %.2f\n", kinetic_energy);
    return 0;
}
```

Solution 10:

```
#include <stdio.h>
int main() {
    int num, last_digit;

    printf("Enter an integer: ");
    scanf("%d", &num);

    last_digit = num % 10;

    printf("Last digit: %d\n", last_digit);
    return 0;
}
```

Solution 11:

```
#include <stdio.h>
int main() {
    int seconds, hours, minutes, remaining_seconds;
    printf("Enter time in seconds: ");
    scanf("%d", &seconds);
    hours = seconds / 3600;
    minutes = (seconds % 3600) / 60;
    remaining_seconds = seconds % 60;
    printf("%d seconds is equivalent to %d hours, %d
minutes, and %d seconds.\n", seconds, hours,
minutes, remaining_seconds);
}
```

Solution 12:

```
#include <stdio.h>

int main() {
    double feet, meters;

    // Prompt user for input
    printf("Enter length in feet: ");
    scanf("%lf", &feet);

    // Conversion factor: 1 foot = 0.3048 meters
    meters = feet * 0.3048;

    // Display result formatted to 4 decimal places
    printf("Equivalent length in meters: %.4f\n", meters);

    return 0;
}
```

Solution 13:

```
#include <stdio.h>
int main() {
    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    printf("ASCII value of %c is %d\n", ch, ch);
    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre

Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Commands to execute Programs using gcc:

1. Editing

```
$ gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 1. Sum of Two Numbers:** Write a program that takes two integers as input and prints their sum.

Solution:

```
#include <stdio.h>

int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    printf("Sum = %d", a + b);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

2. **Swap Two Numbers:** Write a program that swaps two numbers without using a third variable.

Solution:

```
#include <stdio.h>

int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    a = a + b;
    b = a - b;
    a = a - b;
    printf("Swapped numbers: %d %d", a, b);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

3. **ASCII Value of a Character:** Write a program that takes a character as input and prints its ASCII value.

Solution:

```
#include <stdio.h>

int main() {

    char ch;

    printf("Enter a character: ");
    scanf("%c", &ch);

    printf("ASCII Value = %d", ch);

    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

4. A scientist has invented a time machine that allows a person to travel back in time. The user enters the current year and the number of years to go back. Your task is to write a C program that calculates and displays the past year where the person arrives after traveling.

Input :

2025 10

Output:

You have traveled back to the year 2015.

Solution:

```
#include <stdio.h>

int main() {
    int currentYear, yearsBack, pastYear;
    printf("Enter current year and years to travel back: ");
    scanf("%d %d", &currentYear, &yearsBack);
    pastYear = currentYear - yearsBack;
    printf("You have traveled back to the year %d.\n", pastYear);
    return 0;
}
```



-
5. **Perimeter of a Rectangle:** Write a program that takes the length and width of a rectangle as input and prints its perimeter.

Solution:

```
#include <stdio.h>

int main() {
    int length, width;
    printf("Enter length and width: ");
    scanf("%d %d", &length, &width);
    printf("Perimeter = %d", 2 * (length + width));
    return 0;
}
```



6. **Celsius to Fahrenheit Conversion:** Write a program that converts a given Celsius temperature to Fahrenheit.

Solution:

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * 9 / 5) + 32;
    printf("Fahrenheit = %.2f", fahrenheit);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 7. A wizard has placed a magical door that only opens when given a special magic number. The magic number is found by adding two secret numbers together. Your task is to write a C program that takes two numbers as input, calculates their sum, and displays the magic number that will unlock the door.**

Solution:

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

```
int main() {  
    int num1, num2, magicNumber;  
    printf("Enter two numbers: ");  
    scanf("%d %d", &num1, &num2);  
    magicNumber = num1 + num2;  
    printf("The magic number is %d.\n", magicNumber);  
    return 0;  
}
```



8. Reverse a Number: Write a program that reverses a given integer.

Solution:

```
#include <stdio.h>

int main() {

    int num, reversed = 0;

    printf("Enter a number: ");
    scanf("%d", &num);

    while (num != 0) {

        reversed = reversed * 10 + num % 10;

        num /= 10;
    }

    printf("Reversed Number = %d", reversed);

    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 9. A pirate has discovered a treasure chest, but it is locked with a secret code. The code is found by multiplying two given numbers. Write a C program that takes two numbers from the user, calculates their product, and prints the treasure code to unlock the chest.**

Solution:

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

```
int main() {
    int num1, num2, code;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    code = num1 * num2;
    printf("The treasure code is %d.\n", code);
    return 0;
}
```



- 10. A treasure is buried X meters to the east and Y meters to the north from the starting position (0,0). Write a C program that takes the X and Y coordinates as input and calculates the distance to the treasure using the distance formula:**

$$\text{Distance} = \sqrt{X^2 + Y^2}$$

Solution:

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

int main(){
    int x, y;
    double distance;

    printf("Enter the treasure's X and Y coordinates: ");
    scanf("%d %d", &x, &y);

    distance = sqrt(x * x + y * y);
    printf("The treasure is %.2f meters away.\n", distance);

    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

**11. Aliens have a different age system! On their planet, 1 Earth year = 2 Alien years.
Write a C program that asks the user for their Earth age, converts it into Alien years,
and displays the result.**

Solution:

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

```
int main() {
    int earthAge, alienAge;
    printf("Enter your age on Earth: ");
    scanf("%d", &earthAge);
    alienAge = earthAge * 2;
    printf("Your age on the alien planet is %d years.\n", alienAge);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

12. A scientist is making a secret potion and needs to mix two liquids. The final volume of the potion is the sum of the two liquid volumes. Write a C program that takes two volume values as input and prints the total potion volume.

Solution:

```
#include <stdio.h>

int main() {
    int volume1, volume2, totalVolume;
    printf("Enter two liquid volumes (in ml): ");
    scanf("%d %d", &volume1, &volume2);
    totalVolume = volume1 + volume2;
    printf("The secret potion volume is %d ml.\n", totalVolume);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

13. A rocket is launching to space, and its speed is measured in meters per second.

To communicate with scientists, we must convert its speed to kilometers per hour. Write a C program that takes a speed in meters per second, converts it to km/h, and displays the result.

Formula: Speed (km/h) = Speed (m/s) × 3.6

Solution:

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

```
int main() {
    float speed_mps, speed_kph;-
    printf("Enter the rocket's speed in meters per second: ");
    scanf("%f", &speed_mps);
    speed_kph = speed_mps * 3.6;
    printf("The rocket's speed is %.2f km/h.\n", speed_kph);
    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre

Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Commands to execute Programs using gcc:

1. Editing

```
$ gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

- Swap Two Numbers:** Write a program that swaps two numbers without using a third variable.

Solution:

```
#include <stdio.h>

int main() {
    int a, b;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    a = a + b;
    b = a - b;
    a = a - b;
    printf("Swapped numbers: %d %d", a, b);
    return 0;
}
```



2. **Reading Multiple Inputs:** Write a program that takes three integers as input and prints them in reverse order.

Solution:

```
#include <stdio.h>

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

    printf("Enter three integers: ");
    scanf("%d %d %d", &a, &b, &c);

    printf("Reversed order: %d %d %d\n", c, b, a);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

3. **Calculate Average:** Write a program that takes five floating-point numbers as input and prints their average.

Solution:

```
#include <stdio.h>

int main() {
    float num1, num2, num3, num4, num5, average;

    printf("Enter five floating-point numbers: ");
    scanf("%f %f %f %f", &num1, &num2, &num3, &num4, &num5);

    average = (num1 + num2 + num3 + num4 + num5) / 5.0;
    printf("Average: %.2f\n", average);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

4. **Calculate Area of a Circle:** Write a program that takes the radius of a circle as input and prints its area.

Solution:

```
#include <stdio.h>

#define PI 3.14159

int main() {
    float radius, area;

    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);

    area = PI * radius * radius;
    printf("Area of the circle: %.2f\n", area);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

5. **Temperature Conversion (Fahrenheit to Celsius):** Write a program that converts a given Fahrenheit temperature to Celsius.

Solution:

```
#include <stdio.h>

int main() {
    float fahrenheit, celsius;

    printf("Enter temperature in Fahrenheit: ");
    scanf("%f", &fahrenheit);

    celsius = (fahrenheit - 32) * 5 / 9;
    printf("Temperature in Celsius: %.2f\n", celsius);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

6. A scientist has invented a time machine that allows a person to travel back in time. The user enters the current year and the number of years to go back. Your task is to write a C program that calculates and displays the past year where the person arrives after traveling.

Input :

2025 10

Output:

You have traveled back to the year 2015.

Solution:

```
#include <stdio.h>

int main() {
    int currentYear, yearsBack, pastYear;
    printf("Enter current year and years to travel back: ");
    scanf("%d %d", &currentYear, &yearsBack);
    pastYear = currentYear - yearsBack;
    printf("You have traveled back to the year %d.\n", pastYear);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

7. **Perimeter of a Rectangle:** Write a program that takes the length and width of a rectangle as input and prints its perimeter.

Solution:

```
#include <stdio.h>

int main() {
    int length, width;
    printf("Enter length and width: ");
    scanf("%d %d", &length, &width);
    printf("Perimeter = %d", 2 * (length + width));
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 8. A wizard has placed a magical door that only opens when given a special magic number. The magic number is found by adding two secret numbers together. Your task is to write a C program that takes two numbers as input, calculates their sum, and displays the magic number that will unlock the door.**

Solution:

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

```
int main() {  
    int num1, num2, magicNumber;  
    printf("Enter two numbers: ");  
    scanf("%d %d", &num1, &num2);  
    magicNumber = num1 + num2;  
    printf("The magic number is %d.\n", magicNumber);  
    return 0;  
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

9. **Lower to Upper Case Conversion:** Write a program that takes a lowercase character and converts it to uppercase.

Solution:

```
#include <stdio.h>

int main() {
    char ch;
    printf("Enter a lowercase character: ");

    scanf(" %c", &ch);
    printf("Uppercase: %c", ch - 32);

    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 10. Age in years:** Write a program that takes your birth year as input and prints your age.

Solution:

```
#include <stdio.h>

int main() {
    int birthYear, currentYear = 2025;
    printf("Enter your birth year: ");
    scanf("%d", &birthYear);
    printf("Your age is: %d years\n", currentYear - birthYear);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

11. Simple Interest Calculation: Write a program that takes principal, rate, and time as input and calculates simple interest.

Solution:

```
#include <stdio.h>

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

    printf("Enter principal amount: ");
    scanf("%f", &principal);

    printf("Enter rate of interest: ");
    scanf("%f", &rate);

    printf("Enter time in years: ");
    scanf("%f", &time);

    interest = (principal * rate * time) / 100;
    printf("Simple Interest: %.2f\n", interest);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

12. Double the number: Write a program that takes a number as input and prints its double.

Solution:

```
#include <stdio.h>

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Twice the number is: %d\n", num * 2);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

13. A pirate has discovered a treasure chest, but it is locked with a secret code. The code is found by multiplying two given numbers. Write a C program that takes two numbers from the user, calculates their product, and prints the treasure code to unlock the chest.

Solution:

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

```
int main() {
    int num1, num2, code;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    code = num1 * num2;
    printf("The treasure code is %d.\n", code);
    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Jan 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre
Acknowledge Prof. Sowmyashree as the source of this information

Commands to execute Programs using gcc

1. Editing

`$gedit filename.c`

2. Compile

`$ gcc -c filename.c`

3. Linking

`$ gcc filename.o`

4. Loading - Execute

`$./a.out (Linux Operating System)`

`$ a.exe (Windows Operating System)`

Solution 1:

```
#include <stdio.h>
int main() {
    float score1, score2, score3, weighted_avg;

    printf("Enter three test scores (0-100): ");
    scanf("%f %f %f", &score1, &score2, &score3);

    weighted_avg = (score1 * 0.4) + (score2 * 0.3) + (score3 * 0.3);

    printf("Scores: %.1f~%.1f~%.1f\n", score1, score2, score3);
    printf("Weighted Average: %.2f\n", weighted_avg);

    return 0;
}
```

Solution 2 :

```
#include <stdio.h>

int main() {
    int num1, num2, num3;
    float average;

    // Taking three integer inputs
    printf("Enter three integers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    // Calculating the average
    average = (num1 + num2 + num3) / 3.0;

    // Printing the result
    printf("The average is: %.2f\n", average);

    return 0;
}
```

Solution 3:

```
#include <stdio.h>
int main() {
    float celsius;
    int width;

    printf("Enter temperature in Celsius: ");
    scanf("%f %d", &celsius);

    float fahrenheit = (celsius * 9/5) + 32;
    float kelvin = celsius + 273.15;

    printf("%*f\n", width, celsius); // Right-aligned
    printf("%-*f\n", width, fahrenheit); // Left-aligned
    printf("%.*f\n", width, width/2, kelvin); // Centered

    return 0;
}
```

Solution 4:

```
#include <stdio.h>
int main() {
    char item_name[50];
    int quantity;
    float price, total;

    printf("Enter item name: ");
    scanf("%s", item_name);
    printf("Enter quantity: ");
    scanf("%d", &quantity);
    printf("Enter price: ");
    scanf("%f", &price);

    total = quantity * price;

    printf("\n===== RECEIPT =====\n");
    printf("%-15s %5s %10s\n", "Item", "Qty", "Price");
    printf("%-15s %5d %10.2f\n", item_name, quantity, price);
    printf("-----\n");
    printf("Total:%23.2f\n", total);

    return 0;
}
```

Solution 5:

```
#include <stdio.h>

int main() {
    int age;
    float height, weight;

    printf("Enter your age: ");
    scanf("%d", &age);

    printf("Enter your height (in meters): ");
    scanf("%f", &height);

    printf("Enter your weight (in kg): ");
    scanf("%f", &weight);

    printf("Age: %d years\n", age);
    printf("Height: %.2f meters\n", height);
    printf("Weight: %.2f kg\n", weight);

    return 0;
}
```

Solution 6:

```
#include <stdio.h>

int main() {
    int currentYear, yearsBack, pastYear;
    printf("Enter current year and years to travel back: ");
    scanf("%d %d", &currentYear, &yearsBack);
    pastYear = currentYear - yearsBack;
    printf("You have traveled back to the year %d.\n", pastYear);
    return 0;
}
```

Solution 7:

```
#include <stdio.h>
int main() {
    int num;

    printf("Enter a 4-digit number: ");
    scanf("%d", &num);

    printf("Thousands: %d\n", num/1000);
    printf("Hundreds: %d\n", (num/100)%10);
    printf("Tens: %d\n", (num/10)%10);
    printf("Ones: %d\n", num%10);

    return 0;
}
```

Solution 8:

```
#include <stdio.h>
int main() {
    float distance, waiting_time, total_fare;
    const float BASE_FARE = 5.0;

    printf("Enter distance (km): ");
    scanf("%f", &distance);
    printf("Enter waiting time (minutes): ");
    scanf("%f", &waiting_time);

    total_fare = BASE_FARE + (distance * 2.0) + (waiting_time * 0.25);

    printf("Total fare: $%.2f\n", total_fare);

    return 0;
}
```

Solution 9:

```
#include <stdio.h>
#include <math.h>
int main() {
    float radius, height, volume;

    printf("Enter radius and height: ");
    scanf("%f %f", &radius, &height);

    volume = (1.0/3.0) * M_PI * radius * radius * height;

    printf("Volume: %.3f\n", volume);

    return 0;
}
```

Solution 10:

```
#include <stdio.h>
int main() {
    int r, g, b;

    printf("Enter RGB values (0-255): ");
    scanf("%d %d %d", &r, &g, &b);

    r = r - 50 ;
    g = g - 50 ;
    b = b - 50 ;

    printf("%d:%d:%d\n", r, g, b);

    return 0;
}
```

Solution 11:

```
#include <stdio.h>
int main() {
    float distance, time, speed_kmh, speed_ms;

    printf("Enter distance (km) and time (minutes): ");
    scanf("%f %f", &distance, &time);

    speed_kmh = (distance / time) * 60; // Convert to km/h
    speed_ms = speed_kmh * (1000.0/3600.0); // Convert to m/s

    printf("Speed: %.2f km/h\n", speed_kmh);
    printf("Speed: %.2f m/s\n", speed_ms);

    return 0;
}
```

Solution 12:

```
#include <stdio.h>

int main() {
    int num1, num2, magicNumber;
    printf("Enter two numbers: ");
    scanf("%d %d", &num1, &num2);
    magicNumber = num1 + num2;
    printf("The magic number is %d.\n", magicNumber);
    return 0;
}
```

Solution 13:

```
#include <stdio.h>
int main() {
    int num_coffees;
    float price_per_coffee, total_cost;

    printf("Enter number of coffees: ");
    scanf("%d", &num_coffees);
    printf("Enter price per coffee: $");
    scanf("%f", &price_per_coffee);

    total_cost = num_coffees * price_per_coffee;
    total_cost = total_cost + (total_cost * 0.05); // Add 5% service charge

    printf("Total cost with service charge: $%.2f\n", total_cost);

    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre

Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Commands to execute Programs using gcc:

1. Editing

```
$ gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 1. A merchant counts his gold coins daily. Write a program that takes the number of gold coins as input and prints it.**

Solution:

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

```
int main() {
    int goldCoins;
    printf("Enter the number of gold coins: ");
    scanf("%d", &goldCoins);
    printf("The merchant has %d gold coins.\n", goldCoins);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

2. A wizard mixes two potions in a special ratio. Write a program to take two floating-point numbers as input and print them.

Solution:

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

```
int main() {
    float potion1, potion2;
    printf("Enter the quantities of two potions: ");
    scanf("%f %f", &potion1, &potion2);
    printf("You mixed %.2f liters of Potion 1 and %.2f liters of Potion 2.\n", potion1, potion2);
    return 0;
}
```

3. A blacksmith keeps track of his iron stock in kilograms. Take the iron weight as input and print it.

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

```
int main() {
    float ironStock;
    printf("Enter the iron stock (kg): ");
    scanf("%f", &ironStock);
    printf("The blacksmith has %.2f kg of iron.\n", ironStock);
    return 0;
}
```

**UE24CS151B – Problem Solving with C****Laboratory – Week 2**

4. A warrior lifts two weights. Calculate and print his total lifted weight.

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

```
int main() {
    float weight1, weight2, total;
    printf("Enter the two weights lifted: ");
    scanf("%f %f", &weight1, &weight2);
    total = weight1 + weight2;
    printf("Total lifted weight: %.2f kg\n", total);
    return 0;
}
```

5. The king collects tax as 10% of all trade profits. Take the profit amount and compute the tax.

Solution:

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

```
int main() {
    float profit, tax;
    printf("Enter the total trade profit: ");
    scanf("%f", &profit);
    tax = profit * 0.10;
    printf("The king's tax is: %.2f gold\n", tax);
    return 0;
}
```

6. A messenger covers a distance in a given time. Calculate his speed.

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

```
int main() {
    float distance, time, speed;
    printf("Enter distance (km) and time (hours): ");
    scanf("%f %f", &distance, &time);
    speed = distance / time;
    printf("The messenger's speed is %.2f km/h\n", speed);
    return 0;
}
```

7. An architect measures the height of a castle tower. Take the height as input and print it.

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

```
int main() {
    float height;
    printf("Enter the castle tower height (meters): ");
    scanf("%f", &height);
    printf("The castle tower is %.2f meters tall.\n", height);
    return 0;
}
```

8. A scientist has invented a time machine that allows a person to travel back in time. The user enters the current year and the number of years to go back. Your task is to write a C program that calculates and displays the past year where the person arrives after traveling.

Input :

2025 10

Output:

You have traveled back to the year 2015.

Solution:

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

```
int main() {
    int currentYear, yearsBack, pastYear;
```

```

printf("Enter current year and years to travel back: ");
scanf("%d %d", &currentYear, &yearsBack);
pastYear = currentYear - yearsBack;
printf("You have traveled back to the year %d.\n", pastYear);
return 0;
}

```

9. A spell consumes energy equal to power × time. Compute the total energy.

```

#include <stdio.h>

int main() {
    float power, time, energy;
    printf("Enter power (watts) and time (seconds): ");
    scanf("%f %f", &power, &time);
    energy = power * time;
    printf("Total spell energy used: %.2f Joules\n", energy);
    return 0;
}

```



Laboratory – Week 2

10. A farmer measures the area of his land (length × width). Compute and print the total land area.

Solution:

```
#include <stdio.h>

int main() {
    float length, width, area;
    printf("Enter the land dimensions (length and width in meters): ");
    scanf("%f %f", &length, &width);
    area = length * width;
    printf("Total land area: %.2f square meters\n", area);
    return 0;
}
```



11. A stone guardian asks for your favorite letter before letting you pass. Take a character input and print it.

Solution:

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

```
int main() {
    char favLetter;
    printf("Enter your favorite letter: ");
    scanf(" %c", &favLetter);
    printf("The guardian nods. '%c' is a fine choice.\n", favLetter);
    return 0;
}
```



12. Celsius to Fahrenheit Conversion: Write a program that converts a given Celsius temperature to Fahrenheit.

Solution:

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * 9 / 5) + 32;
    printf("Fahrenheit = %.2f", fahrenheit);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

13. A wizard tells you that his age is the average of three ancient numbers. Write a C program to take three integers and compute their average.

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

```
int main() {
    int num1, num2, num3;
    float average;
    printf("Enter three ancient numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    average = (num1 + num2 + num3) / 3.0;
    printf("The wizard's age is %.2f years.\n", average);
    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 2

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programs and Document Prepared by: Prof. Pranjali Thakre

Acknowledge Prof. Sowmyashree as the source of this information



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

Commands to execute Programs using gcc:

1. Editing

```
$ gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 1. You are about to start an adventure. The first step is to introduce yourself. Write a C program that asks for your name's first letter and prints a welcome message.**

Solution:

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

```
int main() {
    char name;
    printf("Enter the first letter of your name: ");
    scanf(" %c", &name);
    printf("Welcome, brave traveler %c!\n", name);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 2. A wise sage asks for your age before guiding you. Write a C program to take the age as input and print it.**Solution:

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

```
int main() {
    int age;
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("You are %d years old. The adventure awaits you!\n", age);
    return 0;
}
```

- 3. You find a magical door that opens only if you enter a secret number. Take an integer input and display it as the entered password.**

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

```
int main() {
    int password;
    printf("Enter the magic door password: ");
    scanf("%d", &password);
    printf("You entered: %d. The door begins to open...\n", password);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 4. A potion seller offers two potions, but first, you must record their prices. Write a C program that takes two floating-point numbers as input and prints them.**

Solution:

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

```
int main() {
    float price1, price2;
    printf("Enter the prices of two potions: ");
    scanf("%f %f", &price1, &price2);
    printf("Potion 1 costs %.2f gold, Potion 2 costs %.2f gold.\n", price1, price2);
    return 0;
}
```

- 5. A warrior must prove his strength. Enter two numbers and find their sum to determine his strength level.**

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

```
int main() {
    int power1, power2, total;
    printf("Enter two power levels: ");
    scanf("%d %d", &power1, &power2);
    total = power1 + power2;
    printf("Total strength level: %d\n", total);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 6. A scientist has invented a time machine that allows a person to travel back in time. The user enters the current year and the number of years to go back. Your task is to write a C program that calculates and displays the past year where the person arrives after traveling.**

Input :

2025 10

Output:

You have traveled back to the year 2015.

Solution:

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

```
int main() {
    int currentYear, yearsBack, pastYear;
    printf("Enter current year and years to travel back: ");
    scanf("%d %d", &currentYear, &yearsBack);
    pastYear = currentYear - yearsBack;
    printf("You have traveled back to the year %d.\n", pastYear);
    return 0;
}
```



- 7. A stone guardian asks for your favorite letter before letting you pass. Take a character input and print it.**

Solution:

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

```
int main() {
    char favLetter;
    printf("Enter your favorite letter: ");
    scanf(" %c", &favLetter);
    printf("The guardian nods. '%c' is a fine choice.\n", favLetter);
    return 0;
}
```



- 8. Celsius to Fahrenheit Conversion:** Write a program that converts a given Celsius temperature to Fahrenheit.

Solution:

```
#include <stdio.h>

int main() {
    float celsius, fahrenheit;
    printf("Enter temperature in Celsius: ");
    scanf("%f", &celsius);
    fahrenheit = (celsius * 9 / 5) + 32;
    printf("Fahrenheit = %.2f", fahrenheit);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 9. A wizard tells you that his age is the average of three ancient numbers. Write a C program to take three integers and compute their average.**

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

```
int main() {
    int num1, num2, num3;
    float average;
    printf("Enter three ancient numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);
    average = (num1 + num2 + num3) / 3.0;
    printf("The wizard's age is %.2f years.\n", average);
    return 0;
}
```

- 10. The village tax collector calculates tax as 5% of the gold collected. Take the gold amount as input and compute the tax amount.**

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

```
int main() {
    float gold, tax;
    printf("Enter the amount of collected gold: ");
    scanf("%f", &gold);
    tax = gold * 0.05;
    printf("The tax to be paid is: %.2f gold\n", tax);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

11. Simple Interest Calculation: Write a program that takes principal, rate, and time as input and calculates simple interest.

Solution:

```
#include <stdio.h>

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

    printf("Enter principal amount: ");
    scanf("%f", &principal);

    printf("Enter rate of interest: ");
    scanf("%f", &rate);

    printf("Enter time in years: ");
    scanf("%f", &time);

    interest = (principal * rate * time) / 100;
    printf("Simple Interest: %.2f\n", interest);
    return 0;
}
```



Department of CSE, PES University

UE24CS151B – Problem Solving with C

Laboratory – Week 2

**12. A messenger runs from one village to another in a given distance and time.
Compute and print his speed.**

```
#include <stdio.h>

int main() {
    float distance, time, speed;
    printf("Enter the distance traveled (km): ");
    scanf("%f", &distance);
    printf("Enter the time taken (hours): ");
    scanf("%f", &time);
    speed = distance / time;
    printf("Messenger's speed: %.2f km/h\n", speed);
    return 0;
}
```



UE24CS151B – Problem Solving with C

Laboratory – Week 2

- 13. A treasure is buried X meters to the east and Y meters to the north from the starting position (0, 0). Write a C program that takes the X and Y coordinates as input and calculates the distance to the treasure using the distance formula:**

$$\text{Distance} = \sqrt{X^2 + Y^2}$$

Solution:

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

int main() {
    int x, y;
    double distance;

    printf("Enter the treasure's X and Y coordinates: ");
    scanf("%d %d", &x, &y);

    distance = sqrt(x * x + y * y);
    printf("The treasure is %.2f meters away.\n", distance);

    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 7

Semester: 2

Course Code: UE24CS151B

Lab Anchor: Dr. Mohan Kumar AV

Session: Feb 2025 – June 2025

Programmed and Documented by: Dr. Mohan Kumar AV

Heartfelt thanks to **Prof. Sowmyashree**, Assistant Professor, Dept. of CSE, for providing this valuable source of information.



Department of CSE, PES University
UE24CS151B – Problem Solving with C
Laboratory-Week 7

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

To learn and solve	Debugging with GDB.
--------------------	---------------------

Common for all sections: P Cycle, C Cycle (8.45AM-4.45PM)

GDB, the GNU Debugger, is a powerful tool that allows developers to inspect and control the execution of programs written in various programming languages, including Ada, Assembly, C, C++, D, Fortran, Go, Objective-C, OpenCL, Modula-2, Pascal, and Rust. It enables users to monitor program execution, set breakpoints, and analyze crashes to understand program behavior and identify issues.

Originally authored by Richard Stallman, GDB is part of the GNU Project and has seen contributions from numerous developers over the years. It is compatible with various operating systems, including Unix-like systems, Microsoft Windows, and macOS.

As of February 1, 2025, the latest version of GDB is 16.2, which includes several enhancements and bug fixes.

Tasks Supported by GDB

GDB provides powerful debugging capabilities to help identify and resolve bugs in your program. The key tasks supported by GDB include:

- ✓ **Starting the Program:** Run your program with specified arguments, environment variables, and execution conditions that may affect its behavior.
- ✓ **Controlling Execution:** Set breakpoints, watchpoints, and conditions to pause execution at specific points.
- ✓ **Inspecting Program State:** Analyze variables, memory, call stack, and registers when the program is paused to understand its behavior.
- ✓ **Modifying Execution:** Alter variable values, memory, or execution flow to test fixes and experiment with different scenarios.

Getting In and Out of GDB

1) Starting GDB (GNU Debugger)

On Windows:

1. Press **Win + R**, type **cmd**, and press Enter to open the Command Prompt.
2. Type **gdb** and press Enter to start GDB.

On Linux:

1. Press **Ctrl + Alt + T** to open the terminal, or search for "**Terminal**" in the applications menu and open it.
2. Type **gdb** and press Enter to start GDB.

```
Command Prompt - gdb
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.

C:\Users\DELL>gdb
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb)
```

Gdb open prompt lets you know that it is ready for commands.

2) Steps to Exit GDB (GNU Debugger)

1. Type **quit** or **q** in the GDB command prompt.
2. Press the Enter key.

Alternative Method:

- Press **Ctrl + D** to exit GDB directly.

```
C:\Users\DELL>gdb
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) q
C:\Users\DELL>
```

Invoking GDB (GNU Debugger)

GDB can be started with different arguments and options to set up the debugging environment. Below are the common ways to invoke GDB with simple explanations and examples.

1. Running GDB with an Executable File

The most common way to start GDB is by specifying the executable filename you want to debug.

Syntax:

`gdb filename`

Example:

`gdb a.exe # Starts GDB and loads the executable "a.exe"`

2. Running GDB with an Executable and a Core Dump File

A core dump file is created when a program crashes. You can use GDB to analyze it.

Syntax:

gdb filename corefile

Example:

gdb a.exe core # Loads "a.exe" and analyzes the core dump file

3. Debugging a Running Process by Process ID (Only for Linux)

If a program is already running, you can attach GDB to it using its process ID (PID).

Syntax:

gdb filename PID

or

gdb -p PID

Examples:

gdb a.exe 1234 # Attaches GDB to process 1234 running "a.exe"

gdb -p 1234 # Attaches GDB to process 1234 (no need to specify filename)

Note: Suppose **1234 is the Process ID (PID) of a running instance of a.exe**. Attaching GDB to this process allows live debugging without restarting the program

4. Running GDB Without Startup Messages

By default, GDB prints introductory messages when started. You can disable them using the --silent, -q, or --quiet options.

Syntax:

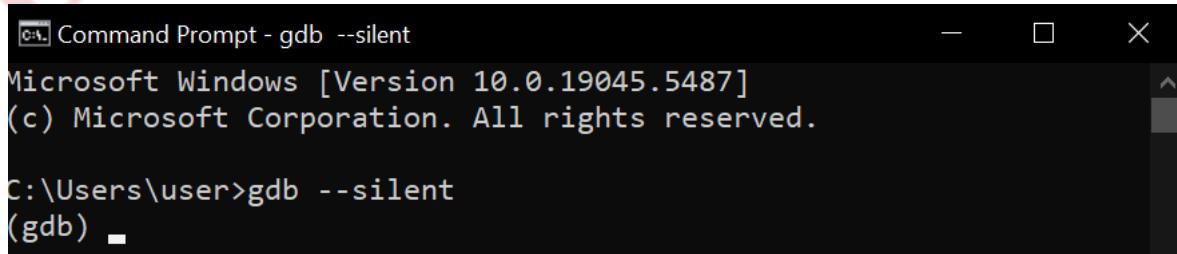
gdb --silent

gdb -q

gdb --quiet

Example:

gdb -q a.exe # Starts GDB with "a.exe" without displaying startup messages



Command Prompt - gdb --silent

Microsoft Windows [Version 10.0.19045.5487]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user>gdb --silent
(gdb) -

Usage of GDB on C code

A. Problem to be solved: Program to add two numbers that shows undefined behavior **gdb1.c**

Step 1: Write the Program (gdb1.c)

Create a new file gdb1.c and add the following code:

```
#include <stdio.h>
int main()
{
    int c;
    int a;
    int b;
    c = a + b; // Undefined behavior: as 'a' and 'b' are uninitialized
    printf("%d\n", c);
    return 0;
}
```

Save the file.

Step 2: Compiling with Debugging Information

To debug with GDB, compile the program with options as below:

```
gcc -g gdb1.c -o gdb1
```

Note: -g is used for debugging, while -o is used for specifying the output file gdb1 instead of default a.out. This generates an executable file named **gdb1** that contains debugging information.

Step 3: Running GDB on the Program

Start GDB with the compiled program:

```
gdb gdb1
```

Once inside GDB, you will see a prompt like:

```
(gdb)
```

Step 4: Setting Breakpoints

Breakpoints allow you to stop execution at a specific line to inspect values.

1. Set a breakpoint at the line where $c = a + b;$ is executed:

```
(gdb) break gdb1.c:7
```
2. Confirm the breakpoint:

```
(gdb) info breakpoints
```

Step 5: Running the Program Under GDB

Start execution:

```
(gdb) run
```

If execution stops at the breakpoint, you should see output like:

Breakpoint 1, main () at gdb1.c:7

7 $c = a + b;$

Step 6: Inspecting Variables

- Check the values of a and b:

(gdb) print a

(gdb) print b

Since they are uninitialized, you may see **garbage values**.

- Check the memory location of a and b:

(gdb) x/wx &a

(gdb) x/wx &b

Step to the next line using:

(gdb) next

Then print c:

(gdb) print c

c will also have an **unpredictable value**.

Note: 1. x/wx &a → Examines memory at the address of a (x = examine, /w = word size, x = hexadecimal format).

2. x/wx &b → Examines memory at the address of b (x = examine, /w = word size, x = hexadecimal format).

Step 7: set print null-stop feature to detect uninitialized values.

Enable warnings for uninitialized values.

(gdb) set print null-stop on

This makes GDB warn if an uninitialized variable is being used

Step to the next line and check c:

(gdb) next

(gdb) print c

This will show an unexpected value due to uninitialized **a** and **b**.

Step 8: Fixing the Issue

Modify the code to initialize a and b:

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

```
int main()
```

```
{
```

```
    int a = 5;
```

```
    int b = 10;
```

```
    int c = a + b;
```

```
    printf("%d\n", c);
```

```
    return 0;
```

```
}
```

Recompile and debug again:

gcc -g gdb1.c -o gdb1

gdb gdb1

Now, when running inside GDB, a, b, and c will have defined values.

Summary of list of actions with commands

Step	Action	Command
1	Write and save the faulty program (gdb1.c)	Create a file and add the given code
2	Compile with debugging information	gcc -g gdb1.c -o gdb1
3	Start GDB on the compiled program	gdb gdb1
4	Set breakpoints	break gdb1.c:7
	Confirm breakpoints	info breakpoints
5	Run the program inside GDB	run
	Check values of uninitialized variables	print a, print b
6	Inspect memory addresses of variables	x/wx &a, x/wx &b
	Step through execution	next
	Print value of c	print c (contains garbage value)
	Exit GDB	quit
7	set print null-stop to detect uninitialized values	set print null-stop on
8	Fix uninitialized variables	Initialize a and b before use
	Recompile the fixed program	gcc -g gdb1.c -o gdb1
	Debug again to verify fix	gdb gdb1

B. Write a C program that calculates the sum of elements in an array.

Step 1: Write the Buggy Program (uninitialized_array.c)

```
#include <stdio.h>
int sum_array(int arr[], int size);
int main()
{
    int arr[5]; // Uninitialized array
    int sum = sum_array(arr, 5);
    printf("Sum of array elements: %d\n", sum);
    return 0;
}
int sum_array(int arr[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++)
    {
        sum += arr[i]; // Using uninitialized values
    }
    return sum;
}
```

Bug: The array arr is declared but not initialized, so it contains garbage values.

Step 2: Compile with Debugging Symbols

Run the following command to compile the program with debugging information:

```
gcc -g uninitialized_array.c -o array_debug.exe
```

Step 3: Start Debugging with GDB

Run the debugger with the compiled program:

```
gdb array_debug.exe
```

Step 4: Set Breakpoints

Set breakpoints at key locations:

```
(gdb) break main      # Stop at main()
(gdb) break sum_array  # Stop at sum_array()
(gdb) break 13         # Stop inside the loop (adjust line number if needed)
```

Step 5: Run the Program in Debugger

Run the program inside GDB:

```
(gdb) run
```

The program will now wait for input.

Step 6: Step Through Execution & Inspect Values

1. Step into sum_array():

```
(gdb) step
```

2. Move inside the loop:

(gdb) next

3. Print variable values:

(gdb) print arr[0]

(gdb) print arr[1]

(gdb) print arr[2]

(gdb) print arr[3]

(gdb) print arr[4]

Problem: The values will be random garbage because the array was not initialized.

Step 7: Fix the Errors

Modify the program to initialize the array in main() before passing it to sum_array().

Fixed Code

```
#include <stdio.h>
int sum_array(int arr[], int size);
int main()
{
    int arr[5] = {1, 2, 3, 4, 5}; // Properly initialized array
    int sum = sum_array(arr, 5);
    printf("Sum of array elements: %d\n", sum);
    return 0;
}
int sum_array(int arr[], int size)
{
    int sum = 0;
    for (int i = 0; i < size; i++)
    {
        sum += arr[i];
    }
    return sum;
}
```

Step 8: Recompile & Debug Again

After fixing errors, recompile and rerun GDB:

```
gcc -g uninitialized_array.c -o array_debug.exe
gdb array_debug.exe
```

Step 9: Test Again

Run the program inside GDB:

(gdb) run

Expected Output:

Sum of array elements: 15

Now the program works correctly!

Summary of list of actions with commands

Step	Action	Command
1	Compile with debugging	gcc -g uninitialized_array.c -o array_debug.exe
2	Start GDB	gdb array_debug.exe
3	Set breakpoints	break main, break sum_array, break 17
4	Run the program	run
5	Step into function	step
6	Step through loop	next
7	Print variables	print arr[0], print arr[4]
8	Detect errors	Found uninitialized array (garbage values)
9	Fix errors	Initialize array in main()
10	Recompile & test	gcc -g ... & gdb array_debug.exe

GDB Commands

- **help** or **h**: Displays a list of command classes available in GDB.
- **help <class>**: Provides a list of individual commands within a specified command class.
 - **Example:** `help breakpoint` displays commands related to breakpoints.
- **help <command>**: Displays a brief description and usage details of a specific command.
 - **Example:** `help run` provides information on executing a program in GDB.
- **help all**: Displays details of all available commands across all command classes.

```
(gdb) help
List of classes of commands:

aliases -- Aliases of other commands
breakpoints -- Making program stop at certain points
data -- Examining data
files -- Specifying and examining files
internals -- Maintenance commands
obscure -- Obscure features
running -- Running the program
stack -- Examining the stack
status -- Status inquiries
support -- Support facilities
tracepoints -- Tracing of program execution without stopping the program
user-defined -- User-defined commands

Type "help" followed by a class name for a list of commands in that class.
Type "help all" for the list of all commands.
Type "help" followed by command name for full documentation.
Type "apropos word" to search for commands related to "word".
Command name abbreviations are allowed if unambiguous.
(gdb) help breakpoints
Making program stop at certain points.

List of commands:
```

Clearing the Screen in GDB

1. For Windows:

- Command:

(gdb) shell cls

- The shell command allows executing system commands from within GDB.
- `cls` is the Windows command to clear the terminal screen.

2. For Linux/macOS:

- Command:

(gdb) shell clear

- `shell` executes system commands from GDB.
- `clear` is the standard command to clear the terminal screen on Linux and macOS.

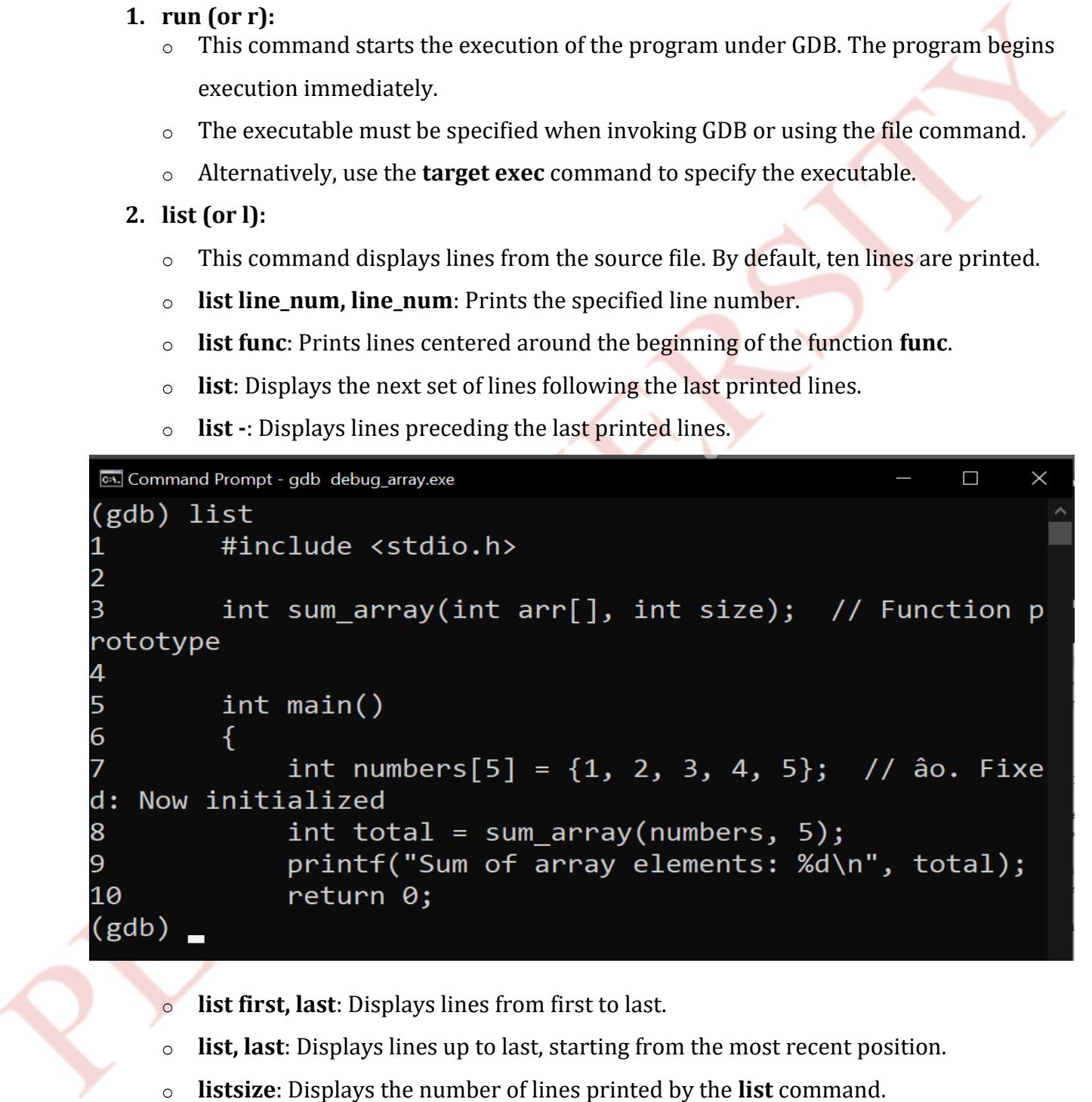
GDB Commands and Their Usage

1. run (or r):

- This command starts the execution of the program under GDB. The program begins execution immediately.
- The executable must be specified when invoking GDB or using the file command.
- Alternatively, use the **target exec** command to specify the executable.

2. list (or l):

- This command displays lines from the source file. By default, ten lines are printed.
- **list line_num, line_num**: Prints the specified line number.
- **list func**: Prints lines centered around the beginning of the function **func**.
- **list**: Displays the next set of lines following the last printed lines.
- **list -**: Displays lines preceding the last printed lines.



```
Command Prompt - gdb debug_array.exe
(gdb) list
1      #include <stdio.h>
2
3      int sum_array(int arr[], int size); // Function p
prototype
4
5      int main()
6      {
7          int numbers[5] = {1, 2, 3, 4, 5}; // ão. Fixe
d: Now initialized
8          int total = sum_array(numbers, 5);
9          printf("Sum of array elements: %d\n", total);
10         return 0;
(gdb) -
```

- **list first, last**: Displays lines from first to last.
- **list, last**: Displays lines up to last, starting from the most recent position.
- **listsize**: Displays the number of lines printed by the **list** command.
- **set listsize number**: Configures the list command to display a specified number of source lines. Setting the count to **unlimited** or **0** removes any restrictions on the number of displayed lines.

○

```
c:\ Command Prompt - gdb debug_array.exe
(gdb) set listsiz 15
(gdb) show listsiz
Number of source lines gdb will list by default is 15.
(gdb) l
11      }
12
13
14      int sum_array(int arr[], int size)
15  {
16      int sum = 0;
17      for (int i = 0; i < size; i++)
18      {
19          sum += arr[i]; // â?O Error: Using uninit
ialized array elements
20      }
21      return sum;
22  }
```

GDB 'print' Command Usage

The **print** (or **p**) command in GDB is used to evaluate and display the value of an expression written in the programming language of the debugged program. It provides an effective way to inspect variables, expressions, and memory contents during debugging.

- **p expression:** Displays the evaluated result of the specified expression.
- **p variable:** Prints the value of the given variable.
- **p function_name::variable:** When referring to a static variable within a specific function or file, the scope resolution operator (:) is used to specify the correct context.

This ensures that the appropriate variable is accessed when multiple variables with the same name exist in different scopes.

```
(gdb) p 1+2
$1 = 3
(gdb) p 1==1
$2 = 1
(gdb) p 1==2
$3 = 0
(gdb) p a
No symbol "a" in current context.
(gdb)
```

GDB Commands - Stopping and Continuing Execution

The GNU Debugger (GDB) allows users to control the execution of a program by setting breakpoints at specific locations or conditions. Below are the key commands related to stopping and continuing execution in GDB:

- **breakpoint**: A breakpoint instructs the program to halt execution when a specified location or condition is met.
- **break <line_number>**: Sets a breakpoint at the beginning of the specified line in the source code.
- **break <function_name>**: Sets a breakpoint at the beginning of the specified function.
- **break *<memory_address>**: Sets a breakpoint at the exact memory address of an instruction.
- **break <condition>**: Sets a conditional breakpoint. The condition is evaluated each time the breakpoint is reached, and execution stops only if the condition evaluates to a nonzero value.
- **break (without arguments)**: Sets a breakpoint at the next instruction to be executed in the currently selected stack frame.

```
C:\ Command Prompt - gdb debug_array.exe
(gdb) break 8
Breakpoint 1 at 0x401496: file debug_array.c, line 8.
(gdb) break 13
Breakpoint 2 at 0x4014cf: file debug_array.c, line 13.
(gdb) info b
Num      Type            Disp Enb Address      What
1        breakpoint      keep y   0x00401496 in main
                                         at debug_array.
c:8
2        breakpoint      keep y   0x004014cf in sum_array
                                         at debug_array.
c:13
(gdb) run
Starting program: C:\Users\user/debug_array.exe
[New Thread 8660.0x3794]
[New Thread 8660.0x914]

Breakpoint 1, main () at debug_array.c:8
8          int total = sum_array(numbers, 5);
(gdb) continue
Continuing.

Breakpoint 2, sum_array (arr=0x61ff08, size=5)
     at debug_array.c:16
16          int sum = 0;
(gdb) ■
```

Watchpoint: A specialized breakpoint that halts program execution whenever the value of a specified expression changes. Unlike traditional breakpoints, watchpoints do not require specifying a particular line of code where the change might occur. Instead, they monitor memory locations associated with the given expression. When the expression is modified, the debugger interrupts execution and displays both the previous and updated values. The expression being monitored can range from a single variable to a more complex combination of multiple variables and operators.

Syntax:

watch expression

```
Command Prompt - gdb debug_array.exe
(gdb) n
17          for (int i = 0; i < size; i++)
(gdb) watch i
Hardware watchpoint 3: i
(gdb) n
Hardware watchpoint 3: i

Old value = 0
New value = 1
0x004014f7 in sum_array (arr=0x61ff08, size=5)
    at debug_array.c:17
17          for (int i = 0; i < size; i++)
(gdb) n
19          sum += arr[i]; // â?O Error: Using
uninitialized array elements
(gdb) -
```

- **Catchpoint:** A specialized breakpoint that halts program execution when a specific event occurs, such as an exception.
- **Delete:** Individual breakpoints, including catchpoints, can be removed by specifying their respective breakpoint numbers.

```
Command Prompt - gdb debug_array.exe
(gdb) info b
Num      Type            Disp Enb Address      What
1        breakpoint      keep y  0x00401496 in main
                                         at debug_
array.c:8
                                         breakpoint already hit 1 time
3        hw watchpoint   keep y
                                         breakpoint already hit 1 time
(gdb) delete 3
(gdb) info b
Num      Type            Disp Enb Address      What
1        breakpoint      keep y  0x00401496 in main
                                         at debug_
array.c:8
                                         breakpoint already hit 1 time
(gdb) delete
Delete all breakpoints? (y or n) y
(gdb) info b
No breakpoints or watchpoints.
(gdb) -
```

clear and disable commands in GDB

1. **clear** – Removes any breakpoints set at the current execution point in the selected stack frame.
2. **clear line_num** – Deletes all breakpoints set at the specified line number within the current source file.
3. **clear file_name** – Removes breakpoints set at the beginning of any function defined in the specified source file.
4. **clear function_name** – Deletes breakpoints at the entry point of the specified function.

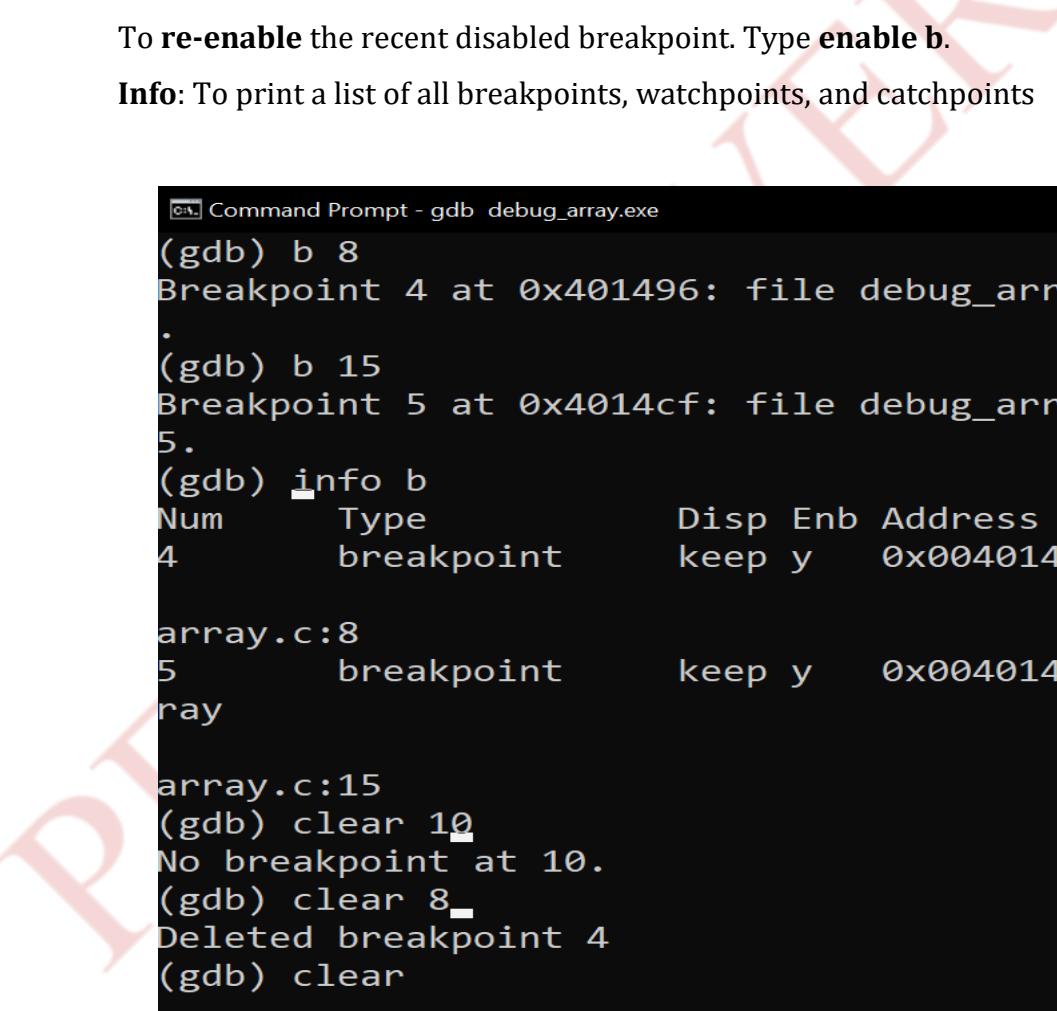
```
(gdb) b 8
Breakpoint 10 at 0x401492: file 1_find_factors_sum.c, line 8.
(gdb) b 15
Breakpoint 11 at 0x4014cb: file 1_find_factors_sum.c, line 15.
(gdb) info b
Num      Type            Disp Enb Address    What
10      breakpoint      keep y  0x00401492 in main at 1_find_factors_sum.c:8
11      breakpoint      keep y  0x004014cb in sum_factors at 1_find_factors_sum.c:15
(gdb) clear 10
No breakpoint at 10.
(gdb) clear 8
Deleted breakpoint 10
(gdb) clear
Deleted breakpoint 11
(gdb)
```

disable: Makes the breakpoint inoperative, but remembers the information on the breakpoint so that it can be enabled later using enable.

```
(gdb) b 8
Breakpoint 1 at 0x401492: file 1_find_factors_sum.c, line 8.
(gdb) b 15
Breakpoint 2 at 0x4014cb: file 1_find_factors_sum.c, line 15.
(gdb) info b
Num      Type            Disp Enb Address      What
1        breakpoint      keep y  0x00401492 in main at 1_find_factors_sum.c:8
2        breakpoint      keep y  0x004014cb in sum_factors at 1_find_factors_sum.c:15
(gdb) disable b 1
(gdb) info b
Num      Type            Disp Enb Address      What
1        breakpoint      keep n  0x00401492 in main at 1_find_factors_sum.c:8
2        breakpoint      keep y  0x004014cb in sum_factors at 1_find_factors_sum.c:15
(gdb)
```

To **re-enable** the recent disabled breakpoint. Type **enable b**.

Info: To print a list of all breakpoints, watchpoints, and catchpoints



```
Command Prompt - gdb debug_array.exe
(gdb) b 8
Breakpoint 4 at 0x401496: file debug_array.c, line 8
.
(gdb) b 15
Breakpoint 5 at 0x4014cf: file debug_array.c, line 15.
(gdb) info b
Num      Type            Disp Enb Address      What
4        breakpoint      keep y  0x00401496 in main
                                at debug_
array.c:8
5        breakpoint      keep y  0x004014cf in sum_ar
ray
                                at debug_
array.c:15
(gdb) clear 10
No breakpoint at 10.
(gdb) clear 8_
Deleted breakpoint 4
(gdb) clear
```

More GDB Commands

file command

When using GDB (GNU Debugger), the file command is used to load an executable file into the debugger. This helps GDB:

- Read **debugging symbols** (if available).
- Load the **executable instructions** of the program.
- Allow us to start running and debugging the program using the **run** command.

If we want to debug a different program during the same GDB session, we can use file again to switch to another executable.

Example

Step 1: Create a Simple C Program (example.c)

```
#include <stdio.h>
int main()
{
    printf("Hello, GDB!\n");
    return 0;
}
```

Step 2: Compile with Debugging Information (-g option)

```
gcc -g example.c -o example
```

Here, **-g** tells the compiler to include debugging symbols, and **-o** example creates an executable file named **example**.

Step 3: Start GDB and Load the File

```
gdb
```

Once inside GDB, use the **file** command:

```
(gdb) file example
```

Output:

Reading symbols from D:\PESU\Week-7 Lab\example.exe...done.

Now, GDB has loaded the symbol table and executable contents of **example**.

Step 4: Run the Program in GDB

```
(gdb) run
```

Output:

Starting program: /path/to/example

Hello, GDB!

[Inferior 1 (process 12345) exited normally]

Here, run executes the loaded program, and we see **Hello, GDB!** printed.

Switching to Another File

If we want to debug another executable (**new_program**), we use:

(gdb) file new_program

This replaces the currently loaded file with **new_program**.

```
C:\ Command Prompt - gdb
C:\Users\user>gdb
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.

There is NO WARRANTY, to the extent permitted by law. Type "show
copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
(gdb) run
Starting program:
No executable specified, use `target exec'.
(gdb) target exec debug_array.exe
(gdb) run
Starting program: C:\Users\user\debug_array.exe
[New Thread 7084.0x2164]
[New Thread 7084.0x293c]
Sum of array elements: 15
```

The **next** (or **n**) command in GDB moves to the next line of code without stepping into functions. If the line contains a function call, GDB will run the entire function and stop at the next line after the function call.

For example: open a notepad greet.c and type the program below:

```
#include <stdio.h>

void greet() {
    printf("Hello!\n");
}

int main() {
    printf("Start\n"); // Line 1
    greet(); // Line 2 (Function call)
    printf("End\n"); // Line 3
    return 0;
}
```

If you use next:

- It will stop at Line 1 → next → stops at Line 2 → next → stops at Line 3
- It does not step inside greet(), it just executes it and moves forward.

Step 2: Compile with Debugging Symbols

Run this command in your terminal or command prompt:

gcc -g greet.c -o greet

Step 3: Start GDB

Launch GDB with the compiled program:

gdb greet.exe or **gdb greet** (in linux)

Step 4: Set Breakpoint at main()

(gdb) break main

This ensures that the program stops at the start of main().

Step 5: Run the Program

(gdb) run

The program stops at main() before executing any statements.

Step 6: Use next (n) Command

1. Execute the first line:

(gdb) next

It executes printf("Start\n"); and moves to greet();.

2. Execute the second line:

(gdb) next

It does not step into greet(). Instead, it just executes the function and moves to printf("End\n");.

3. Execute the third line:

(gdb) next

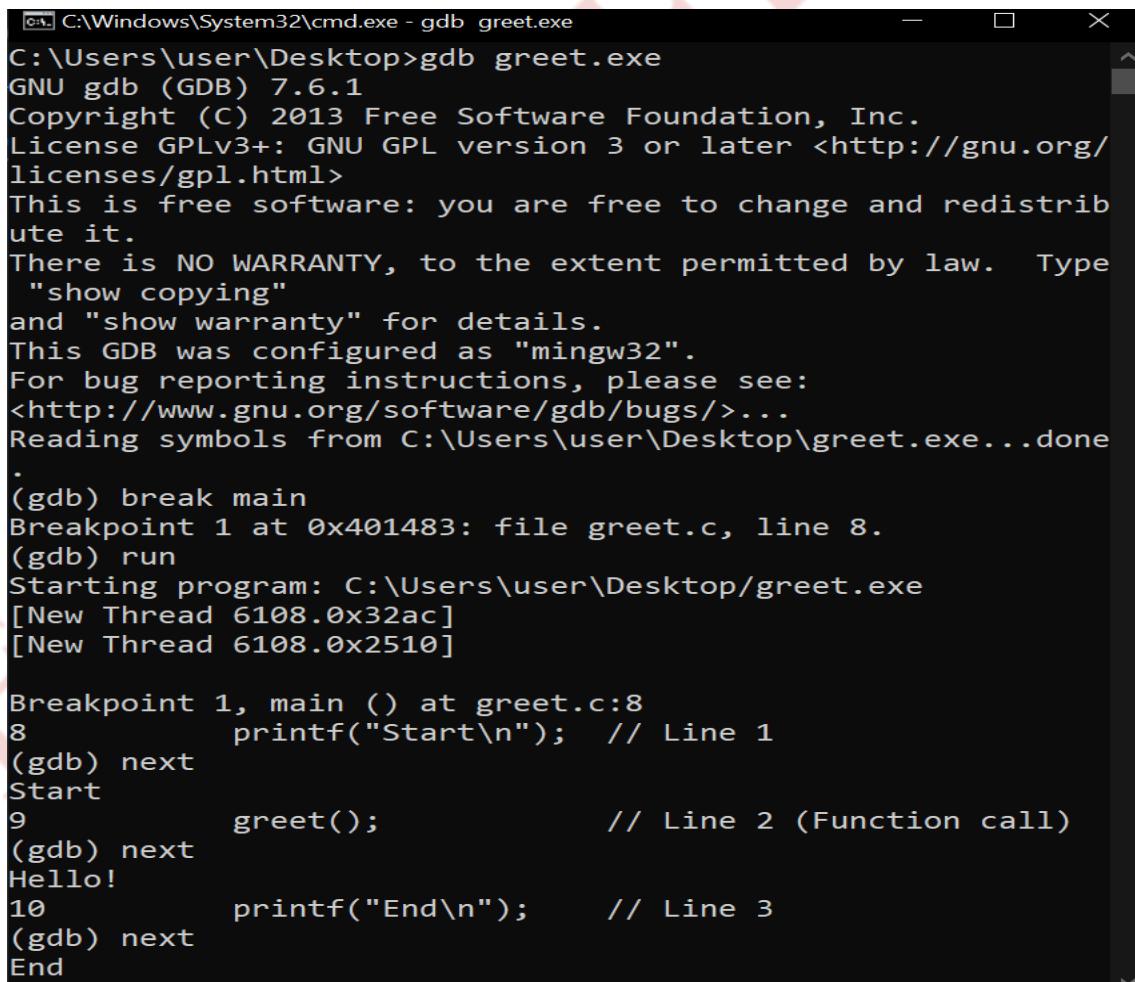
It prints "End" and moves to return 0;

4. To Step Inside greet()

If you want to step inside the greet() function instead of skipping it, use:

(gdb) step

This will move inside greet() and let you debug it line by line.



C:\Windows\System32\cmd.exe - gdb greet.exe
C:\Users\user\Desktop>gdb greet.exe
GNU gdb (GDB) 7.6.1
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <<http://gnu.org/licenses/gpl.html>>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type
"show copying"
and "show warranty" for details.
This GDB was configured as "mingw32".
For bug reporting instructions, please see:
<<http://www.gnu.org/software/gdb/bugs/>>...
Reading symbols from C:\Users\user\Desktop\greet.exe...done
. (gdb) break main
Breakpoint 1 at 0x401483: file greet.c, line 8.
(gdb) run
Starting program: C:\Users\user\Desktop\greet.exe
[New Thread 6108.0x32ac]
[New Thread 6108.0x2510]

Breakpoint 1, main () at greet.c:8
8 printf("Start\n"); // Line 1
(gdb) next
Start
9 greet(); // Line 2 (Function call)
(gdb) next
Hello!
10 printf("End\n"); // Line 3
(gdb) next
End

1. **step (s)** – Moves to the next line of code and enters inside any function calls if there are any.
 - o **Example:** If you have a function call like **sum(a, b);**, using **step** will take you inside the **sum** function to debug it line by line.
2. **continue (c)** – Resumes program execution from the current stopping point and runs until the next breakpoint (or program completion).
 - o **Example:** If your program is paused at a breakpoint, using **continue** will make it run again until it reaches another breakpoint or finishes execution.
3. **what is** – Tells you the type of a variable or expression.
 - o **Example:** If you type **what is a** in GDB, it will return something like **int** if **a** is an integer.
 - o If used without arguments, it tells you the type of the last computed value.
4. **set var** – Changes the value of a variable while debugging.
 - o **Example:** If **a = 5** in your program, you can change it to **10** by typing **set var a = 10** in GDB. This helps test different values without modifying your actual code.

```
(gdb) p a
$1 = 4194432
(gdb) p b
$2 = 2793472
(gdb) p c
$3 = 0
(gdb) set var a 10
A syntax error in expression, near `10'.
(gdb) whatis a
type = int
(gdb) set var a=10
(gdb) p a
$4 = 10
(gdb) set var b=10
(gdb) p b
$5 = 10
(gdb) p c
$6 = 0
(gdb) run
```

Backtrace, where and finish commands

Consider C Program (example.c)

```
#include <stdio.h>

void functionC() {
    int *ptr = NULL; // NULL pointer
    *ptr = 10; // Causes segmentation fault
}

void functionB() { functionC(); }

void functionA() { functionB(); }

int main() { functionA(); return 0; }
```

Compile with debugging support:

```
gcc -g example.c -o example
```

1. backtrace (or bt) – Find Crash Location

Scenario: The program crashes, and you want to see how.

```
gdb example
```

```
(gdb) run
```

Program crashes with Segmentation fault in functionC()

```
(gdb) backtrace
```

Output:

```
#0 functionC() at example.c:6
#1 functionB() at example.c:10
#2 functionA() at example.c:14
#3 main() at example.c:18
```

Tells us exactly where the crash happened and how you got there.

2. where – Check Current Execution

Scenario: The program hasn't crashed, but you want to see where it is.

```
(gdb) break functionB # Set a breakpoint
```

```
(gdb) run      # Start program
```

```
(gdb) where     # Check call stack
```

Output:

#0 functionB() at example.c:10

#1 functionA() at example.c:14

#2 main() at example.c:18

Shows where the program is currently executing.

3. finish – Skip to Function Exit

Scenario: You're inside a function and don't want to step through every line.

(gdb) break functionB

(gdb) run

(gdb) step # Step into functionB

(gdb) finish # Skip to end of functionB

Output:

Run till exit from functionB()

Returning to functionA() at example.c:14

Jumps to the function exit, saving time!



Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write functions to find the Greatest Common Divisor (GCD) and the Least Common Multiple (LCM) of two numbers. You can use the Euclidean algorithm to find the GCD.

```
#include <stdio.h>

// Function to find GCD using Euclidean algorithm

int gcd(int a, int b) {

    // Euclidean algorithm

    while (b != 0) {

        int temp = b;

        b = a % b;

        a = temp;

    }

    return a;
}

// Function to find LCM

int lcm(int a, int b) {

    return (a * b) / gcd(a, b); // LCM formula: (a * b) / GCD(a, b)
}

int main() {

    int num1, num2;

    // Taking input from the user

    printf("Enter two numbers: ");

    scanf("%d %d", &num1, &num2);

    // Calculate GCD

    int gcd_result = gcd(num1, num2);

    // Calculate LCM

    int lcm_result = lcm(num1, num2);

    // Output the results

    printf("GCD of %d and %d is: %d\n", num1, num2, gcd_result);

    printf("LCM of %d and %d is: %d\n", num1, num2, lcm_result);

    return 0;
}
```

2. Write a C program that defines a function void reverseArray(int arr[], int size) to reverse an array. The function should modify the array in place. Take an array as input from the user and display the reversed array.

```
#include <stdio.h>

void reverseArray(int arr[], int size) {

    int i, temp;

    for (i = 0; i < size / 2; i++) {

        temp = arr[i];

        arr[i] = arr[size - i - 1];

        arr[size - i - 1] = temp;

    }

}

int main() {

    int n, i;

    printf("Enter size of array: ");

    scanf("%d", &n);

    int arr[n];

    printf("Enter %d elements: ", n);

    for (i = 0; i < n; i++)

        scanf("%d", &arr[i]);

    reverseArray(arr, n);

    printf("Reversed array: ");

    for (i = 0; i < n; i++)

        printf("%d ", arr[i]);

    printf("\n");

    return 0;
```

- 3. Write a C program that swaps two numbers using pointers. Define a function void swap(int *a, int *b), which swaps the values of two integer variables using pointers.**

```
#include <stdio.h>

void swap(int *a, int *b) {

    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {

    int x, y;
    printf("Enter two numbers: ");
    scanf("%d %d", &x, &y);

    printf("Before swap: x = %d, y = %d\n", x, y);
    swap(&x, &y);
    printf("After swap: x = %d, y = %d\n", x, y);

    return 0;
}
```

- 4. Write a C program that finds the largest element in an array using a function and pointers. Define a function int findMax(int *arr, int size), which returns the maximum value in the given integer array.**

```
#include <stdio.h>

int findMax(int *arr, int size) {
    int max = *arr; // Initialize max with the first element
    for (int i = 1; i < size; i++) {
        if (*(arr + i) > max)
            max = *(arr + i);
    }
    return max;
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Maximum element: %d\n", findMax(arr, n));
}

return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Jan 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre



Department of CSE, PES University
UE23CS151B - Problem Solving with C

Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

1. Write a function in C to convert a decimal number to a binary number.

Expected output when the test code is run is as below:

Input any decimal

number: 8 The Binary value is : 1000

Input any decimal number: -8 The Binary value is : -1000

```
#include <stdio.h>

// Function to convert decimal to binary
void decimalToBinary(int n) {
    if (n < 0) {
        printf("-");
        n = -n; // Make the number positive for binary conversion
    }

    int binary[32]; // Array to store binary number (assuming 32-bit
    integers)
    int index = 0;

    // Special case for 0
    if (n == 0) {
        printf("0");
        return;
    }

    // Convert decimal to binary by dividing by 2
    while (n > 0) {
        binary[index++] = n % 2;
        n = n / 2;
    }

    // Print binary number from array in reverse order
    for (int i = index - 1; i >= 0; i--) {
        printf("%d", binary[i]);
    }
}

int main() {
    int num;

    // Get input from the user
    printf("Input any decimal number: ");
    scanf("%d", &num);

    // Call function to print binary value
    printf("The Binary value is : ");
    decimalToBinary(num);

    return 0;
}
```

-
- 2. Write a C program that defines a function void countEvenOdd(int arr[], int size, int *evenCount, int *oddCount) to count even and odd numbers in an array using pointers.Solution:**

```
#include <stdio.h>

// Function to count even and odd numbers in an array
void countEvenOdd(int arr[], int size, int *evenCount, int *oddCount)
{
    *evenCount = *oddCount = 0;
    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0)
            (*evenCount)++;
        else
            (*oddCount)++;
    }
}

int main() {
    int n, even, odd;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    countEvenOdd(arr, n, &even, &odd);
    printf("Even count: %d, Odd count: %d\n", even, odd);

    return 0;
}
```

-
- 3. Find the Largest Difference Between Two Adjacent Elements:** Write a function to find the largest absolute difference between two adjacent elements in an array. If the array size is less than 2, display an appropriate message.

Example:

Input: [3, 9, 1, 4, 7]

Output: 8 (the largest difference is between 1 and 9)

Solution:

```
#include <stdio.h>
#include <stdlib.h> // For the abs() function

// Function to find the largest absolute difference between two adjacent elements
void largestAdjacentDifference(int arr[], int size) {
    // If the array size is less than 2, display an appropriate message
    if (size < 2) {
        printf("Array size should be at least 2 to calculate the difference between adjacent
elements.\n");
        return;
    }

    int maxDiff = abs(arr[1] - arr[0]); // Initialize with the absolute difference between the first
two elements

    // Traverse the array and find the largest difference between adjacent elements
    for (int i = 1; i < size - 1; i++) {
        int currentDiff = abs(arr[i + 1] - arr[i]);
        if (currentDiff > maxDiff) {
            maxDiff = currentDiff;
        }
    }

    // Display the largest difference
    printf("The largest absolute difference between adjacent elements is: %d\n", maxDiff);
}

int main() {
    int arr[] = {3, 9, 1, 4, 7};
    int size = sizeof(arr) / sizeof(arr[0]);

    // Call the function to find the largest adjacent difference
    largestAdjacentDifference(arr, size);

    return 0;
}
```

-
- 4. Write a C program that defines a function int sumArray(int *arr, int size) to calculate the sum of elements in an array using pointer arithmetic.**

Solution:

```
#include <stdio.h>

// Function to find sum of elements in an array using pointer arithmetic
int sumArray(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i);
    }
    return sum;
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    printf("Sum of array elements: %d\n", sumArray(arr, n));
    return 0;
}
```



Department of CSE, PES University
UE23CS151B - Problem Solving with C



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Jan 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre



Instructions to Faculty

1. All faculty members shall be in the Lab on time.
2. Faculty members should not leave lab sessions unattended, when students are present, and Lab sessions should be engaged fully.
3. Faculty should explain all the lab programs during the lab hours only and make students execute all the lab programs given for that week.
4. Faculty should not share the faculty copy of the lab manual to students.
5. Insist students to use lab systems only and not their laptops.

Commands to execute Programs using gcc

1. Editing

```
$gedit filename.c
```

2. Compile

```
$ gcc -c filename.c
```

3. Linking

```
$ gcc filename.o
```

4. Loading - Execute

```
$ ./a.out (Linux Operating System)
```

```
$ a.exe (Windows Operating System)
```

-
- 1) Write a C program that defines a function int fibonacci(int n) to find the nth Fibonacci number.**

Solution:

```
#include <stdio.h>

// Iterative function to find the nth Fibonacci number
int fibonacci(int n) {
    if (n == 0) {
        return 0;
    } else if (n == 1) {
        return 1;
    }

    int a = 0, b = 1, c;
    for (int i = 2; i <= n; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return b;
}

int main() {
    int n;

    // Get input from the user
    printf("Enter the value of n to find the nth Fibonacci number: ");
    scanf("%d", &n);

    // Calling the fibonacci function
    int result = fibonacci(n);

    // Output the result
    printf("The %dth Fibonacci number is: %d\n", n, result);

    return 0;
}
```

-
- 2) Write a C program that defines a function void swap(int *a, int *b) to swap two numbers without using a temporary variable.**

Solution:

```
#include <stdio.h>

// Function to swap two numbers without using a temporary variable
void swap(int *a, int *b) {
    *a = *a + *b;
    *b = *a - *b;
    *a = *a - *b;
}

int main() {
    int x, y;
    printf("Enter two numbers: ");
    scanf("%d %d", &x, &y);

    printf("Before swap: x = %d, y = %d\n", x, y);
    swap(&x, &y);
    printf("After swap: x = %d, y = %d\n", x, y);

    return 0;
}
```

- 3) Write a C program that defines a function int secondLargest(int arr[], int size) to find the second largest number in an array.**

Solution:

```
#include <stdio.h>

// Function to find second largest number in an array
int secondLargest(int arr[], int size) {
    int largest = arr[0], second = -1;
    for (int i = 1; i < size; i++) {
        if (arr[i] > largest) {
            second = largest;
            largest = arr[i];
        } else if (arr[i] > second && arr[i] != largest) {
            second = arr[i];
        }
    }
    return second;
}

int main() {
    int n;
    printf("Enter size of array: ");
    scanf("%d", &n);
    int arr[n];

    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++)
        scanf("%d", &arr[i]);

    int result = secondLargest(arr, n);
    if (result == -1)
        printf("No second largest element found.\n");
    else
        printf("Second largest element: %d\n", result);

    return 0;
}
```

5. Write a function to find the sum of the anti-diagonal (secondary diagonal) elements of an $n \times n$ matrix. The anti-diagonal elements are elements at positions $(0, n-1), (1, n-2), \dots, (n-1, 0)$.

Example:

Input:

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 16

Output:

The sum of anti-diagonal elements is: $4 + 7 + 10 + 13 = 34$

Solution:

```
#include <stdio.h>

// Function to calculate the sum of the anti-diagonal (secondary diagonal) elements of
// an n×n matrix
int sumOfAntiDiagonal(int matrix[][4], int n) {
    int sum = 0;

    // Traverse the matrix and sum the anti-diagonal elements
    for (int i = 0; i < n; i++) {
        sum += matrix[i][n - 1 - i]; // Anti-diagonal elements are at positions (i, n-1-i)
    }

    return sum;
}

int main() {
    int n = 4;
    int matrix[4][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    // Call function to calculate the sum of anti-diagonal elements
    int result = sumOfAntiDiagonal(matrix, n);

    // Display the result
    printf("The sum of the anti-diagonal elements is: %d\n", result);

    return 0;
}
```




Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

Programs on functions, Array with functions, pointer

1. Find the Maximum Element in an Array

Write a C program to find the maximum element in a given array of size ‘n’. The array should be passed as a parameter to a user-defined function, and the function should return the maximum value.

Solution:

```
#include <stdio.h>

// Function to find the maximum element in an array

int find_max(int arr[], int n) {

    int max = arr[0]; // Assume the first element is the maximum

    for (int i = 1; i < n; i++) {

        if (arr[i] > max) {

            max = arr[i]; // Update max if a larger element is found
        }
    }

    return max; // Return the maximum value
}

int main() {

    int n;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &n);

    // Input the elements of the array

    int arr[n];

    printf("Enter %d elements: \n", n);
```

```

for (int i = 0; i < n; i++) {

    scanf("%d", &arr[i]);

}

// Call the function to find the maximum element and print the result

int max = find_max(arr, n);

printf("The maximum element in the array is: %d\n", max);

return 0;

}

```

2. Increment a Variable Using a Pointer

Write a function in C that takes a pointer to an integer as an argument and increments the value of the integer by 1. After calling the function, print the modified value in the main function.

Solution:

```

#include <stdio.h>

// Function to increment the value of the integer by 1 using a pointer

void increment_by_one(int *ptr) {

    (*ptr)++; // Increment the value pointed to by ptr

}

int main() {

    int num;

    // Input a value for the integer variable

    printf("Enter a number: ");

    scanf("%d", &num);

}

// Call the function and pass the address of 'num' using the pointer

increment_by_one(&num);

```

```

// Print the modified value

printf("The incremented value is: %d\n", num);

return 0;

}

```

3. Check for Armstrong Number

Write a function that takes an integer as an argument and returns whether the number is an Armstrong number or not. In the main function, print an appropriate message. (An Armstrong number is a number that is equal to the sum of its digits each raised to the power of the number of digits. Example: $153 = 1^3 + 5^3 + 3^3$)

```

#include <stdio.h>

#include <math.h>

// Function to check if the number is an Armstrong number

int is_armstrong(int num) {

    int original_num = num;

    int sum = 0;

    int digits = 0;

    // Count the number of digits in the number

    while (num != 0) {

        num /= 10;

        digits++;

    }

    num = original_num;

    // Calculate the sum of each digit raised to the power of the number of digits

    while (num != 0) {

        int digit = num % 10;

```

```
    sum += pow(digit, digits); // Use pow to raise the digit to the power of the
    number of digits

    num /= 10;

}

// Return 1 if the sum is equal to the original number, otherwise return 0

if (sum == original_num) {

    return 1;

} else {

    return 0;

}

int main() {

    int num;

    // Input a number from the user

    printf("Enter a number: ");

    scanf("%d", &num);

    // Check if the number is an Armstrong number

    if (is_armstrong(num)) {

        printf("%d is an Armstrong number.\n", num);

    } else {

        printf("%d is not an Armstrong number.\n", num);

    }

    return 0;

}
```

4. Updating Product Prices in a Store

You are developing an inventory system for a store where each product's price is stored in an array. The system allows updating the price of a specific product.

Question:

Write a C function that accepts an array of float values representing product prices and a specific product's ID (index). The function should increase the price of the specified product by a given percentage. Use a pointer to modify the original array.

Solution:

```
#include <stdio.h>

// Function to update the price of a product
void update_price(float *prices, int product_id, float percentage) {
    // Calculate the price increment
    prices[product_id] += prices[product_id] * (percentage / 100);
}

int main() {
    int n, product_id;
    float percentage;

    // Input number of products
    printf("Enter the number of products: ");
    scanf("%d", &n);

    // Create an array to store the prices of products
    float prices[n];

    // Input the prices of products
    printf("Enter the prices of %d products: \n", n);
```

```
for (int i = 0; i < n; i++) {  
    printf("Price of product %d: ", i + 1);  
    scanf("%f", &prices[i]);  
}  
  
// Input the product ID to update and the percentage increase  
printf("Enter the product ID to update (0 to %d): ", n - 1);  
scanf("%d", &product_id);  
  
printf("Enter the percentage increase: ");  
scanf("%f", &percentage);  
  
// Call the function to update the price  
update_price(prices, product_id, percentage);  
  
// Print the updated price of the product  
printf("Updated price of product %d: %.2f\n", product_id + 1,  
prices[product_id]);  
  
// Optionally print all product prices after the update  
printf("\nUpdated product prices:\n");  
for (int i = 0; i < n; i++) {  
    printf("Product %d: %.2f\n", i + 1, prices[i]);  
}  
  
return 0;  
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

Programs on functions, Array with functions, pointer

1. Sum of Even Numbers in an Array

Write a program in C to calculate the sum of all even numbers in a given array of size ‘n’. The array should be passed as a parameter to a user-defined function.

Solution:

```
#include <stdio.h>

// Function to calculate the sum of all even numbers in the array

int sum_of_even_numbers(int arr[], int n) {

    int sum = 0;

    // Iterate through the array and add the even numbers to sum

    for (int i = 0; i < n; i++) {

        if (arr[i] % 2 == 0) { // Check if the number is even

            sum += arr[i];

        }

    }

    return sum;
}

int main() {

    int n;

    // Input the size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &n);

    // Declare the array

    int arr[n];
```

```

// Input the elements of the array

printf("Enter %d elements: \n", n);

for (int i = 0; i < n; i++) {

    scanf("%d", &arr[i]);

}

// Call the function to calculate the sum of even numbers

int sum = sum_of_even_numbers(arr, n);

// Print the result

printf("The sum of all even numbers in the array is: %d\n", sum);

return 0;

}

```

2. Swapping Elements in an Array Using Pointers:

Write a function in C that accepts an array of integers and swaps two elements at given indices using pointers. The main function should print the modified array.

Solution:

```

#include <stdio.h>

// Function to swap two elements in the array using pointers

void swap_elements(int *arr, int index1, int index2) {

    // Swapping using pointers

    int temp = *(arr + index1); // Get the value at index1

    *(arr + index1) = *(arr + index2); // Set the value at index1 to the value at index2

    *(arr + index2) = temp; // Set the value at index2 to the original value of index1

}

```

```
int main() {  
    int n, index1, index2;  
  
    // Input the size of the array  
  
    printf("Enter the number of elements in the array: ");  
  
    scanf("%d", &n);  
  
    // Declare the array and input the elements  
  
    int arr[n];  
  
    printf("Enter %d elements:\n", n);  
  
    for (int i = 0; i < n; i++) {  
  
        scanf("%d", &arr[i]);  
  
    }  
  
    // Input the indices of the elements to swap  
  
    printf("Enter the indices to swap (0 to %d): ", n - 1);  
  
    scanf("%d %d", &index1, &index2);  
  
    // Check if indices are within bounds  
  
    if (index1 >= 0 && index1 < n && index2 >= 0 && index2 < n) {  
  
        // Call the function to swap elements  
  
        swap_elements(arr, index1, index2);  
  
        // Print the modified array  
  
        printf("Array after swapping elements at indices %d and %d:\n", index1, index2);  
  
        for (int i = 0; i < n; i++) {  
  
            printf("%d ", arr[i]);  
  
        }  
  
        printf("\n");  
  
    } else {
```

```
    printf("Invalid indices!\n");

}

return 0;

}
```

3. Check for Palindrome Number

Write a function that takes an integer as an argument and returns whether the number is a palindrome or not. In the main function, print an appropriate message.

Solution:

```
#include <stdio.h>

// Function to check if the number is a palindrome

int is_palindrome(int num) {

    int original_num = num; // Store the original number

    int reversed_num = 0;

    int remainder;

    // Reverse the number

    while (num != 0) {

        remainder = num % 10; // Get the last digit

        reversed_num = reversed_num * 10 + remainder; // Build the reversed number

        num /= 10; // Remove the last digit

    }

    // Check if the original number and the reversed number are the same

    if (original_num == reversed_num) {

        return 1; // Number is a palindrome

    } else {

        return 0; // Number is not a palindrome

    }

}
```

```

}

int main() {
    int num;

    // Input the number from the user
    printf("Enter a number: ");
    scanf("%d", &num);

    // Check if the number is a palindrome and print the appropriate message
    if (is_palindrome(num)) {
        printf("%d is a palindrome number.\n", num);
    } else {
        printf("%d is not a palindrome number.\n", num);
    }
    return 0;
}

```

4. Update Student Marks Using a 2D Array and Pointers

You are developing a student record management system where each student's marks are stored in an 2Darray. The system allows updating marks for a specific student.

Question:

Write a C function that accepts a 2D array of integers. Each row represents a student, and each column represents the marks in different subjects. The program will allow updating the marks of a specific student by adding extra marks (bonus) to the student's marks in all subjects..

Solution:

```
#include <stdio.h>

// Function to update the marks of a student by adding bonus marks
void update_marks(int (*marks)[5], int student_id, int bonus) {
    // Update the marks of the specified student by adding bonus marks to each subject
    for (int i = 0; i < 5; i++) {
```

```
    *(marks[student_id] + i) += bonus; // Use pointer arithmetic to modify marks
}

}

int main() {
    int n, subjects = 5, student_id, bonus;

    // Input the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Declare a 2D array for student marks
    int marks[n][5];

    // Input marks for each student in 5 subjects
    printf("Enter marks for %d students in 5 subjects:\n", n);
    for (int i = 0; i < n; i++) {
        printf("Enter marks for student %d:\n", i + 1);
        for (int j = 0; j < 5; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%d", &marks[i][j]);
        }
    }

    // Input the student ID and the bonus marks to be added
    printf("Enter the student ID to update (0 to %d): ", n - 1);
    scanf("%d", &student_id);
    printf("Enter the bonus marks to add: ");
}
```

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

// Call the function to update the marks
update_marks(marks, student_id, bonus);

// Print the updated marks of all students
printf("\nUpdated marks of all students:\n");
for (int i = 0; i < n; i++) {
    printf("Student %d marks:\n", i + 1);
    for (int j = 0; j < 5; j++) {
        printf("Subject %d: %d\n", j + 1, marks[i][j]);
    }
    printf("\n");
}

return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

Programs on functions, Array with functions, pointer

1. Sum of Odd Numbers in an Array

Write a program in C to calculate the sum of all Odd numbers in a given array of size ‘n’. The array should be passed as a parameter to a user-defined function.

Solution:

```
#include <stdio.h>

// Function to calculate the sum of odd numbers in the array

int sumOfOddNumbers(int* arr, int size) {

    int sum = 0;

    // Loop through the array and sum up the odd numbers

    for (int i = 0; i < size; i++) {

        if (arr[i] % 2 != 0) { // Check if the number is odd

            sum += arr[i];

        }

    }

    return sum;

}

int main() {

    int n;

    // Input size of the array

    printf("Enter the size of the array: ");

    scanf("%d", &n);

    int arr[n]; // Declare an array of size n
```

```

// Input elements of the array

printf("Enter the elements of the array: \n");

for (int i = 0; i < n; i++) {

    scanf("%d", &arr[i]);

}

// Call the function to get the sum of odd numbers

int result = sumOfOddNumbers(arr, n);

// Output the result

printf("The sum of odd numbers in the array is: %d\n", result);

return 0;

}

```

2. Swapping Elements in an Array Using Pointers:

Write a function in C that accepts an array of integers and swaps two elements at given indices using pointers. The main function should print the modified array.

Solution:

```

#include <stdio.h>

// Function to swap two elements of the array using pointers

void swapElements(int* arr, int index1, int index2) {

    int temp = arr[index1]; // Temporary variable to hold the value at index1

    arr[index1] = arr[index2]; // Assign the value at index2 to index1

    arr[index2] = temp; // Assign the value stored in temp to index2

}

int main() {

    int n;

    // Input the size of the array

    printf("Enter the size of the array: ");


```

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

int arr[n]; // Declare an array of size n

// Input elements of the array

printf("Enter the elements of the array:\n");

for (int i = 0; i < n; i++) {

    scanf("%d", &arr[i]);

}

// Input the indices to swap

int index1, index2;

printf("Enter the indices to swap (0 to %d): ", n - 1);

scanf("%d %d", &index1, &index2);

// Check if indices are valid

if (index1 >= 0 && index1 < n && index2 >= 0 && index2 < n) {

    // Call the function to swap the elements at index1 and index2

    swapElements(arr, index1, index2);

    // Print the modified array

    printf("Modified array: ");

    for (int i = 0; i < n; i++) {

        printf("%d ", arr[i]);

    }

    printf("\n");

} else {

    printf("Invalid indices. Please enter valid indices.\n");

}

return 0;
```

```
}
```

3. Check for Palindrome Number

Write a function that takes an integer as an argument and returns whether the number is a palindrome or not. In the main function, print an appropriate message.

```
#include <stdio.h>

// Function to check if the number is a palindrome

int isPalindrome(int num) {

    int originalNum = num;

    int reversedNum = 0, remainder;

    // Reverse the number

    while (num != 0) {

        remainder = num % 10; // Get the last digit

        reversedNum = reversedNum * 10 + remainder; // Build the reversed number

        num /= 10; // Remove the last digit

    }

    // If the original number and the reversed number are the same, it's a palindrome

    if (originalNum == reversedNum) {

        return 1; // Palindrome

    } else {

        return 0; // Not a palindrome

    }

}

int main() {

    int num;

    // Input the number
```

```

printf("Enter a number: ");

scanf("%d", &num);

// Call the function to check if the number is a palindrome

if (isPalindrome(num)) {

    printf("The number %d is a palindrome.\n", num);

} else {

    printf("The number %d is not a palindrome.\n", num);

}

return 0;
}

```

4. Write a C function that accepts a 2D array of integers where each row represents a student and each column represents the marks in different subjects. The program should allow updating the marks for a specific student by adding bonus marks to all of their subjects. The user will specify the student index and the bonus marks to add.

Original marks:

```

80 90 85
78 88 92
83 89 95
72 75 80
85 95 88

```

Enter the student index (0 to 4) to update: 2

Enter the bonus marks to add: 5

Updated marks:

```

80 90 85
78 88 92
88 94 100
72 75 80
85 95 88

```

Solution:

```

#include <stdio.h>

// Function to update marks for a specific student across all subjects

```

```

void updateStudentMarks(int arr[5][3], int studentIndex, int bonusMarks) {
    // Loop through all subjects (columns) for the specified student
    for (int j = 0; j < 3; j++) {
        arr[studentIndex][j] += bonusMarks; // Add bonus marks to each subject for the student
    }
}

void printMarks(int arr[5][3], int numStudents, int numSubjects) {
    // Print the marks of all students
    for (int i = 0; i < numStudents; i++) {
        for (int j = 0; j < numSubjects; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int arr[5][3] = {
        {80, 90, 85}, // Marks of student 1
        {78, 88, 92}, // Marks of student 2
        {83, 89, 95}, // Marks of student 3
        {72, 75, 80}, // Marks of student 4
        {85, 95, 88} // Marks of student 5
    };
    int studentIndex, bonusMarks;

    // Print the original marks
    printf("Original marks:\n");
    printMarks(arr, 5, 3);
}

```

```
// Input the student index and the bonus marks
printf("\nEnter the student index (0 to 4) to update: ");
scanf("%d", &studentIndex);
printf("Enter the bonus marks to add: ");
scanf("%d", &bonusMarks);

// Check if the student index is valid
if (studentIndex >= 0 && studentIndex < 5) {
    // Call the function to update marks for the specified student
    updateStudentMarks(arr, studentIndex, bonusMarks);

    // Print the updated marks
    printf("\nUpdated marks:\n");
    printMarks(arr, 5, 3);
} else {
    printf("Invalid student index!\n");
}
return 0;
}
```



**Problem Solving with C
LABORATORY MANUAL
Week 6**

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Write a C function to find the sum of elements of an array using a function.

Solution:

```
#include <stdio.h>

int sumArray(int arr[], int size) {
    int sum = 0;
    for (int i = 0; i < size; i++)
        sum += arr[i];
    return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Sum of array elements: %d\n", sumArray(arr, size));
    return 0;
}
```

2. Write a program to remove the duplicate elements from a sorted array.
Implement the remove_duplicates function to solve this problem.

Input :

```
int arr[] = {1, 1, 2, 3, 3, 4, 5, 5, 6, 7, 8, 8, 9};
```

Expected output:

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Solution:

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

```
// Function to remove duplicates from a sorted array and store in another array
void remove_duplicates(int arr[], int n, int result[], int* new_size) {
    // If the array has no elements or just one, return early
    if (n == 0) {
        *new_size = 0;
    }
```

```
return;
}

// Initialize the result array with the first element from the original array
result[0] = arr[0];
*new_size = 1; // The result array initially has one element

// Traverse the original array
for (int i = 1; i < n; i++) {
    // If the current element is different from the previous one
    if (arr[i] != arr[i - 1]) {
        result[*new_size] = arr[i]; // Add unique element to result
        (*new_size)++;
    }
}
}

// Function to print the array
void print_array(int arr[], int size) {
    printf("[");
    for (int i = 0; i < size; i++) {
        printf("%d", arr[i]);
        if (i < size - 1) {
            printf(", ");
        }
    }
    printf("]\n");
}

int main() {
    // Example sorted array with duplicates
    int arr[] = {1, 1, 2, 2, 3, 4, 4, 5, 6, 6};
    int n = sizeof(arr) / sizeof(arr[0]);

    // Create an array to store the unique elements
    int result[n]; // The result array can be at most the size of the input array
    int new_size = 0; // Variable to store the size of the result array

    // Call the function to remove duplicates and store in the result array
    remove_duplicates(arr, n, result, &new_size);
```

- ```
// Print the result array with unique elements
print_array(result, new_size);

return 0;
}

3. Write a C program that reverses an integer array in-place using pointer
arithmetic.
```

**Solution:**

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

```
// Function to reverse an array using pointer arithmetic
void reverseArray(int *arr, int size) {
 int *start = arr;
 int *end = arr + size - 1;
 while (start < end) {
 int temp = *start;
 *start = *end;
 *end = temp;
 start++;
 end--;
 }
}

int main() {
 int arr[] = {10, 20, 30, 40, 50};
 int size = sizeof(arr) / sizeof(arr[0]);
 reverseArray(arr, size);

 printf("Reversed array: ");
 for (int i = 0; i < size; i++) {
 printf("%d ", arr[i]);
 }
 printf("\n");
 return 0;
}
```

- ```
4. Write a C program that calculates the average of an array's elements using a
function that employs pointer arithmetic.
```

Solution:

```
#include <stdio.h>

// Function to compute the average of array elements using pointers
double computeAverage(int *arr, int size) {
    int sum = 0;
    for (int i = 0; i < size; i++) {
        sum += *(arr + i);
    }
    return (double) sum / size;
}

int main() {
    int arr[] = {5, 10, 15, 20, 25};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Average of array: %.2f\n", computeAverage(arr, size));
    return 0;
}
```



**Problem Solving with C
LABORATORY MANUAL
Week 5**

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchor: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Write a function to check if a number is prime.

Solution:

```
#include <stdio.h>

int isPrime(int n) {
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (isPrime(num))
        printf("%d is a prime number\n", num);
    else
        printf("%d is not a prime number\n", num);
    return 0;
}
```

**2. Find the Product of Two Adjacent Elements with the Largest Sum: Write a function that takes an array and its size, finds the largest sum of two adjacent elements, and returns their product. If the array size is less than 2, display an appropriate message. Example: Input: [2, 8, 3, 7, 5]
Output: 56 (since 8 + 7 = 15, and their product is 8 * 7 = 56)**

Solution:

```
#include <stdio.h>

// Function to find the product of two adjacent elements with the largest sum
int productOfLargestSumPair(int arr[], int size) {
    if (size < 2) {
        printf("Array size must be at least 2 to find adjacent pairs.\n");
        return -1; // Return an error code
    }
```

}

```

int maxSum = arr[0] + arr[1]; // Initialize with the sum of the first adjacent
pair
int maxProduct = arr[0] * arr[1]; // Initialize the product of the first adjacent
pair

// Iterate through the array to find the largest sum of adjacent elements
for (int i = 1; i < size - 1; i++) {
    int currentSum = arr[i] + arr[i + 1];
    if (currentSum > maxSum) {
        maxSum = currentSum;
        maxProduct = arr[i] * arr[i + 1]; // Update product if a larger sum is found
    }
}

return maxProduct; // Return the product of the pair with the largest sum
}

int main() {
    int arr[] = {2, 8, 3, 7, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    // Call the function to find the product of the largest sum pair
    int result = productOfLargestSumPair(arr, size);

    // Display the result if the array size is valid
    if (result != -1) {
        printf("The product of the two adjacent elements with the largest sum is:
%d\n", result);
    }

    return 0;
}

```

3. Check if an Array is Sorted Write a function to check if a given array is sorted in non-decreasing order.

Solution:

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

```

// Function to check if the array is sorted in non-decreasing order
int isSorted(int arr[], int size) {
    // If the array has less than 2 elements, it's considered sorted
    if (size < 2) {
        return 1; // Return 1 (true) if sorted
    }

    // Iterate through the array to check adjacent elements
    for (int i = 0; i < size - 1; i++) {
        if (arr[i] > arr[i + 1]) {
            return 0; // Return 0 (false) if an element is greater than the next element
        }
    }
    return 1; // Return 1 (true) if no violations were found
}

int main() {
    int arr[] = {1, 2, 2, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    // Check if the array is sorted
    if (isSorted(arr, size)) {
        printf("The array is sorted in non-decreasing order.\n");
    } else {
        printf("The array is not sorted.\n");
    }

    return 0;
}

```

4. Write a function that calculates the transpose of a given $n \times n$ matrix. The transpose of a matrix is formed by swapping rows and columns.

Solution:

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

```

// Function to calculate the transpose of a given nxn matrix
void transposeMatrix(int matrix[][4], int n, int transpose[][4]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            transpose[j][i] = matrix[i][j]; // Swap rows with columns
        }
    }
}

```

```
        }
    }
}

// Function to print the matrix
void printMatrix(int matrix[][4], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int n = 4;
    int matrix[4][4] = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };

    int transpose[4][4]; // Matrix to store the transpose

    // Call the function to calculate the transpose
    transposeMatrix(matrix, n, transpose);

    // Display the original matrix
    printf("Original Matrix:\n");
    printMatrix(matrix, n);

    // Display the transposed matrix
    printf("\nTransposed Matrix:\n");
    printMatrix(transpose, n);

    return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a function in C that takes a **pointer to an integer** and an **increment value** as arguments. The function should increase the integer's value by the given amount. After calling the function, print the modified value in the **main** function.

Solution:

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

```
// Function to increment the integer by a given amount
```

```
void incrementValue(int *num, int increment) {
```

```
    *num += increment;
```

```
}
```

```
int main() {
```

```
    int value = 10;
```

```
    int incrementAmount = 5;
```

```
    printf("Before increment: %d\n", value);
```

```
// Call the function, passing the address of value
```

```
    incrementValue(&value, incrementAmount);
```

```
    printf("After increment: %d\n", value);
```

```
    return 0;
```

```
}
```

2. Write a function in C that takes an integer as an argument and returns whether the number is even or odd. In the main function, print an appropriate message.

Solution:

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

```

// Function to check if a number is even or odd

int isEven(int num) {
    return (num % 2 == 0); // Returns 1 if even, 0 if odd
}

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);

    // Check the result and print appropriate message
    if (isEven(number)) {
        printf("%d is an even number.\n", number);
    } else {
        printf("%d is an odd number.\n", number);
    }

    return 0;
}

```

3. Write a function that finds the second largest element in a given array and returns that element. It should be printed in the main function.

Solution:

```

#include <stdio.h>

int secondLargest(int arr[], int size) {
    int first = arr[0], second = -1;
    for (int i = 1; i < size; i++) {
        if (arr[i] > first) {

```

```

        second = first;
        first = arr[i];
    } else if (arr[i] > second && arr[i] != first) {
        second = arr[i];
    }
}

return second;
}

int main() {
    int arr[] = {12, 35, 1, 10, 34, 1};
    int size = sizeof(arr) / sizeof(arr[0]);

    int secLargest = secondLargest(arr, size);
    printf("Second largest element: %d\n", secLargest);

    return 0;
}

```

4. You are building a student marks management system for a university. It stores students' marks for a subject in an array.

Write a function that takes an array of student marks and the total number of students and returns the average marks of the class. Use pointers to traverse the array.

Solution:

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

```

float calculateAverage(int *marks, int n) {
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += *(marks + i);
    }
}
```

```
    }

    return (float)sum / n;

}

int main() {

    int marks[] = {85, 90, 78, 92, 88};

    int n = sizeof(marks) / sizeof(marks[0]);

    float avg = calculateAverage(marks, n);

    printf("Average Marks: %.2f\n", avg);

    return 0;

}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a function in C that takes an integer as an argument and returns whether the number is a perfect square or not. In the main function, print an appropriate message.

Solution:

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

// Function to check if a number is a perfect square
int isPerfectSquare(int num) {
    if (num < 0) return 0; // Negative numbers can't be perfect squares
    int sqrtValue = sqrt(num);
    return (sqrtValue * sqrtValue == num);
}

int main() {
    int number;
    printf("Enter a number: ");
    scanf("%d", &number);

    if (isPerfectSquare(number)) {
        printf("%d is a perfect square.\n", number);
    } else {
        printf("%d is not a perfect square.\n", number);
    }

    return 0;
}
```

2. Write a function in C that takes an integer as an argument and returns the number of digits in it. In the `main` function, print the result.

Solution:

```
#include <stdio.h>

// Function to count the number of digits

int countDigits(int num) {

    if (num == 0) return 1; // Special case for 0

    int count = 0;

    while (num != 0) {

        count++;

        num /= 10;

    }

    return count;

}

int main() {

    int number;

    printf("Enter a number: ");

    scanf("%d", &number);

    int digitCount = countDigits(number);

    printf("The number %d has %d digits.\n", number, digitCount);

    return 0;

}
```

3. Write a function in C that takes an array and its size as arguments, and returns the count of even and odd numbers.

Solution:

```
#include <stdio.h>

// Function to count even and odd numbers
void countEvenOdd(int arr[], int size, int *evenCount, int *oddCount) {
    *evenCount = 0;
    *oddCount = 0;

    for (int i = 0; i < size; i++) {
        if (arr[i] % 2 == 0)
            (*evenCount)++;
        else
            (*oddCount)++;
    }
}

int main() {
    int numbers[] = {10, 25, 30, 47, 50, 33};
    int size = sizeof(numbers) / sizeof(numbers[0]);
    int evenCount, oddCount;

    countEvenOdd(numbers, size, &evenCount, &oddCount);

    printf("Even numbers count: %d\n", evenCount);
    printf("Odd numbers count: %d\n", oddCount);

    return 0;
}
```

```
}
```

4. Student Marks Management System Problem: A university stores students' marks for a subject in an array. Use pointers to access and modify the array.

Write a function that takes an array of student marks, a student index, and the new marks. The function should update the marks of the specified student.

Solution:

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

```
void updateMarks(int *marks, int index, int newMarks) {
```

```
    *(marks + index) = newMarks;
```

```
}
```

```
int main() {
```

```
    int marks[] = {85, 90, 78, 92, 88};
```

```
    int n = sizeof(marks) / sizeof(marks[0]);
```

```
    int studentIndex = 2; // Updating marks of student at index 2
```

```
    int newMarks = 95;
```

```
    printf("Before Update: ");
```

```
    for (int i = 0; i < n; i++) {
```

```
        printf("%d ", marks[i]);
```

```
    }
```

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

```
    updateMarks(marks, studentIndex, newMarks);
```

```
    printf("After Update: ");
```

```
for (int i = 0; i < n; i++) {  
    printf("%d ", marks[i]);  
}  
printf("\n");  
  
return 0;  
}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a C program that takes an integer as input, calculates its **square** using a function, and prints the result. The function should accept a **pointer to an integer** and modify the original value.

Solution:

```
#include <stdio.h>

// Function to calculate the square using a pointer
void findSquare(int *num) {
    *num = (*num) * (*num); // Squaring the value
}

int main() {
    int number;

    // User input
    printf("Enter a number: ");
    scanf("%d", &number);

    // Function call (passing address of number)
    findSquare(&number);

    // Printing the modified value
    printf("The square of the number is: %d\n", number);

    return 0;
}
```

2. Reverse an Array

Write a program that reverses the elements of an array in place by passing the array to a function using a pointer.

Solution:

```
#include <stdio.h>

void reverseArray(int *arr, int size) {

    int *start = arr;
    int *end = arr + size - 1;

    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;

        start++;
        end--;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    reverseArray(arr, size);

    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
    return 0;  
}  
}
```

3. Write a function in C that takes a pointer to an array and its size as parameters and returns the maximum element in the array. In the main() function, print the maximum value.

Solution:

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

```
// Function to find the maximum element using pointers  
int findMax(int *arr, int size) {  
    int max = *arr;      // Assume first element is max  
  
    for (int i = 1; i < size; i++) {  
        if (*(arr + i) > max) {          // Access element using pointer arithmetic  
            max = *(arr + i);           // Update max if a larger element is found  
        }  
    }  
  
    return max; // Return the maximum value  
}  
  
int main() {  
    int arr[] = {10, 45, 67, 23, 89, 34, 99, 12};    // Example array  
    int size = sizeof(arr) / sizeof(arr[0]);           // Calculate array size  
  
    // Call function and print the maximum value  
    printf("The maximum element in the array is: %d\n", findMax(arr, size));  
  
    return 0;  
}
```

4. Student Marks Management System Problem: A university stores students' marks for a subject in an array. Use pointers to access and modify the array.

Write a function that takes an array of student marks and counts how many students passed (marks ≥ 40) and how many failed.

Solution:

```
#include <stdio.h>

void countPassFail(int *marks, int n, int *pass, int *fail) {
    *pass = 0;
    *fail = 0;
    for (int i = 0; i < n; i++) {
        if (*(marks + i) >= 40) {
            (*pass)++;
        } else {
            (*fail)++;
        }
    }
}

int main() {
    int marks[] = {85, 90, 30, 92, 38, 50};
    int n = sizeof(marks) / sizeof(marks[0]);
    int pass, fail;

    countPassFail(marks, n, &pass, &fail);
}
```

```
printf("Number of students who passed: %d\n", pass);
printf("Number of students who failed: %d\n", fail);

return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a function in C that takes a pointer to an integer as an argument and doubles the value of the integer. After calling the function, print the modified value in the main function.

```
#include <stdio.h>

// Function to double the value of the integer passed via pointer
void doubleValue(int *num) {
    *num = *num * 2; // Dereference the pointer and double the value
}

int main() {
    int number = 10; // Initialize the integer

    printf("Original value: %d\n", number); // Print the original value

    // Call the function, passing the address of 'number' (pointer)
    doubleValue(&number);

    // Print the modified value
    printf("Modified value after doubling: %d\n", number);

    return 0;
}
```

2. Write a function to find the "missing number" in an array containing numbers from 1 to n, where exactly one number is missing.

```
#include <stdio.h>

int findMissingNumber(int arr[], int size) {
    int totalSum = (size + 1) * (size + 2) / 2; // Sum of numbers from 1 to n
    int arraySum = 0;
    for (int i = 0; i < size; i++) {
        arraySum += arr[i];
    }
    return totalSum - arraySum;
}

int main() {
    int arr[] = {1, 2, 4, 5, 6};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("The missing number is: %d\n", findMissingNumber(arr, size));
    return 0;
}
```

3. Write a function `multiply2DArrays` that takes two 2D arrays (with compatible dimensions) and returns their matrix product.

```
#include <stdio.h>

void multiply2DArrays(int A[2][3], int B[3][2], int C[2][2]) {
    int i, j, k;

    // Multiply A and B and store the result in C
    for (i = 0; i < 2; i++) {
        for (j = 0; j < 2; j++) {
            C[i][j] = 0; // Initialize the result element to 0
            for (k = 0; k < 3; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}

// Function to print the matrix
void printMatrix(int matrix[2][2]) {
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 2; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    // Example matrices A and B
    int A[2][3] = {{1, 2, 3}, {4, 5, 6}};
    int B[3][2] = {{7, 8}, {9, 10}, {11, 12}};
    int C[2][2]; // Resultant matrix

    // Call the multiply function
    multiply2DArrays(A, B, C);

    // Print the result matrix C
    printf("Product of matrices A and B is:\n");
    printMatrix(C);

    return 0;
}
```

4. You are building an inventory management system for a small bookstore. The bookstore tracks its inventory of books using an array where each element represents a book's quantity. The system allows the user to perform various operations such as adding stock, removing stock, and checking the current stock of a particular book.

Question:

Write a C function that accepts an array of integers representing the current stock of books and a specific book's ID. The function should increase the stock of the specified book by a given quantity. Use a pointer to modify the original array.

```
#include <stdio.h>

void reorderStock(int *stock, int size, int threshold) {
    for (int i = 0; i < size; i++) {
        if (stock[i] < threshold) {
            printf("Book ID %d needs to be reordered. Current stock: %d\n", i,
stock[i]);
        }
    }
}

int main() {
    int stock[] = {10, 2, 15, 0, 30}; // Initial stock of books
    int size = sizeof(stock) / sizeof(stock[0]);
    int threshold;

    printf("Enter the reorder threshold: ");
    scanf("%d", &threshold);

    reorderStock(stock, size, threshold);
    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a function that checks if a number is a perfect number. A *perfect number* is a positive integer that is equal to the sum of its proper divisors (excluding the number itself). For example, 6 is a perfect number because its divisors (excluding 6 itself) are 1, 2, and 3, and $1 + 2 + 3 = 6$.

```
#include <stdio.h>

int isPerfectNumber(int n) {
    int sum = 1;
    for (int i = 2; i <= n / 2; i++) {
        if (n % i == 0)
            sum += i;
    }
    return sum == n;
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    if (isPerfectNumber(num))
        printf("%d is a perfect number.\n", num);
    else
        printf("%d is not a perfect number.\n", num);
    return 0;
}
```

2. Write a function to reverse an array.

```
#include <stdio.h>
void reverseArray(int arr[], int size) {
    int temp;
    for (int i = 0; i < size / 2; i++) {
        temp = arr[i];
        arr[i] = arr[size - i - 1];
        arr[size - i - 1] = temp;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    reverseArray(arr, size);
    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
```

```
    return 0;  
}
```

3. Write a function that counts how many times a given element appears in an array. The function should accept a pointer to the array, the size of the array, and the element to search for as parameters.

```
#include <stdio.h>  
  
int countOccurrences(int* arr, int size, int target) {  
    int count = 0;  
  
    for (int i = 0; i < size; i++) {  
        if (*(arr + i) == target) {  
            count++;  
        }  
    }  
  
    return count;  
}  
  
int main() {  
    int arr[] = {1, 2, 3, 4, 5, 2, 2, 6};  
    int size = sizeof(arr) / sizeof(arr[0]);  
    int target = 2;  
  
    printf("Element %d occurs %d times in the array.\n", target,  
countOccurrences(arr, size, target));  
  
    return 0;  
}
```

4. Write a program to add two matrices using pointers.

```
#include <stdio.h>  
  
void addMatrices(int *a, int *b, int *result, int rows, int cols) {  
    for (int i = 0; i < rows; i++) {  
        for (int j = 0; j < cols; j++) {  
            *(result + i * cols + j) = *(a + i * cols + j) + *(b + i * cols +  
j);  
        }  
    }  
}  
  
int main() {  
    int rows = 2, cols = 2;  
    int matrix1[2][2] = {{1, 2}, {3, 4}};  
    int matrix2[2][2] = {{5, 6}, {7, 8}};  
    int result[2][2];
```

```
addMatrices((int *)matrix1, (int *)matrix2, (int *)result, rows, cols);

printf("Resultant Matrix:\n");
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        printf("%d ", result[i][j]);
    }
    printf("\n");
}
return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Write a function that counts the number of set bits (1's) in the binary representation of an integer.

```
#include <stdio.h>

int countSetBits(int n) {
    int count = 0;
    while (n) {
        count += n & 1; // Add 1 if the last bit is set
        n >>= 1;           // Right shift n by 1
    }
    return count;
}

int main() {
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Number of set bits in %d: %d\n", num, countSetBits(num));
    return 0;
}
```

2. Write a function to find the second largest element in an array.

```
#include <stdio.h>

int secondLargest(int arr[], int size) {
    int largest = arr[0];
    int secondLargest = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] > secondLargest && arr[i] != largest) {
            secondLargest = arr[i];
        }
    }
    return secondLargest;
}

int main() {
    int arr[] = {12, 35, 1, 10, 34, 1};
    int size = sizeof(arr) / sizeof(arr[0]);
    printf("Second largest element is: %d\n", secondLargest(arr, size));
    return 0;
}
```

- 3. Write a C function that searches for an element in an array. The function will return the index of the element if it's found, or -1 if the element is not present in the array.**

Input 1:

```
int a[5]={10,20,30,40,50}
```

```
Enter the element to search: 30
```

Output 1:

```
Element 30 found at index 2.
```

```
#include <stdio.h>

// Function to search for an element in the array
int searchElement(int* arr, int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i; // Element found, return its index
        }
    }
    return -1; // Element not found
}

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target;

    // Ask the user for an element to search
    printf("Enter the element to search: ");
    scanf("%d", &target);

    // Call the search function and store the result
    int result = searchElement(arr, size, target);

    // Print the result
    if (result != -1) {
        printf("Element %d found at index %d.\n", target, result);
    } else {
        printf("Element %d not found in the array.\n", target);
    }

    return 0;
}
```

4. You need to analyse the inventory by calculating the average stock of all books in the bookstore.

Question: Write a C function that accepts an array of integers representing the current stock of books. The function should calculate and return the average stock of all books. Use a pointer to traverse through the array.

```
#include <stdio.h>

float calculateAverageStock(int *stock, int size) {
    int total = 0;
    for (int i = 0; i < size; i++) {
        total += *(stock + i); // Sum the stock of all books
    }
    return (float)total / size; // Return average stock
}

int main() {
    int stock[] = {10, 20, 30, 40, 50}; // Initial stock of books
    int size = sizeof(stock) / sizeof(stock[0]);

    printf("Average stock of all books: %.2f\n", calculateAverageStock(stock,
size));
    return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 6

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

- 1. Write a C function to reverse an array using pointers. The function should accept the array and its size as arguments and print the reversed array.**

Solution:

```
#include <stdio.h>
void reverseArray(int* arr, int size) {
    int* start = arr;
    int* end = arr + size - 1;

    while (start < end) {
        int temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    printf("Original array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    reverseArray(arr, size);

    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

- 2. Write a function that finds the maximum element in an array using pointers.
The function should return the maximum value.**

Solution:

```
#include <stdio.h>

int findMax(int* arr, int size) {

    int max = *arr;

    for (int i = 1; i < size; i++) {

        if (*(arr + i) > max) {

            max = *(arr + i);

        }

    }

    return max;

}

int main() {

    int arr[] = {10, 20, 30, 50, 40};

    int size = sizeof(arr) / sizeof(arr[0]);

    int max = findMax(arr, size);

    printf("Maximum element is: %d\n", max);

    return 0;

}
```

- 3. Write a function that calculates the sum of all elements in an array using pointer arithmetic.**

Solution:

```
#include <stdio.h>

int sumArray(int* arr, int size) {
```

```

int sum = 0;
for (int i = 0; i < size; i++) {
    sum += *(arr + i);
}
return sum;
}

int main() {
    int arr[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr) / sizeof(arr[0]);

    int total = sumArray(arr, size);
    printf("Sum of elements: %d\n", total);

    return 0;
}

```

4. Write a function that counts the occurrences of a given element in an array using pointers.

Solution:

```

#include <stdio.h>

int countOccurrences(int* arr, int size, int element) {
    int count = 0;
    for (int i = 0; i < size; i++) {
        if (*(arr + i) == element) {
            count++;
        }
    }
    return count;
}

```

```

}

int main() {
    int arr[] = {1, 2, 3, 2, 4, 2, 5};
    int size = sizeof(arr) / sizeof(arr[0]);
    int element = 2;
    int count = countOccurrences(arr, size, element);
    printf("Element %d occurs %d times\n", element, count);
    return 0;
}

```

5. Write a function to copy one array into another using pointers.

Solution:

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

```

void copyArray(int* source, int* destination, int size) {
    for (int i = 0; i < size; i++) {
        *(destination + i) = *(source + i);
    }
}

```

```

int main() {
    int source[] = {1, 2, 3, 4, 5};
    int size = sizeof(source) / sizeof(source[0]);
    int destination[5];
}

```

```
copyArray(source, destination, size);
```

```
printf("Destination array: ");
```

```
for (int i = 0; i < size; i++) {
```

```

    printf("%d ", destination[i]);
}

printf("\n");

return 0;
}

```

6. Write a function that compares two arrays and returns 1 if they are equal and 0 otherwise.

Solution:

```

#include <stdio.h>

int compareArrays(int* arr1, int* arr2, int size) {
    for (int i = 0; i < size; i++) {
        if (*(arr1 + i) != *(arr2 + i)) {
            return 0;
        }
    }
    return 1;
}

int main() {
    int arr1[] = {1, 2, 3, 4, 5};
    int arr2[] = {1, 2, 3, 4, 5};
    int size = sizeof(arr1) / sizeof(arr1[0]);

    int result = compareArrays(arr1, arr2, size);
    if (result == 1) {
        printf("Arrays are equal\n");
    } else {
        printf("Arrays are not equal\n");
    }
}

```

```
}
```

```
return 0;
```

```
}
```

7. Write a function that calculates the sum of the diagonal elements of a square matrix using pointers.

Solution:

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

```
int sumDiagonal(int* matrix, int n) {
```

```
    int sum = 0;
```

```
    for (int i = 0; i < n; i++) {
```

```
        sum += *(matrix + i * n + i); // Accessing the diagonal element
```

```
}
```

```
    return sum;
```

```
}
```

```
int main() {
```

```
    int n = 3;
```

```
    int matrix[3][3] = {
```

```
        {1, 2, 3},
```

```
        {4, 5, 6},
```

```
        {7, 8, 9}
```

```
    };
```

```
    int sum = sumDiagonal((int*)matrix, n);
```

```
    printf("Sum of diagonal elements: %d\n", sum);
```

```
    return 0;
```

```
}
```

8. Write a function to compute the transpose of a matrix using pointers. The function should modify the original matrix.

Solution:

```
#include <stdio.h>

void transposeMatrix(int* matrix, int rows, int cols) {

    for (int i = 0; i < rows; i++) {
        for (int j = i + 1; j < cols; j++) {
            int temp = *(matrix + i * cols + j);
            *(matrix + i * cols + j) = *(matrix + j * cols + i);
            *(matrix + j * cols + i) = temp;
        }
    }
}

int main() {
    int rows = 2, cols = 3;
    int matrix[2][3] = {
        {1, 2, 3},
        {4, 5, 6}
    };

    printf("Original matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}
```

```

transposeMatrix((int*)matrix, rows, cols);

printf("Transposed matrix:\n");

for (int i = 0; i < cols; i++) {

    for (int j = 0; j < rows; j++) {

        printf("%d ", matrix[j][i]);

    }

    printf("\n");

}

return 0;

}

```

- 9. Given two matrices, A and B, you are required to calculate their product C. The number of columns in matrix A must equal the number of rows in matrix B for the multiplication to be valid. The result will be a matrix C whose dimensions are the number of rows of A and the number of columns of B.**

Solution:

```

#include <stdio.h>

#define MAX 10 // Maximum size of matrix

// Function to multiply two matrices

void multiplyMatrices(int A[MAX][MAX], int B[MAX][MAX], int C[MAX][MAX], int rowA,
int colA, int rowB, int colB) {

    // Initialize the result matrix C with zeros

    for (int i = 0; i < rowA; i++) {

        for (int j = 0; j < colB; j++) {

            C[i][j] = 0;

        }

    }

    // Perform matrix multiplication

    for (int i = 0; i < rowA; i++) {

```

```

        for (int j = 0; j < colB; j++) {
            for (int k = 0; k < colA; k++) {
                C[i][j] += A[i][k] * B[k][j];
            }
        }
    }

// Function to print a matrix

void printMatrix(int matrix[MAX][MAX], int row, int col) {
    for (int i = 0; i < row; i++) {
        for (int j = 0; j < col; j++) {
            printf("%d ", matrix[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int rowA, colA, rowB, colB;
    // Input dimensions for matrix A
    printf("Enter the number of rows and columns for matrix A: ");
    scanf("%d %d", &rowA, &colA);
    // Input dimensions for matrix B
    printf("Enter the number of rows and columns for matrix B: ");
    scanf("%d %d", &rowB, &colB);
    // Check if multiplication is possible (cols of A must be equal to rows of B)
    if (colA != rowB) {
        printf("Matrix multiplication is not possible. The number of columns of A must be
equal to the number of rows of B.\n");
    }
}

```

```
    return 1;
}

int A[MAX][MAX], B[MAX][MAX], C[MAX][MAX];

// Input elements of matrix A

printf("Enter elements of matrix A:\n");

for (int i = 0; i < rowA; i++) {

    for (int j = 0; j < colA; j++) {

        scanf("%d", &A[i][j]);
    }
}

// Input elements of matrix B

printf("Enter elements of matrix B:\n");

for (int i = 0; i < rowB; i++) {

    for (int j = 0; j < colB; j++) {

        scanf("%d", &B[i][j]);
    }
}

// Multiply matrices A and B, and store the result in matrix C

multiplyMatrices(A, B, C, rowA, colA, rowB, colB);

// Output the result

printf("Resultant matrix C (A x B):\n");

printMatrix(C, rowA, colB);

return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhuram Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

To Learn and Solve	Programs on Nested Control structures
---------------------------	---------------------------------------

1) Create a calculator program that:

- Uses a switch statement for operation selection.
- Uses nested loops for multiple calculations.
- Supports:
 1. Basic Arithmetic (+, -, *, /)
 2. Power and Square Root
 3. Factorial Calculation
- Allows the user to continue operations without restarting the program (do while).

Solution:

```

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

int main() {
    int choice, num1, num2, op;
    double result;
    char cont;

    do {
        printf("\nCalculator Menu:\n1. Basic Arithmetic\n2. Power & Square Root\n3.
Factorial\n4. Exit\nEnter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: // Basic Arithmetic
                do {
                    printf("\nSelect operation:\n1. Add\n2. Subtract\n3. Multiply\n4.
Divide\nEnter choice: ");
                    scanf("%d", &op);
                    printf("Enter two numbers: ");
                    if (scanf("%d %d", &num1, &num2) != 2) {
                        printf("Invalid input! Please enter two numbers.\n");
                        while (getchar() != '\n'); // Clear input buffer
                        continue;
                    }
                    switch (op) {
                        case 1: result = num1 + num2; break;
                        case 2: result = num1 - num2; break;
                        case 3: result = num1 * num2; break;
                        case 4:
                            if (num2 != 0)
                                result = (double)num1 / num2;
                            else {
                                printf("Error! Division by zero is not allowed.\n");
                                continue;
                            }
                            break;
                        default: printf("Invalid operation!\n");
                    }
                    printf("Result: %.2lf\n", result);
                    printf("Do you want another calculation? (y/n): ");
                    scanf(" %c", &cont);
                } while (cont == 'y' || cont == 'Y');
                break;

            case 2: // Power & Square Root
                do {
                    printf("Enter a number: ");
                    if (scanf("%d", &num1) != 1) {
                        printf("Invalid input! Please enter a number.\n");
                        while (getchar() != '\n'); // Clear input buffer
                        continue;
                    }
                }
        }
    }
}

```

```

printf("Choose:\n1. Power (square)\n2. Square root\nEnter choice: ");
int subChoice;
scanf("%d", &subChoice);
if (subChoice == 1)
    result = pow(num1, 2);
else if (subChoice == 2)
    result = sqrt(num1);
else {
    printf("Invalid choice!\n");
    continue;
}
printf("Result: %.2lf\n", result);
printf("Do you want to return to this menu? (y/n): ");
scanf(" %c", &cont);
} while (cont == 'y' || cont == 'Y');
break;

case 3: // Factorial
do {
    printf("Enter a number: ");
    if (scanf("%d", &num1) != 1 || num1 < 0) {
        printf("Invalid input! Please enter a non-negative number.\n");
        while (getchar() != '\n'); // Clear input buffer
        continue;
    }
    int fact = 1;
    for (int i = 1; i <= num1; i++)
        fact *= i;
    printf("Factorial: %d\n", fact);
    printf("Do you want to return to this menu? (y/n): ");
    scanf(" %c", &cont);
} while (cont == 'y' || cont == 'Y');
break;

return 0;

default:
    printf("Invalid choice!\n");
}

// Ensure correct handling of returning to the main menu
do {
    printf("Do you want to return to the main menu? (y/n): ");
    scanf(" %c", &cont);
} while (cont != 'y' && cont != 'Y' && cont != 'n' && cont != 'N');

} while (cont == 'y' || cont == 'Y');

printf("Exiting...\n");
return 0;
}

```

2) Write a C program that implements a **Guess the Number Game** where the user has to guess a randomly generated number between 1 and 100. The program should:

1. Use a while loop to keep prompting the user to guess the number until they guess it correctly.
2. Provide hints after each guess:
 - o "Too High" if the guess is greater than the target number.
 - o "Too Low" if the guess is less than the target number.
3. Track the number of attempts made by the user to guess the number.
4. After a correct guess, prompt the user whether they want to play again. The game should allow multiple rounds of guessing.
5. Use a **nested loop** to allow multiple rounds, where the outer loop manages the rounds and the inner loop handles the guessing within each round.

Input:

- The program should generate a random number between 1 and 100.
- The user will input guesses in each round until they correctly guess the number.

Output:

- After each guess, the program should output whether the guess is "Too High" or "Too Low."
- When the correct number is guessed, the program should congratulate the user and show the number of attempts.
- After each round, the program should ask if the user wants to play again. If they input 1, the game continues; if they input 0, the game ends.

HINT:

```
1 #include <time.h>
2
3 srand(time(NULL)); // Seed the random number generator
        with the current time
4 int number = rand() % 100 + 1; // Generates a random
        number between 1 and 100
5 |
```

Solution:

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

int main() {
    int playAgain = 1;

    // Seed the random number generator
    srand(time(NULL));

    while (playAgain) {
        int number = rand() % 100 + 1;
        int guess;
        int attempts = 0;

        printf("Welcome to Guess the Number Game!\n");
        printf("I have selected a number between 1 and 100. Try to guess it!\n");

        while (1) {
            printf("Enter your guess: ");
            scanf("%d", &guess);
            attempts++;

            if (guess > number) {
                printf("Too High! Try again.\n");
            } else if (guess < number) {
                printf("Too Low! Try again.\n");
            } else {
                printf("Congratulations! You've guessed the correct number %d in %d
attempts!\n", number, attempts);
                break;
            }
        }

        printf("\nDo you want to play again? (1 = Yes, 0 = No): ");
        scanf("%d", &playAgain);
    }

    printf("Thanks for playing! Goodbye!\n");

    return 0;
}
```

3) Write a program to print a **hollow pyramid** pattern using nested loops. The program should print a pyramid shape where only the first and last row are fully filled with stars (*), and the interior of the pyramid has stars only at the boundaries.

```

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

Solution:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the height of the pyramid (5-20): ");
    scanf("%d", &n);

    // Loop for each row
    for (int i = 1; i <= n; i++) {
        // Loop to print spaces
        for (int j = 1; j <= n - i; j++) {
            printf(" ");
        }

        // Loop to print stars
        for (int j = 1; j <= (2 * i - 1); j++) {
            // Print star on the boundaries or for the first and last row
            if (i == n || j == 1 || j == (2 * i - 1)) {
                printf("*");
            } else {
                printf(" ");
            }
        }

        printf("\n");
    }

    return 0;
}
```

4) Write a program that prints a cross-number pattern for a given odd integer n. The program should generate a pattern of numbers where numbers decrease inward, with each line forming a "cross" as shown below for n = 5.

1	2	3	4	5
2	3	4		
3				
2	3	4		
1	2	3	4	5

Solution:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the pattern (odd number): ");
    scanf("%d", &n);

    if (n % 2 == 0) {
        printf("Please enter an odd number.\n");
        return 1;
    }

    // Upper half
    for (int i = 0; i < n / 2 + 1; i++) {
        for (int j = 0; j < n; j++) {
            if (j >= i && j < n - i)
                printf("%d ", j + 1);
            else
                printf(" ");
        }
        printf("\n");
    }

    // Lower half (mirror image of upper half)
    for (int i = n / 2 - 1; i >= 0; i--) {
        for (int j = 0; j < n; j++) {
            if (j >= i && j < n - i)
                printf("%d ", j + 1);
            else
                printf(" ");
        }
        printf("\n");
    }

    return 0;
}
```



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



**Problem Solving with C
LABORATORY MANUAL**

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhу R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

To Learn and Solve	Programs on Nested Control structures
---------------------------	---------------------------------------

- 1) Write a C program that accepts three side lengths of a triangle and determines whether the triangle is Equilateral, Isosceles, Scalene, or if the given sides do not form a valid triangle.

Solution:

```
#include <stdio.h>

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

    printf("Enter three sides of the triangle: ");
    if (scanf("%d %d %d", &a, &b, &c) != 3 || a <= 0 || b <= 0 || c <= 0) {
        printf("Invalid input! Please enter positive integers only.\n");
        return 1;
    }
    if (a + b <= c || a + c <= b || b + c <= a) {
        printf("Invalid triangle! The given sides do not satisfy the triangle inequality theorem.\n");
        return 1;
    }
    int type;
    if (a == b && b == c)
        type = 1; // Equilateral
    else if (a == b || b == c || a == c)
        type = 2; // Isosceles
    else
        type = 3; // Scalene
    switch (type) {
        case 1:
            printf("Equilateral Triangle\n");
            break;
        case 2:
            printf("Isosceles Triangle\n");
            break;
        case 3:
            printf("Scalene Triangle\n");
            break;
        default:
            printf("Error in determining triangle type!\n");
    }
    return 0;
}
```

2) Write a program in C to print all the prime numbers between 1 and a given number n using nested loops.

- A prime number is a number greater than 1 that is divisible only by 1 and itself.
- The program should take an upper limit n as input and print all the prime numbers in the range from 1 to n.

Solution:

```
#include <stdio.h>

int main() {
    int n, i, j, isPrime;

    printf("Enter the upper limit: ");
    scanf("%d", &n);

    printf("Prime numbers between 1 and %d are:\n", n);

    for(i = 2; i <= n; i++) {
        isPrime = 1;
        for(j = 2; j * j <= i; j++) {
            if(i % j == 0) {
                isPrime = 0;
                break;
            }
        }

        if(isPrime) {
            printf("%d ", i);
        }
    }

    return 0;
}
```

Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4

3) Write a C program that prints Floyd's Triangle. The program should take an integer input from the user, representing the number of rows in the triangle. The program should then print the triangle in the following format:

#Rows=4

```
1
2 3
4 5 6
7 8 9 10
```

Solution:

```
#include <stdio.h>

int main() {
    int rows, num = 1;

    printf("Enter number of rows: ");
    scanf("%d", &rows);

    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", num);
            num++;
        }
        printf("\n");
    }

    return 0;
}
```

4) Write a program in C to print the multiplication table for a given number using a do-while loop.

- The program should take an integer n as input.
- The program should print the multiplication table for n from 1 to 10 in the format: $n \times i = \text{result}$ where i ranges from 1 to 10.

Solution:

```
#include <stdio.h>

int main() {
    int n, i = 1;

    printf("Enter a number: ");
    scanf("%d", &n);

    do {
        printf("%d x %d = %d\n", n, i, n * i);
        i++;
    } while(i <= 10);

    return 0;
}
```



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



**Problem Solving with C
LABORATORY MANUAL**

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhу R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

To Learn and Solve	Programs on Nested Control structures
---------------------------	---------------------------------------

- 1) Write a C program to find and print all Armstrong numbers between 1 and a given number n. An Armstrong number (or narcissistic number) of d digits is a number where the sum of each digit raised to the power of d equals the number itself.

Example:

For 153:

$1^3 + 5^3 + 3^3 = 153 \rightarrow \text{Armstrong number}$

Solution:

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

int main() {
    int n;

    printf("Enter the upper limit: ");
    scanf("%d", &n);

    printf("Armstrong numbers between 1 and %d:\n", n);

    for (int i = 1; i <= n; i++) {
        int originalNum = i, sum = 0, digits = 0, temp = i, remainder;

        while (temp != 0) {
            temp /= 10;
            digits++;
        }

        temp = i;
        while (temp != 0) {
            remainder = temp % 10;
            sum += pow(remainder, digits);
            temp /= 10;
        }

        if (sum == originalNum) {
            printf("%d ", i);
        }
    }

    printf("\n");
    return 0;
}
```

2) Write a C program to print all palindromic numbers between 1 and a given number n.

A palindromic number is a number that remains the same when its digits are reversed.

For example:

- 121 → Reverse is 121 (Palindromic)
- 131 → Reverse is 131 (Palindromic)
- 152 → Reverse is 251 (Not Palindromic)

```
#include <stdio.h>

int main() {
    int n;

    printf("Enter the upper limit: ");
    scanf("%d", &n);

    printf("Palindromic numbers between 1 and %d:\n", n);

    for (int i = 1; i <= n; i++) {
        int original = i, reversed = 0, remainder, temp = i;

        while (temp != 0) {
            remainder = temp % 10;
            reversed = reversed * 10 + remainder;
            temp /= 10;
        }

        if (original == reversed) {
            printf("%d ", original);
        }
    }

    printf("\n");
    return 0;
}
```

3) Write a C program to find and print all twin prime numbers between 1 and a given number n.
Twin primes are pairs of prime numbers that differ by exactly 2.

For example:

- (3, 5) → Both are prime and $5 - 3 = 2$
- (5, 7) → Both are prime and $7 - 5 = 2$
- (8, 10) → Not prime
- (11, 13) → Both are prime and $13 - 11 = 2$

Solution:

```
#include <stdio.h>

int main() {
    int n;

    printf("Enter the upper limit: ");
    scanf("%d", &n);
    printf("Twin prime numbers between 1 and %d:\n", n);

    // Loop through numbers to find twin primes
    for (int i = 2; i <= n - 2; i++) {
        int isPrime1 = 1, isPrime2 = 1;

        // Check if 'i' is prime
        if (i < 2) isPrime1 = 0;
        for (int j = 2; j * j <= i; j++) {
            if (i % j == 0) {
                isPrime1 = 0;
                break;
            }
        }
        // Check if 'i+2' is prime
        int next = i + 2;
        if (next < 2) isPrime2 = 0;
        for (int j = 2; j * j <= next; j++) {
            if (next % j == 0) {
                isPrime2 = 0;
                break;
            }
        }
        if (isPrime1 && isPrime2) {
            printf("%d, %d)\n", i, next);
        }
    }
    return 0;
}
```

4) Write a C program that repeatedly asks the user for a number and checks if it is a perfect number. A perfect number is one where the sum of its proper divisors equals the number itself. The program should stop when the user enters 0.

Solution:

```
#include <stdio.h>

int main() {
    int num, sum, i;

    do {
        printf("Enter a number: ");
        scanf("%d", &num);

        if (num == 0) {
            printf("Exiting the program.\n");
            break;
        }

        sum = 0;
        for (i = 1; i <= num / 2; i++) {
            if (num % i == 0) {
                sum += i;
            }
        }

        if (sum == num) {
            printf("%d is a perfect number.\n", num);
        } else {
            printf("%d is not a perfect number.\n", num);
        }
    } while (num != 0);

    return 0;
}
```



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



Department of CSE, PES University
UE24CS151B - Problem Solving with C
Laboratory-Week 4



Problem Solving with C

LABORATORY MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

- 1) Write a C program to perform simple tasks based on operators. Use switch case statements.
- If \$ - Take next 3 floating point numbers input as P, R and T respectively and calculate simple interest
 - If ! - Take next 2 integers and perform arithmetic operations, that is, +, -, *, /. Take the operator also as input.
 - If ? - Take next 2 integers and perform bitwise operations, that is, &, |, Left shift (<) and Right shift(>). Take operator also as input.

Solution:

```
#include <stdio.h>
int
main()
{
    char choice;
    scanf(" %c", &choice);

    switch(choice)
    {
        case '$':
        {
            float P, R, T;
            scanf("%f %f %f", &P, &R, &T);

            float simple_interest = (P * R * T) / 100.0; printf("%.2f\n",
simple_interest);
        }
        break;

        case '!':
        {
            int num1, num2; char
            operator;
            scanf("%d %d %c", &num1, &num2, &operator);
            int result;

            switch(operator)
            {
                case '+':
                    result = num1 + num2;
                    printf("%d\n", result);
                    break;
                case '-':

```

```

        result = num1 - num2;
        printf("%d\n", result); break;
    case '*':
        result = num1 * num2;
        printf("%d\n", result); break;
    case '/':
        if (num2 == 0)
        {
            printf("Error\n"); return 0;
        }
        float res;
        res = (float)num1 / (float)num2;
        printf("%.2f\n", res);
        break;
    default:
        printf("Invalid operator\n");
    }

}

break;

case '?':
{
    int num1, num2;
    char operator;
    scanf("%d %d %c", &num1, &num2, &operator);

    int result;
    switch(operator)
    {
        case '&':
            result = num1 & num2; break;
        case '|':
            result = num1 | num2; break;
        case '<':
            result = num1 << num2; break;
        case '>':
            result = num1 >> num2; break;
        default:
            printf("Invalid operator\n"); return 0;
    }

    printf("%d\n", result);
}
break;

default:
    printf("Invalid choice\n");
}

return 0;
}

```

- 2) Find the smallest and second largest number from the given set of 5 integers using control structures.

Solution:

```
#include <stdio.h>

int main()
{
    int num, largest, secondLargest, smallest;
    scanf("%d", &num);

    largest = secondLargest = num;
    smallest = num;

    for (int i = 1; i <= 4; ++i)
    {
        scanf("%d", &num);

        if (num > largest)
        {
            secondLargest = largest; largest =
            num;
        }
        else if (num > secondLargest && num != largest)
            secondLargest = num;

        if (num < smallest)
            smallest = num;
    }

    printf("%d %d\n", smallest, secondLargest);

    return 0;
}
```

3. Write a C program to print Pascal's Triangle up to a given number of rows.

```

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
```

Solution:

```

#include <stdio.h>
int main() {
    int rows;
    scanf("%d", &rows);
    for (int i = 0; i < rows; i++) {
        int coef = 1;
        for (int j = 0; j <= i; j++) {
            printf("%d ", coef);
            coef = coef * (i - j) / (j + 1);
        }
        printf("\n");
    }
    return 0;
}

```

4. Write a C program using loops to print the sum of Fibonacci prime number series up to a given number.

Solution:

```

#include <stdio.h>

int main() {

    int limit;

    if (scanf("%d", &limit) != 1 || limit <= 0)

    { printf("Wrong input\n");

```

```
return 0;

}

int prev = 0, curr = 1, next, sum = 0;

while (curr <= limit) {

    int is_prime = 1;

    if (curr == 1) {

        is_prime = 0;

    } else {

        for (int i = 2; i * i <= curr; i++) {

            if (curr % i == 0) {

                is_prime = 0; break;

            }

        }

    }

    if (is_prime)

        sum += curr;

    next = prev + curr;

    prev = curr;

    curr = next;

}

printf("%d\n", sum);

return 0;

}
```



Department of CSE, PES University
UE24CS151B – Problem Solving
with C Laboratory-Week 4



Problem Solving with C

LABORATORY MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

1. Write a C program that takes an integer as input, counts the number of digits in it and displays the same, and then checks if the (sum of its digits + the product of its digits) is equal to the original number. If the conditions are met, print the number; otherwise, display zero.

Solution:

```
#include<stdio.h>

int main()
{
    int num, originalNum, digit, sum = 0, product = 1, count = 0;
    scanf("%d", &num);
    originalNum = num;

    while(num != 0)
    {
        digit = num % 10;
        sum += digit;
        product *= digit;
        num /= 10;
        count++;
    }

    printf("%d\n", count);
    if ((sum + product) == originalNum)

    {
        printf("%d\n", originalNum);
    }
    else
        printf("0\n");
    return 0;
}
```

2. Develop a C program to determine whether a given positive integer is a Dudeney number. A Dudeney number is defined as a number that is equal to the cube of the sum of its digits.

e.g.

512:

$$5+1+2=8$$

$$8^3 = 512$$

Solution:

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

int main() {
    int number,check, sum = 0, temp, digitCount = 0;
    // Prompt the user to enter a number
    scanf("%d", &number);
    temp = number;
    // Calculate the sum of the cubes of the digits
    while (temp > 0) {
        int digit = temp % 10;
        digitCount++;
        sum += digit;
        temp /= 10;
    }

    check = sum * sum * sum;
    // Check if the entered number is a Dudeney number
    if (check == number) {
        printf("Dudeney number\n");
    } else {
        printf("Not a Dudeney number\n");
    }
}

return 0;
}
```

3. Write a C program to find the sum of all prime numbers in a given range.

Explanation

- Prime numbers between **10 and 30**: **11, 13, 17, 19, 23, 29**
- Their sum: **11 + 13 + 17 + 19 + 23 + 29 = 129**

Solution:

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

```
#include <stdio.h>

int main() {

    int lower, upper, i, j, sum = 0, isPrime;

    // Input range from user

    printf("Enter the lower limit: ");

    scanf("%d", &lower);

    printf("Enter the upper limit: ");

    scanf("%d", &upper);

    // Loop through each number in the range

    for (i = lower; i <= upper; i++) {

        if (i < 2) continue; // Ignore numbers less than 2

        isPrime = 1; // Assume number is prime

        // Check if 'i' is prime

        for (j = 2; j * j <= i; j++) {

            if (i % j == 0) {

                isPrime = 0; // Not a prime number

                break;
            }
        }

        // If prime, add to sum

        if (isPrime) {
```

```
sum += i;  
}  
}  
  
// Display the result  
  
printf("Sum of prime numbers between %d and %d is: %d\n", lower, upper, sum);  
  
return 0;  
}
```

4. Write a C program to find the closest Fibonacci number to a given number.

Explanation

- Fibonacci sequence: **0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...**
- **20** is between **13 and 21**.
- **21** is closer to **20** than 13, so the program returns **21**.

Solution:

```
#include <stdio.h>  
  
int main() {  
    int n, a = 0, b = 1, next, closest;  
  
    // Input from user  
    printf("Enter a number: ");  
    scanf("%d", &n);  
  
    // Handle small cases  
    if (n == 0) {
```

```
printf("Closest Fibonacci number is: 0\n");

return 0;

}

// Generate Fibonacci numbers until we pass 'n'

while (b < n) {

    next = a + b;
    a = b;
    b = next;
}

// Find the closest Fibonacci number

if (n - a <= b - n)

    closest = a;
else

    closest = b;

// Print the result

printf("Closest Fibonacci number is: %d\n", closest);

return 0;

}
```



Problem Solving with C

LABORATORY MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Prof. Sindhu R Pai

Lab Anchors: Prof. Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Write a C program to print a pyramid of numbers based on user input.

```
1  
1 2 3  
1 2 3 4 5  
1 2 3 4 5 6 7  
1 2 3 4 5 6 7 8 9
```

Solution:

```
#include <stdio.h>  
  
int main() {  
    int rows;  
  
    printf("Enter the number of rows: ");  
  
    scanf("%d", &rows);  
  
    for (int i = 1; i <= rows; i++) {  
  
        for (int j = 1; j <= rows - i; j++)  
  
            printf(" "); // Print spaces for alignment  
  
        for (int j = 1; j <= (2 * i - 1); j++)  
  
            printf("%d", j); // Print numbers  
  
        printf("\n");  
    }  
  
    return 0;  
}
```

2. Write a C program to find the sum of all even numbers and the product of all odd numbers in a given group of n numbers.

Solution:

```
#include <stdio.h>

int main() {
    int N, sumEven = 0, productOdd = 1;
    scanf("%d", &N);

    int num;
    for (int i = 0; i < N; i++) {
        scanf("%d", &num);

        if (num % 2 == 0) {
            sumEven += num;
        } else {
            productOdd *= num;
        }
    }

    printf("%d %d\n", sumEven, productOdd);
    return 0;
}
```

3. Given all three sides and angles of a triangle print whether the triangle is Equilateral, Isosceles, Scalene by checking the sides and Acute, Right, Obtuse Triangle by checking the angles.

Solution:

```
#include<stdio.h>

int main()
{
    float s1, s2, s3;
    int a1, a2, a3;

    scanf("%f%f%f", &s1, &s2, &s3);
    scanf("%d%d%d", &a1, &a2, &a3);

    if (s1 == s2 && s2 == s3){
        printf("Equilateral");
    }
    else if (s1 == s2 || s2 == s3 || s3 == s1){
        printf("Isoceles");
    }
    else if (s1 != s2 && s2 != s3){
        printf("Scalene");
    }

    if (a1>90 || a2>90 || a3>90){
        printf(" Obtuse");
    }
    else if (a1 == 90 || a2 == 90 || a3 == 90){
        printf(" Right");
    }
    else if (a1<90 || a2<90 ||
             a3<90){ printf(" Acute");
    }
    return 0;
}
```

-
4. You are given an integer n, calculate the sum of prime digits present in the given integer n.

Solution:

```
#include <stdio.h>
int main() {
    int num;
    // Taking input for num
    scanf("%d", &num);

    // Calculate the sum of prime digits for num
    int sum = 0;
    int temp = num;
    while (temp > 0) {
        int digit = temp % 10;
        if (digit == 2 || digit == 3 || digit == 5 || digit == 7) {
            sum += digit;
        }
        temp /= 10;
    }

    // Output the results
    printf("%d", sum);
    return 0;
}
```



Problem Solving with C LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Count Even and Odd Digits

Question: Write a C program that takes an integer as input and counts the number of even and odd digits in it. Print the counts separately.

Answer:

```
#include <stdio.h>

int main() {
    int num, digit, even = 0, odd = 0;
    scanf("%d", &num);
    while (num != 0) {
        digit = num % 10;
        if (digit % 2 == 0)
            even++;
        else
            odd++;
        num /= 10;
    }
    printf("Even digits: %d\nOdd digits: %d\n", even, odd);
    return 0;
}
```

2. Sum of Digits

Question: Write a C program that takes a number as input and prints the sum of all its digits.

Answer:

```
#include <stdio.h>

int main() {
    int num, sum = 0;
    scanf("%d", &num);
    while (num != 0) {
        sum += num % 10;
        num /= 10;
    }
    printf("Sum of digits: %d\n", sum);
    return 0;
}
```

3. Check for Armstrong Number

Question: Write a C program to check whether a given number is an Armstrong number. A number is an Armstrong number if the sum of the cubes of its digits equals the number itself.

Answer:

```
#include <stdio.h>

int main() {
    int num, original, remainder, result = 0;
    scanf("%d", &num);
    original = num;
    while (original != 0) {
        remainder = original % 10;
        result += remainder * remainder * remainder;
        original /= 10;
    }
    if (result == num)
        printf("%d is an Armstrong number.\n", num);
    else
        printf("%d is not an Armstrong number.\n", num);
    return 0;
}
```

4. Prime Number Series Sum

Question: Write a C program to calculate the sum of all prime numbers within a given range [L, R], where L and R are provided by the user.

Answer:

```
int isPrime(int n) {
    if (n <= 1) return 0;
    for (int i = 2; i * i <= n; i++) {
        if (n % i == 0) return 0;
    }
    return 1;
}

int main() {
    int L, R, sum = 0;
    scanf("%d %d", &L, &R);
    for (int i = L; i <= R; i++) {
        if (isPrime(i)) sum += i;
    }
    printf("Sum of prime numbers: %d\n", sum);
}
```



Problem Solving with C LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Programmed and Documented by: Prof. Pranjali Thakre

1. Reverse a Number

Question: Write a C program that takes an integer as input and prints its reverse.

Answer:

```
#include <stdio.h>

int main() {
    int num, rev = 0;
    scanf("%d", &num);
    while (num != 0) {
        rev = rev * 10 + num % 10;
        num /= 10;
    }
    printf("Reversed number: %d\\n", rev);
    return 0;
}
```

2. Check for Palindrome Number

Question: Write a C program to check whether a given integer is a palindrome (the same when reversed).

Answer:

```
#include <stdio.h>

int main() {
    int num, original, rev = 0;
    scanf("%d", &num);
    original = num;
    while (num != 0) {
        rev = rev * 10 + num % 10;
        num /= 10;
    }
    if (original == rev)
        printf("%d is a palindrome.\\n", original);
    else
        printf("%d is not a palindrome.\\n", original);
    return 0;
}
```

3. Factorial of a Number

Question: Write a C program to calculate the factorial of a given positive integer using loops.

Answer:

```
#include <stdio.h>

int main() {
    int num;
    long long factorial = 1;
    scanf("%d", &num);
    if (num < 0)
        printf("Factorial of negative numbers doesn't exist.\n");
    else {
        for (int i = 1; i <= num; ++i) factorial *= i;
        printf("Factorial of %d = %lld\n", num, factorial);
    }
    return 0;
}
```

4. Fibonacci Prime Series Sum

Question: Write a C program to compute the sum of all Fibonacci prime numbers up to a given limit provided by the user.

Answer:

```
#include <stdio.h>

int isPrime(int num) {
    if (num <= 1) return 0;
    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) return 0;
    }
    return 1;
}

int main() {
    int limit, prev = 0, curr = 1, next, sum = 0;
    scanf("%d", &limit);
    while (curr <= limit) {
        if (isPrime(curr)) sum += curr;
        next = prev + curr;
        prev = curr;
        curr = next;
    }
    printf("Sum of Fibonacci prime numbers up to %d is %d\n", limit, sum);
    return 0;
}
```



**Problem Solving with C
LABORATORY MANUAL
Week 4**

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

To Learn and Solve	Programs on Nested Control structures
---------------------------	---------------------------------------

- 1) Write a C program that takes an integer n ($1 \leq n \leq 10$) as input. The program should print an $n \times n$ multiplication table using nested **for** loops.

Solution:

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

```
int main()
{
    int n;
    scanf("%d", &n);

    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
        {
            printf("%d\t", i * j);
        }
        printf("\n");
    }

    return 0;
}
```

2. Write a C program that takes a long long integer as input. The program should print the hexadecimal representation of the absolute value of the input along with the counts of vowels, consonants, and digits in its hexadecimal form. Use if-else statements and logical operators in your code.

Solution:

```
#include <stdio.h>

int main()
{
    long long int number;
    scanf("%lld", &number);
    int vowels = 0, consonants = 0, digits = 0;
    long long int temp = (number < 0) ? -number : number;
    printf("%llx\n", temp);
    while (temp != 0)
    {
        int digit = temp % 16;
        temp /= 16;

        if (digit == 10 || digit == 14)
            vowels++;

        else if (digit > 10)
            consonants++;
        else
            digits++;
    }

    printf("%d,%d,%d\n", vowels, consonants, digits);
    return 0;
}
```

3. Write a C program that takes an integer n ($1 \leq n \leq 10$) as input, calculates its factorial, and counts how many unique digits are present in its factorial result. Use nested loops and conditional checks.

```
#include <stdio.h>

int main() {
    int n, i, j, factorial = 1, temp, digit, uniqueCount = 0;
    int checkedDigits = 0; // Variable to track checked digits using bitwise flags

    // Taking input
    printf("Enter an integer (1 to 10): ");
    scanf("%d", &n);

    // Calculating factorial
    for (i = 1; i <= n; i++) {
        factorial *= i;
    }

    temp = factorial; // Storing factorial in a temp variable
```

```
// Counting unique digits
while (temp > 0) {
    digit = temp % 10; // Extracting last digit

    // Checking if the digit has been counted before using bitwise flag
    if ((checkedDigits & (1 << digit)) == 0) {
        uniqueCount++;
        checkedDigits |= (1 << digit); // Marking the digit as counted
    }

    temp /= 10; // Removing last digit
}

// Output results
printf("Factorial of %d is %d\n", n, factorial);
printf("Number of unique digits in factorial: %d\n", uniqueCount);

return 0;
}
```

4. Create a pattern printer (using switch case and loops). Take user input for the choice of pattern to be printed. Only valid choice is 1. If the user's choice is 1, print Floyd's triangle with numbers. Also take the number of rows as input from the user. For any other number for the user's choice print Invalid.

Solution:

```
#include<stdio.h>

int main()
{
    int choice;
    scanf("%d", &choice);

    switch (choice)
    {
        case 1:
        {
            int rows, i, j, number = 1;
            scanf("%d", &rows);
            for (i = 1; i <= rows; i++)
            {
                for (j = 1; j <= i; ++j)
                {
                    printf("%d ", number);
                    ++number;
                }
                printf("\n");
            }
            break;
        }

        default:
            printf("Invalid");
    }

    return 0;
}
```



Problem Solving with C

LABORATORY MANUAL

Week 4

Semester: 2

Course Code: UE23CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by: Prof. Pranjali Thakre

To Learn and Solve	Programs on Nested Control structures
---------------------------	---------------------------------------

- 1) Print reverse pascal's Triangle in pyramid shape. Take the number of rows as input from the user. Accept only positive input, handle non-positive input cases.

Solution:

```
#include <stdio.h>

int main()
{
    int numRows;
    scanf("%d", &numRows);

    if (numRows <= 0)
    {
        printf("Invalid input\n");
        return 0;
    }

    for (int i = numRows - 1; i >= 0; --i)
    {
        for (int space = 0; space < numRows - i - 1; ++space)
            printf(" ");

        for (int j = 0; j <= i; ++j)
        {
            long coefficient = 1;
            for (int k = 1; k <= j; ++k)
            {
                coefficient *= (i - k + 1);
                coefficient /= k;
            }
            printf("%ld ", coefficient);
        }
        printf("\n");
    }
    return 0;
}
```

- 2) Write a C program that takes an integer as input, counts the number of digits in it and displays the same, and then checks if the (sum of its digits + the product of its digits) is equal to the original number. If the conditions are met, print the number; otherwise, display zero.

Solution:

```
#include<stdio.h>

int main()
{
    int num, originalNum, digit, sum = 0, product = 1, count = 0;
    scanf("%d", &num);
    originalNum = num;

    while(num != 0)
    {
        digit = num % 10;
        sum += digit;
        product *= digit;
        num /= 10;
        count++;
    }

    printf("%d\n", count);
    if ((sum + product) == originalNum)
    {
        printf("%d\n", originalNum);
    }
    else
        printf("0\n");
    return 0;
}
```

3. Write a C program to find the largest among three numbers using nested if-else statements.

Solution:

```
#include <stdio.h>

int main() {
    int num1, num2, num3;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &num1, &num2, &num3);

    if (num1 >= num2) {
        if (num1 >= num3) {
            printf("%d is the largest number.\n", num1);
        } else {
            printf("%d is the largest number.\n", num3);
        }
    } else {
        if (num2 >= num3) {
            printf("%d is the largest number.\n", num2);
        } else {
            printf("%d is the largest number.\n", num3);
        }
    }

    return 0;
}
```

Department of CSE, PES University
UE23CS151B - Problem Solving with C
Laboratory-Week 4

4. Write a C program using bitwise operators for the following : Input a number and the bit position (starting from 0th position)
- i) check whether specified bit is set or not
 - If the bit is set, print "BIT IS SET" Else, "BIT IS NOT SET"
 - ii) set the specified bit (if it is not set) and print the result in new line
 - iii) clear the specified bit (if it is set) and print the result in new line

Solution:

```
#include <stdio.h>

int main()
{
    int number, bitPosition;
    scanf("%d", &number);
    scanf("%d", &bitPosition);

    if (( number & (1 << bitPosition)) != 0)
    {
        printf("BIT IS SET\n");
    }
    else
    {
        printf("BIT IS NOT SET\n");
    }

    int setNumber = number | (1 << bitPosition);
    printf("%d\n", setNumber);

    int clearNumber = number & ~(1 << bitPosition);
    printf("%d\n", clearNumber);
    return 0;
}
```



Department of CSE, PES University
UE23CS151B - Problem Solving with C
Laboratory-Week 4



Problem Solving

with C

LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and Documented by – Prof. Pranjali Thakre

To Learn and Solve: Programs on Nested Control Structures

- 1. Write a C program that takes an integer n ($1 \leq n \leq 10$) as input and prints a right-angled triangle of numbers. The triangle should have n rows, with each row containing numbers from 1 to the row number. Use nested loops.**

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

```
int main() {
    int n;
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }

    return 0;
}
```

- 2. Write a C program that takes an integer n ($1 \leq n \leq 10$) as input and prints a checkerboard pattern using '*' and '-'. Use nested loops.**

Enter an integer (1 to 10): 5

```
* - * - *
- * - * -
* - * - *
- * - * -
* - * - *
```

Solution :

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

```
int main() {
    int n;
    scanf("%d", &n);

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if ((i + j) % 2 == 0)
                printf("* ");
            else
                printf("- ");
        }
    }
}
```

```

    }
    printf("\n");
}

return 0;
}

```

- 3. Write a C program that takes an integer n ($1 \leq n \leq 10$) as input and prints all prime numbers up to n^2 using nested loops.**

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

```

int main() {
    int n;
    scanf("%d", &n);

    for (int num = 2; num <= n * n; num++) {
        int is_prime = 1;
        for (int div = 2; div * div <= num; div++) {
            if (num % div == 0) {
                is_prime = 0;
                break;
            }
        }
        if (is_prime)
            printf("%d ", num);
    }
    printf("\n");

    return 0;
}

```

- 4. Develop a C program to implement a versatile unit converter for length measurements. The program should prompt the user to input the type of conversion and the value to be converted. The supported conversion types and their respective conversion values are as follows:**

- a) Feet to Meters: 1 foot = 0.3048 meters

- b) **Miles to Kilometers:** 1 mile = 1.60934 kilometers
- c) **Inches to Centimeters:** 1 inch = 2.54 centimeters
- d) **Yards to Meters:** 1 yard = 0.9144 meters
- e) **Nautical Miles to Meters:** 1 nautical mile = 1852 meters

The program should perform the selected conversion and display the result along with the corresponding units.

Solution:

```
#include <stdio.h>

int main() {
    int choice;
    double inputValue, result;

    // Prompt the user to enter the type of conversion
    scanf("%d", &choice);

    // Prompt the user to enter the value to convert
    scanf("%lf", &inputValue);

    // Perform the conversion based on the selected option
    switch (choice) {
        case 1:
            // Convert Feet to Meters: 1 foot = 0.3048 meters
            result = inputValue * 0.3048;
            printf("%.2lf ft = %.2lf m\n", inputValue, result);
            break;
        case 2:
            // Convert Miles to Kilometers: 1 mile = 1.60934 kilometers
            result = inputValue * 1.60934;
            printf("%.2lf mi = %.2lf km\n", inputValue, result);
            break;
        case 3:
            // Convert Inches to Centimeters: 1 inch = 2.54 centimeters
            result = inputValue * 2.54;
            printf("%.2lf in = %.2lf cm\n", inputValue, result);
            break;
        case 4:
    }
}
```

```
// Convert Yards to Meters: 1 yard = 0.9144 meters
result = inputValue * 0.9144;
printf("%.2lf yd = %.2lf m\n", inputValue, result); break;

case 5:
    // Convert Nautical Miles to Meters: 1 nautical mile = 1852 meters result =
    inputValue * 1852.0;

    printf("%.2lf nmi = %.2lf m\n", inputValue, result); break;

default:
    printf("Invalid choice.\n"); return
    1;

}
return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

- 1) Write a C program that takes a long long integer as input. The program should print the hexadecimal representation of the absolute value of the input along with the counts of vowels, consonants, and digits in its hexadecimal form. Use if-else statements and logical operators in your code.

Solution:

```
#include <stdio.h>

int main()
{
    long long int number;
    scanf("%lld", &number);
    int vowels = 0, consonants = 0, digits = 0;
    long long int temp = (number < 0) ? -number : number;
    printf("%llx\n", temp);
    while (temp != 0)
    {
        int digit = temp % 16;
        temp /= 16;

        if (digit == 10 || digit == 14)
            vowels++;

        else if (digit > 10)
            consonants++;
        else
            digits++;

    }

    printf("%d,%d,%d\n", vowels, consonants, digits);

    return 0;
}
```

2. Write a C program using nested for loops to print the following pattern:

```
1
2 3
4 5 6
7 8 9 10
```

```

#include <stdio.h>

int main() {
    int i, j, num = 1;

    // Outer loop for rows
    for (i = 1; i <= 4; i++) {
        // Inner loop for columns
        for (j = 1; j <= i; j++) {
            printf("%d ", num);
            num++; // Increment the number
        }
        printf("\n"); // Move to the next line
    }

    return 0;
}

```

3. Write a C program using a nested for loop to check whether a given number is prime or not.

```

#include <stdio.h>

int main() {
    int num, i, isPrime = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num <= 1) {
        printf("The number is not prime.\n");
    } else {
        for (i = 2; i * i <= num; i++) {
            if (num % i == 0) {
                isPrime = 0;
                break;
            }
        }

        if (isPrime) {
            printf("The number is prime.\n");
        } else {
            printf("The number is not prime.\n");
        }
    }

    return 0;
}

```

4. Write a C program to perform simple tasks based on operators. Use switch case statements.
- If \$ - Take next 3 floating point numbers input as P, R and T respectively and calculate simple interest
 - If ! - Take next 2 integers and perform arithmetic operations, that is, +, -, *, /. Take the operator also as input.
 - If ? - Take next 2 integers and perform bitwise operations, that is, &, |, Left shift (<) and Right shift(>). Take operator also as input.

Solution:

```
#include <stdio.h>
int main()
{
    char choice;
    scanf(" %c", &choice);

    switch(choice)
    {
        case '$':
        {
            float P, R, T;
            scanf("%f %f %f", &P, &R, &T);

            float simple_interest = (P * R * T) / 100.0;
            printf("%.2f\n", simple_interest);
        }
        break;

        case '!':
        {
            int num1, num2;
            char operator;
            scanf("%d %d %c", &num1, &num2, &operator);
            int result;

            switch(operator)
            {
                case '+':

```



```

        result = num1 - num2;
        printf("%d\n", result);
        break;
    case '*':
        result = num1 * num2;
        printf("%d\n", result);
        break;
    case '/':
        if (num2 == 0)
        {
            printf("Error\n");
            return 0;
        }
        float res;
        res = (float)num1 / (float)num2;
        printf("%.2f\n", res);
        break;
    default:
        printf("Invalid operator\n");
        return 0;
    }

}

break;

case '?':
{
    int num1, num2;
    char operator;
    scanf("%d %d %c", &num1, &num2, &operator);

    int result;
    switch(operator)
    {

        case '&':
            result = num1 & num2;
            break;
        case '|':
            result = num1 | num2;
            break;
        case '<':
            result = num1 << num2;
            break;
        case '>':
            result = num1 >> num2;
            break;
        default:
            printf("Invalid operator\n");
            return 0;
    }
}

```

```
    printf("%d\n", result);
}

break;

default:
    printf("Invalid choice\n");
    return 0;
}

return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Write a C program using nested for loops to print a right triangle of stars (*), like this:

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

```
#include <stdio.h>  
  
int main() {  
    int i, j;  
  
    for (i = 1; i <= 5; i++) {  
        for (j = 1; j <= i; j++) {  
            printf("*");  
        }  
        printf("\n");  
    }  
  
    return 0;  
}
```

2. Write a C program that prints the Fibonacci sequence up to a given number of terms using nested loops.

```
#include <stdio.h>  
  
int main() {  
    int n;  
  
    // Ask the user for the number of terms in the Fibonacci sequence  
    printf("Enter the number of terms in the Fibonacci sequence: ");  
    scanf("%d", &n);  
  
    // Check if the number of terms is valid  
    if (n <= 0) {  
        printf("Please enter a positive integer.\n");  
        return 1;  
    }  
  
    // Initialize the first two Fibonacci numbers  
    int fib1 = 0, fib2 = 1;  
  
    // Outer Loop to control the number of terms  
    for (int i = 1; i <= n; i++) {  
        // Nested Loop to print Fibonacci numbers up to the i-th term
```

```

for (int j = 1; j <= i; j++) {
    if (j == 1) {
        printf("%d ", fib1); // First term
    } else if (j == 2) {
        printf("%d ", fib2); // Second term
    } else {
        // Fibonacci formula for subsequent terms
        int nextTerm = fib1 + fib2;
        printf("%d ", nextTerm);
        fib1 = fib2;
        fib2 = nextTerm;
    }
}
printf("\n"); // Print a new Line after each row
}

return 0;
}

```

3. Write a C program that uses nested for loops to calculate the sum of the first n natural numbers, where n is provided by the user. The program should also display each number being added step by step.

Sample Input and Output:

Enter the value of n: 5

Step-by-step summation process:

Adding: 1 = 1 (Current Sum: 1)
 Adding: 1 + 2 = 2 (Current Sum: 3)
 Adding: 1 + 2 + 3 = 3 (Current Sum: 6)
 Adding: 1 + 2 + 3 + 4 = 4 (Current Sum: 10)
 Adding: 1 + 2 + 3 + 4 + 5 = 5 (Current Sum: 15)

The sum of the first 5 natural numbers is: 15

Solution:

```

#include <stdio.h>

int main() {
    int n, i, j, sum = 0;

    // Taking user input for 'n'
    printf("Enter the value of n: ");
    scanf("%d", &n);

    // Input validation
    if (n <= 0) {
        printf("Please enter a positive integer.\n");
    }
}

```

```

        return 1; // Exit the program with an error code
    }

    printf("\nStep-by-step summation process:\n");

    // Outer loop to simulate the process of summation
    for (i = 1; i <= n; i++) {
        printf("Adding: ");

        // Inner loop to display numbers being added
        for (j = 1; j <= i; j++) {
            printf("%d", j);
            if (j < i) {
                printf(" + ");
            }
        }

        sum += i; // Accumulate sum
        printf(" = %d (Current Sum: %d)\n", i, sum);
    }

    // Display final sum
    printf("\nThe sum of the first %d natural numbers is: %d\n", n,
sum);

    return 0;
}

```

- 4. Write a C program using a switch-case statement that implements a simple banking system. The program should allow the user to:**

- 1. Check Account Balance**
- 2. Deposit Money**
- 3. Withdraw Money**
- 4. Exit**

The user should be able to perform multiple operations until they choose to exit. Implement appropriate validations, such as ensuring withdrawals do not exceed the available balance.

```

#include <stdio.h>

int main() {
    int choice;
    float balance = 1000.00, amount; // Initial balance

    while (1) { // Infinite loop, breaks when user chooses to exit
        // Display menu options
        printf("\n===== BANKING SYSTEM MENU =====\n");
        printf("1. Check Balance\n");

```

```

printf("2. Deposit Money\n");
printf("3. Withdraw Money\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);

// Switch case for user operations
switch (choice) {
    case 1:
        // Check balance
        printf("\nYour current balance is: $%.2f\n", balance);
        break;

    case 2:
        // Deposit money
        printf("\nEnter amount to deposit: $");
        scanf("%f", &amount);
        if (amount > 0) {
            balance += amount;
            printf("Deposit successful! New balance: $%.2f\n", balance);
        } else {
            printf("Invalid deposit amount!\n");
        }
        break;

    case 3:
        // Withdraw money
        printf("\nEnter amount to withdraw: $");
        scanf("%f", &amount);
        if (amount > 0 && amount <= balance) {
            balance -= amount;
            printf("Withdrawal successful! Remaining balance: $%.2f\n", balance);
        } else if (amount > balance) {
            printf("Insufficient funds! Your balance is $%.2f\n", balance);
        } else {
            printf("Invalid withdrawal amount!\n");
        }
        break;

    case 4:
        // Exit the program
        printf("\nThank you for using our banking system!\n");
        return 0; // Terminates the program

    default:
        printf("\nInvalid choice! Please select a valid option.\n");
}
}

```

```
    return 0;  
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 4

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchors: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Check Perfect Number : A **Perfect Number** is a number whose sum of divisors (excluding itself) equals the number itself.

Sample Input and Output:

Enter a number: 28

28 is a Perfect number.

Solution

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

```
int main() {
    int num, sum = 0;
    printf("Enter a number: ");
    scanf("%d", &num);

    for (int i = 1; i < num; i++) { // Find divisors
        if (num % i == 0) { // Check if `i` is a divisor
            sum += i;
        }
    }

    if (sum == num) {
        printf("%d is a Perfect number.\n", num);
    } else {
        printf("%d is not a Perfect number.\n", num);
    }

    return 0;
}
```

2. Write a C program using nested for loops to print the following number pattern:

```
1
22
333
4444
55555
```

```
#include <stdio.h>

int main() {
    int i, j;

    for (i = 1; i <= 5; i++) {
        for (j = 1; j <= i; j++) {
            printf("%d", i);
        }
        printf("\n");
    }
}
```

```
    }

    return 0;
}
```

3. Print a hollow square of n x n size.

Example (n = 5):

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

Solution:

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

```
int main() {
    int n;
    printf("Enter size: ");
    scanf("%d", &n);

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

4. Write a C program that takes two integers as input and determines the type of triangle they could represent based on their sum (for an equilateral, isosceles, or scalene classification). Implement this using nested switch statements.

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

```
int main() {

    int angle1, angle2, angle3;

    // Input two angles
    printf("Enter two angles of the triangle: ");
```

```
scanf("%d %d", &angle1, &angle2);

// Compute the third angle
angle3 = 180 - (angle1 + angle2);

// Check if the triangle is valid
if (angle1 <= 0 || angle2 <= 0 || angle3 <= 0) {
    printf("Invalid triangle. Angles must be positive and sum to 180.\n");
    return 0;
}

printf("The third angle is: %d\n", angle3);

// Outer switch - Based on first two angles
switch (angle1 == angle2) {
    case 1: // First two angles are equal
        switch (angle2 == angle3) {
            case 1: // All three angles are equal
                printf("The triangle is Equilateral.\n");
                break;
            case 0: // Only first two are equal
                printf("The triangle is Isosceles.\n");
                break;
        }
        break;

    case 0: // First two angles are NOT equal
        switch (angle1 == angle3 || angle2 == angle3) {
            case 1: // Two angles are equal
                printf("The triangle is Isosceles.\n");
                break;
        }
}
```

```
case 0: // All three angles are different
    printf("The triangle is Scalene.\n");
    break;
}
break;

}

return 0;
}
```



Problem Solving with

C LABORATORY

MANUAL

Week 4 Extra Questions

Semester: 2

Course Code: UE24CS151B

Course Anchor: Sindhu R Pai

Lab Anchor: Pranjali Thakre

Session: Feb 2025 – June 2025

Program and documented by : Prof. Pranjali Thakre

1. Check If a Number is Special (Sum of Factorial of Digits)

A number is special if the sum of the factorial of its digits equals the number itself.

◆ Example: $145 = 1! + 4! + 5! = 1 + 24 + 120$

Solution:

```
#include <stdio.h>
int main() {
    int num, temp, digit, fact, sum = 0;
    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;
    while (temp > 0) {
        digit = temp % 10;
        fact = 1;

        for (int i = 1; i <= digit; i++)
            fact *= i;

        sum += fact;
        temp /= 10;
    }

    if (sum == num)
        printf("%d is a Special number\n", num);
    else
        printf("%d is NOT a Special number\n", num);

    return 0;
}
```

2. Hollow Rhombus Pattern

For $n = 5$, output:

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

Solution:

```
#include <stdio.h>
int main() {
    int n;
```

```

printf("Enter size: ");
scanf("%d", &n);

for (int i = 1; i <= n; i++) {
    for (int j = 1; j <= n - i; j++)
        printf(" ");

    for (int j = 1; j <= n; j++) {
        if (i == 1 || i == n || j == 1 || j == n)
            printf("*");
        else
            printf(" ");
    }
    printf("\n");
}

return 0;
}

```

3. Nested if-else to Classify Triangle

Solution:

```

#include <stdio.h>
int main() {
    int a, b, c;
    printf("Enter three sides of a triangle: ");
    scanf("%d %d %d", &a, &b, &c);

    if (a + b > c && a + c > b && b + c > a) {
        if (a == b && b == c)
            printf("Equilateral Triangle\n");
        else if (a == b || b == c || a == c)
            printf("Isosceles Triangle\n");
        else
            printf("Scalene Triangle\n");
    } else {
        printf("Not a Valid Triangle\n");
    }

    return 0;
}

```

4. The Magic Elevator: You enter a mystical building with an elevator that operates in a strange way:

- **If the floor number is even, you move up two floors.**
- **If the floor number is odd, you move down one floor.**

- If the floor number is a multiple of 5, you get teleported to the ground floor.

Problem Statement:

Write a program that simulates the elevator movement until it reaches floor 0 or 10. The user enters a starting floor, and the program calculates the next floor until the journey ends.

Solution:

```
#include <stdio.h>

int main() {
    int floor;
    printf("Enter starting floor: ");
    scanf("%d", &floor);

    while (floor > 0 && floor < 10) {
        printf("You are on floor %d\n", floor);

        if (floor % 5 == 0)
            floor = 0; // Teleport to ground
        else if (floor % 2 == 0)
            floor += 2; // Move up
        else
            floor -= 1; // Move down
    }

    printf("Elevator stopped at floor %d\n", floor);
    return 0;
}
```

5. The Secret Treasure Code: You find an ancient treasure chest with a lock that opens if the sum of its digits is even and the product of digits is odd.

Problem Statement:

Write a program that takes a 4-digit number as input, calculates the sum and product of its digits, and checks if it meets the unlocking conditions.

Solution:

```
#include <stdio.h>
int main() {
    int num, sum = 0, product = 1, digit;
    printf("Enter a 4-digit number: ");
    scanf("%d", &num);

    int temp = num;
    while (temp > 0) {
        digit = temp % 10;
        sum += digit;
```

```

        product *= digit;
        temp /= 10;
    }

    if (sum % 2 == 0 && product % 2 != 0)
        printf("The treasure chest unlocks!\n");
    else
        printf("The chest remains locked.\n");

    return 0;
}

```

6. The Haunted Castle Doors: You are trapped in a haunted castle with three doors:

- 1. Red Door – Leads to a fire trap (if your age is below 18, you survive).**
- 2. Blue Door – Leads to a bottomless pit (if your weight is above 50kg, you survive).**
- 3. Green Door – Leads to freedom (but only if today's date is odd).**

Problem Statement:

Write a program that asks for your age, weight, and date and determines your fate.

Solution:

```
#include <stdio.h>

int main() {
    int choice, age, weight, date;
    printf("Enter your age: ");
    scanf("%d", &age);
    printf("Enter your weight (kg): ");
    scanf("%d", &weight);
    printf("Enter today's date (1-31): ");
    scanf("%d", &date);

    printf("Choose a door: 1. Red 2. Blue 3. Green\n");
    scanf("%d", &choice);

    if (choice == 1) {
        if (age < 18) printf("You survive the fire trap!\n");
        else printf("You are burned alive!\n");
    }
    else if (choice == 2) {
        if (weight > 50) printf("You survive the pit!\n");
        else printf("You fall into darkness!\n");
    }
    else if (choice == 3) {
        if (date % 2 != 0) printf("You escape the castle!\n");
        else printf("The door is locked forever!\n");
    }
}
```

```

    }
else {
    printf("Invalid choice! A ghost consumes you.\n");
}

return 0;
}

```

7. The Time Travel Machine: You find a time machine with four options:

- **Year < 1500 → You enter the Medieval Era**
- **1500 to 1900 → You visit the Industrial Revolution**
- **1900 to 2100 → You reach the Modern Era**
- **Year > 2100 → You see the Futuristic World**

Solution:

```

#include <stdio.h>

int main() {
    int year;

    printf("Enter the year you want to travel to: ");
    scanf("%d", &year);

    switch (year / 500) {
        case 0:
        case 1:
        case 2:
            printf("You have traveled to the Medieval Era!\n");
            break;
        case 3:
            printf("You have reached the Industrial Revolution!\n");
            break;
        case 4:
        case 5:
            printf("Welcome to the Modern Era!\n");
            break;
        default:
            printf("You are in a Futuristic World with AI overlords!\n");
    }

    return 0;
}

```

8. Write a program that:

Takes Employee ID, Base Salary, and Job Role as input.

Uses switch-case to determine:

Manager: 30% Bonus

Engineer: 20% Bonus

Technician: 10% Bonus

Uses nested if-else to check if:

Salary after bonus is above 100,000, apply tax deduction.

If salary is below 20,000, give an additional stipend.

Solution:

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

```
int main() {
    int empID;
    float baseSalary, bonus = 0, finalSalary;
    char role;

    // Taking input
    printf("Enter Employee ID: ");
    scanf("%d", &empID);

    printf("Enter Base Salary: ");
    scanf("%f", &baseSalary);

    printf("Enter Job Role (M for Manager, E for Engineer, T for Technician): ");
    scanf(" %c", &role); // Space before %c to avoid newline issue

    // Using switch-case to determine the bonus
    switch (role) {
        case 'M':
        case 'm':
            bonus = 0.30 * baseSalary;
            break;
        case 'E':
        case 'e':
            bonus = 0.20 * baseSalary;
            break;
        case 'T':
        case 't':
            bonus = 0.10 * baseSalary;
            break;
        default:
            printf("Invalid Job Role! Exiting...\n");
    }
}
```

```

        return 1;
    }

// Calculate final salary after bonus
finalSalary = baseSalary + bonus;

// Applying tax or stipend using nested if-else
if (finalSalary > 100000) {
    float tax = 0.10 * finalSalary; //
    finalSalary -= tax;
    printf("Salary above 100,000: 10%% tax applied ($%.2f deducted)\n", tax); }
    else if (finalSalary < 20000)
    { float stipend = 2000; // Fixed stipend
    finalSalary += stipend;
    printf("Salary below 20,000: Additional stipend of $2000 given\n"); }
    // Display the final salary
printf("\nEmployee ID: %d\n", empID);
printf("Base Salary: $%.2f\n", baseSalary);
printf("Bonus: $%.2f\n", bonus);
printf("Final Salary after deductions/additions: $%.2f\n", finalSalary);
return 0; }

```

9. The Space Station Fuel Crisis

Problem Statement:

You are piloting a spaceship that needs exactly 100 units of fuel to land safely.

You can refuel in chunks of 10 or 20.

If you exceed 100, the ship explodes.

Solution:

```
#include <stdio.h>

int main() {
    int fuel = 0, choice;

    while (fuel < 100) {
        printf("\nCurrent Fuel: %d\n", fuel);
        printf("Choose fuel amount to add:\n1. +10\n2. +20\n");
        scanf("%d", &choice);

        if (choice == 1) {
            fuel += 10;
        } else if (choice == 2) {
            fuel += 20;
        } else {
```

```

        printf("Invalid choice, try again!\n");
        continue;
    }

    if (fuel > 100) {
        printf("Fuel Overloaded! The spaceship explodes!\n");
        return 0;
    }
}

printf("You landed safely with exactly 100 fuel units!\n");
return 0;
}

```

10. The Wizard's Potion

You mix a potion where:

Ingredient A reacts with B to form poison.

Ingredient A with C gives healing.

Ingredient B alone explodes.

Problem Statement:

Write a nested switch-case program to determine potion effects.

Solution:

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

```

int main() {
    char ingredient1, ingredient2;
    printf("Enter first ingredient (A/B/C): ");
    scanf(" %c", &ingredient1);
    printf("Enter second ingredient (A/B/C): ");
    scanf(" %c", &ingredient2);

    switch (ingredient1) {
        case 'A':
            switch (ingredient2) {
                case 'B': printf("Poison Created!\n"); break;
                case 'C': printf("Healing Potion Created!\n"); break;
                default: printf("Invalid Combination!\n");
            }
            break;
        case 'B':
            printf("Explosion! You are dead.\n");
            break;
        default:
    }
}
```

```

        printf("No reaction.\n");
    }

    return 0;
}

```

11. Hollow Square Pattern

Example (N = 5):

```

* * * * *
*       *
*       *
*       *
* * * * *

```

Solution:

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

```

int main() {
    int n;
    printf("Enter the size of the square: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            if (i == 1 || i == n || j == 1 || j == n)
                printf("* ");
            else
                printf(" ");
        }
        printf("\n");
    }
    return 0;
}

```

12. Hollow Right-Angled Triangle

Example (N = 5):

```

*
* *
*   *
*   *
* * * * *

```

Solution:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the height of the triangle: ");
    scanf("%d", &n);

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

13. Diamond Pattern

Example (N = 5):

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

C Program:

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the number of rows for half diamond: ");
    scanf("%d", &n);

    // Upper half
    for (int i = 1; i <= n; i++) {
        for (int j = i; j < n; j++)
            printf(" ");
        for (int k = 1; k <= (2 * i - 1); k++)
```

```

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

// Lower half
for (int i = n - 1; i >= 1; i--) {
    for (int j = n; j > i; j--)
        printf(" ");
    for (int k = 1; k <= (2 * i - 1); k++)
        printf("*");
    printf("\n");
}
return 0;
}

```

14. Hollow Pyramid Pattern

Example (N = 5):

```

*
*
*
*
*****

```

C Program:

```

#include <stdio.h>

int main() {
    int n;
    printf("Enter the height of the pyramid: ");
    scanf("%d", &n);

    for (int i = 1; i <= n; i++) {
        for (int j = i; j < n; j++)
            printf(" ");
        for (int k = 1; k <= (2 * i - 1); k++) {
            if (k == 1 || k == (2 * i - 1) || i == n)
                printf("*");
            else
                printf(" ");
        }
        printf("\n");
    }
    return 0;
}

```

