SRM Institute of Science and Technology

Delhi – Meerut Road, Sikri Kalan, Ghaziabad, Uttar Pradesh – 201204

Department of Computer Applications

Circular – 2024-25

BCA 1st Sem

Programming for Problem Solving (USA24102J)

Lab Manual

Lab 1: Basic Program

Title: Basic Program

Aim: To write a simple program to display output on the screen.

Procedure:

1. Write a C program using printf to display a message like "Hello, World!".
2. Save the file with a .c extension (e.g., hello.c).
3. Compile the program using a C compiler (e.g., gcc hello.c -o hello).
4. Run the executable file (e.g., ./hello).

Source Code:

```
#include <stdio.h>

int main() {
  printf("Hello, World!\n");
  return 0;
}
```

Input: None

Expected Output:

```
Hello, World!
```

Lab 2: Program using Input and Output Statements

Title: Input and Output Statements

Aim: To write a program to take input from the user and display it.

Procedure:

1. Write a C program using scanf to take input from the user.
2. Use printf to display the entered input.
3. Save, compile, and run the program.

Source Code:

```
#include <stdio.h>

int main() {
  int number;
  printf("Enter an integer: ");
  scanf("%d", &number);
  printf("You entered: %d\n", number);
  return 0;
}
```

Input:

```
10
```

Expected Output:

```
You entered: 10
```

Lab 3: Program using Operators

Title: Operators

Aim: To write a program to perform arithmetic operations using operators.

Procedure:

1. Write a C program to perform addition, subtraction, multiplication, and division.
2. Take two numbers as input from the user.
3. Display the results of each operation.
4. Save, compile, and run the program.

Source Code:

```
#include <stdio.h>

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

  printf("Sum: %d\n", num1 + num2);
  printf("Difference: %d\n", num1 - num2);
  printf("Product: %d\n", num1 * num2);
  if (num2 != 0) {
    printf("Quotient: %.2f\n", (float)num1 / num2);
  } else {
    printf("Cannot divide by zero.\n");
  }
  return 0;
}
```

Input:

```
10 5
```

Expected Output:

```
Sum: 15
Difference: 5
Product: 50
Quotient: 2.00
```

Lab 4: Operators and Expressions

Title: Operators and Expressions

Aim: To evaluate expressions using different types of operators.

Procedure:

1. Write a C program to evaluate expressions involving arithmetic, relational, and logical operators.
2. Take necessary inputs (if any) from the user or define variables.
3. Display the result of the expression.
4. Save, compile, and run.

Source Code:

```
#include <stdio.h>

int main() {
  int a = 10, b = 5, c = 20;
  int result;

  // Arithmetic Operators
  result = a + b * c;  // Precedence: * > +
  printf("Result of a + b * c: %d\n", result); //410

  // Relational and Logical Operators
  if (a > b && a < c) {
    printf("a is greater than b AND less than c\n");
  }

  // Unary Operators
  a++;
  printf("Value of a after increment: %d\n", a); //11

  return 0;
}
```

Input: None

Expected Output:

```
Result of a + b * c: 110
a is greater than b AND less than c
Value of a after increment: 11
```

Lab 5: Control Statements

Title: Control Statements

Aim: To implement programs using control statements like if, else, switch, for, while, and do-while.

Procedure:

1. Write a C program to demonstrate the use of control statements. For example, find the greatest of three numbers using if-else-if.
2. Save, compile, and run the program.

Source Code:

```
#include <stdio.h>

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

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

  if (num1 >= num2 && num1 >= num3) {
    printf("%d is the greatest number.\n", num1);
  } else if (num2 >= num1 && num2 >= num3) {
    printf("%d is the greatest number.\n", num2);
  } else {
    printf("%d is the greatest number.\n", num3);
  }
  return 0;
}
```

Input:

```
25 10 15
```

Expected Output:

```
25 is the greatest number.
```

Lab 6: Arrays - One Dimensional

Title: One-Dimensional Arrays

Aim: To write a program to work with one-dimensional arrays.

Procedure:

1. Write a C program to read elements into an array and perform operations like finding the sum or the largest element.
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>

int main() {
  int n, i;
  printf("Enter the size of the array: ");
  scanf("%d", &n);
  int arr[n]; // Use variable length array

  printf("Enter the elements of the array:\n");
  for (i = 0; i < n; i++) {
    scanf("%d", &arr[i]);
  }

  int sum = 0;
  for (i = 0; i < n; i++) {
    sum += arr[i];
  }
  printf("Sum of the elements: %d\n", sum);

  int max = arr[0];
  for(i=1; i<n; i++){
    if(arr[i] > max){
        max = arr[i];
    }
  }
  printf("Largest Element: %d\n", max);
  return 0;
}
```

Input:

```
5
1 2 3 4 5
```

Expected Output:

```
Sum of the elements: 15
Largest Element: 5
```

Lab 7: Arrays - Multi-dimensional

Title: Multi-dimensional Arrays

Aim: To write a program to work with multi-dimensional arrays (e.g., 2D arrays).

Procedure:

1. Write a C program to perform matrix operations like addition or multiplication using 2D arrays.
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>

int main() {
  int rows, cols, i, j;

  printf("Enter the number of rows: ");
  scanf("%d", &rows);
  printf("Enter the number of columns: ");
  scanf("%d", &cols);

  int matrix[rows][cols];

  printf("Enter the elements of the matrix:\n");
  for (i = 0; i < rows; i++) {
    for (j = 0; j < cols; j++) {
      scanf("%d", &matrix[i][j]);
    }
  }

  printf("The matrix is:\n");
  for (i = 0; i < rows; i++) {
    for (j = 0; j < cols; j++) {
      printf("%d ", matrix[i][j]);
    }
    printf("\n");
  }
  return 0;
}
```

Input:

```
2
2
1 2
3 4
```

Expected Output:

```
The matrix is:
1 2
3 4
```

Lab 8: Strings, Structures, and Union

Title: Strings, Structures, and Union

Aim: To write a program to work with strings, structures, and unions.

Procedure:

1. Write a C program to demonstrate string manipulation (e.g., finding length, copying), create a structure to store student details, and use a union to store different data types.
2. Save, compile, and run.

Source Code:

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

struct Student {
  int rollNumber;
  char name[50];
  float marks;
};

union Data {
  int i;
  float f;
  char str[20];
};

int main() {
  // Strings
  char str[50] = "Hello, World!";
  printf("String: %s\n", str);
  printf("Length of string: %ld\n", strlen(str));

  // Structures
  struct Student student1;
  student1.rollNumber = 101;
  strcpy(student1.name, "John Doe");
  student1.marks = 85.5;

  printf("Student Details:\n");
  printf("Roll Number: %d\n", student1.rollNumber);
  printf("Name: %s\n", student1.name);
  printf("Marks: %.2f\n", student1.marks);

  // Union
  union Data data;
  data.i = 10;
  printf("Integer data: %d\n", data.i);

  data.f = 20.5;
  printf("Float data: %.2f\n", data.f);

  strcpy(data.str, "Hello");
  printf("String data: %s\n", data.str);

  return 0;
}
```

Input: None

Expected Output:

```
String: Hello, World!
Length of string: 13
Student Details:
Roll Number: 101
Name: John Doe
Marks: 85.50
Integer data: 10
Float data: 20.50
String data: Hello
```

Lab 9: Functions

Title: Functions

Aim: To write programs using functions to modularize code.

Procedure:

1. Write a C program to define a function (e.g., to calculate the factorial of a number).
2. Call the function from the main function.
3. Save, compile, and run.

Source Code:

```
#include <stdio.h>

int factorial(int n) {
  if (n == 0)
    return 1;
  else
    return n * factorial(n - 1);
}

int main() {
  int num;
  printf("Enter a non-negative integer: ");
  scanf("%d", &num);
  if(num < 0){
    printf("Invalid input. Please enter a non-negative number.\n");
  }
  else{
    printf("Factorial of %d is %d\n", num, factorial(num));
  }

  return 0;
}
```

Input:

```
5
```

Expected Output:

```
Factorial of 5 is 120
```

Lab 10: Functions

Title: Functions (Continued)

Aim: To further explore functions, including different types of function calls (e.g., call by value, call by reference).

Procedure:

1. Write a C program to demonstrate call by value and call by reference.
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>

void callByValue(int x) {
  x = x + 10;
  printf("Inside callByValue, x = %d\n", x);
}

void callByReference(int *y) {
  *y = *y + 10;
  printf("Inside callByReference, *y = %d\n", *y);
}

int main() {
  int a = 5, b = 5;

  printf("Before function calls, a = %d, b = %d\n", a, b);

  callByValue(a);
  printf("After callByValue, a = %d\n", a);

  callByReference(&b);
  printf("After callByReference, b = %d\n", b);

  return 0;
}
```

Input: None

Expected Output:

```
Before function calls, a = 5, b = 5
Inside callByValue, x = 15
After callByValue, a = 5
Inside callByReference, *y = 15
After callByReference, b = 15
```

Lab 11: Pointers

Title: Pointers

Aim: To write programs to understand the concept of pointers.

Procedure:

1. Write a C program to declare a pointer, assign the address of a variable to it, and access the variable's value using the pointer.
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>

int main() {
  int num = 10;
  int *ptr;

  ptr = &num;

  printf("Address of num: %p\n", &num);
  printf("Value of num: %d\n", num);

  printf("Address stored in ptr: %p\n", ptr);
  printf("Value pointed to by ptr: %d\n", *ptr);

  *ptr = 20;
  printf("Modified value of num: %d\n", num);

  return 0;
}
```

Input: None

Expected Output: (Note: The address values will vary)

```
Address of num: 0x7ffc34e47c78
Value of num: 10
Address stored in ptr: 0x7ffc34e47c78
Value pointed to by ptr: 10
Modified value of num: 20
```

Lab 12: Pointers

Title: Pointers (Continued)

Aim: To further explore pointers, including pointer arithmetic and using pointers with arrays.

Procedure:

1. Write a C program to demonstrate pointer arithmetic and access array elements using pointers.
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>

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

  printf("Elements of the array using pointer arithmetic:\n");
  for (int i = 0; i < 5; i++) {
    printf("arr[%d] = %d\n", i, *(ptr + i));
  }

  printf("\nElements of the array using pointer and index:\n");
   for (int i = 0; i < 5; i++) {
    printf("arr[%d] = %d\n", i, *(arr + i));
  }

  return 0;
}
```

Input: None

Expected Output:

```
Elements of the array using pointer arithmetic:
arr[0] = 10
arr[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50

Elements of the array using pointer and index:
arr[0] = 10
arr[1] = 20
arr[2] = 30
arr[3] = 40
arr[4] = 50
```

Lab 13: File: Reading and Writing

Title: File Reading and Writing

Aim: To write a program to read from and write to a file.

Procedure:

1. Write a C program to open a file in write mode, write data to it, close the file, open the same file in read mode, read the data, and display it.
2. Save, compile, and run.

Source Code:

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

int main() {
  FILE *fp;
  char filename[] = "myfile.txt";
  char dataToWrite[] = "Hello, this is a test.";
  char dataRead[100];

  // Write to file
  fp = fopen(filename, "w");
  if (fp == NULL) {
    perror("Error opening file for writing");
    return 1;
  }
  fprintf(fp, "%s", dataToWrite);
  fclose(fp);

  // Read from file
  fp = fopen(filename, "r");
  if (fp == NULL) {
    perror("Error opening file for reading");
    return 1;
  }
  fgets(dataRead, sizeof(dataRead), fp);
  printf("Data read from file: %s\n", dataRead);
  fclose(fp);

  return 0;
}
```

Input: None (The program creates and reads from a file)

Expected Output:

```
Data read from file: Hello, this is a test.
```

Lab 14: File Handling: fputw(), fgetw(), remove()

Title: File Handling with fputw(), fgetw(), remove()

Aim: To write a program to use the fputw(), fgetw(), and remove() functions for file handling.

Procedure:

1. Write a C program to write integers to a file using fputw(), read integers from the file using fgetw(), and then delete the file using remove().
2. Save, compile, and run.

Source Code:

```
#include <stdio.h>


#include <stdlib.h>

int main() {
  FILE *fp;
  char filename[] = "numbers.dat";
  int numbers[] = {10, 20, 30, 40, 50};
  int numRead;

  // Write integers to file using fputw()
  fp = fopen(filename, "wb"); // Use "wb" for binary write
  if (fp == NULL) {
    perror("Error opening file for writing");
    return 1;
  }
  for (int i = 0; i < sizeof(numbers) / sizeof(numbers[0]); i++) {
    fputw(numbers[i], fp);
  }
  fclose(fp);

  // Read integers from file using fgetw()
  fp = fopen(filename, "rb"); // Use "rb" for binary read
  if (fp == NULL) {
    perror("Error opening file for reading");
    return 1;
  }
  printf("Numbers read from file:\n");
  while ((numRead = fgetw(fp)) != EOF) {
    printf("%d\n", numRead);
  }
  fclose(fp);

  // Remove the file
  if (remove(filename) == 0) {
    printf("File %s deleted successfully.\n", filename);
  } else {
    perror("Error deleting file");
    return 1;
  }

  return 0;
}
```

Input: None

Expected Output:

```
Numbers read from file:
10
20
30
40
50
File numbers.dat deleted successfully.
```

Lab 15: Creating Macros

Title: Creating Macros

Aim: To write a program to create and use macros.

Procedure:

1. Write a C program to define a macro using the #define preprocessor directive (e.g., a macro to calculate the square of a number).
2. Use the macro in the main function.
3. Save, compile, and run.

Source Code:

```
#include <stdio.h>

#define SQUARE(x) ((x) * (x))

int main() {
  int num = 5;
  int result;

  result = SQUARE(num);
  printf("Square of %d is %d\n", num, result);

  result = SQUARE(num + 2); // Important to parenthesize macro parameters
  printf("Square of %d is %d\n", num + 2, result);

  return 0;
}
```

Input: None

Expected Output:

```
Square of 5 is 25
      Square of 7 is 49
```