Subject Name & Code:

PROGRAMMING FOR PROBLEM SOLVING- BE01R00121

SELF LEARNING SOLUTION

S.L – 1 (CO-3)

Topic: Develop simple programs using appropriate data structures and standard libraries

Q-1: Write a program to swap number using functions.

Answer:

```
#include <stdio.h>

void swap(int *a, int *b);

int main() {
    int num1, num2;

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

    printf("Before swapping: num1 = %d, num2 = %d\n", num1, num2);

    swap(&num1, &num2);

    printf("After swapping: num1 = %d, num2 = %d\n", num1, num2);
```

```
    return 0;
}

void swap(int *a, int *b) {
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

Explanation: This program uses **call by reference**. The addresses of the variables are passed to the function swap. Inside the function, the values at these addresses are interchanged using a temporary variable. This change is reflected in the main function.

Q-2: Write a program to print fibonacci series using functions.

Answer:

```
#include <stdio.h>

void printFibonacci(int n);

int main() {
    int n;

    printf("Enter the number of terms: ");
    scanf("%d", &n);
```

```
    printf("Fibonacci Series: ");
    printFibonacci(n);

    return 0;
}

void printFibonacci(int n) {
    int i, t1 = 0, t2 = 1, nextTerm;

    for (i = 1; i <= n; ++i) {
        printf("%d, ", t1);
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;
    }
}
```

Explanation: The function printFibonacci takes the number of terms n as input. It initializes the first two terms and then uses a loop to calculate and print each subsequent term by updating the values of t1 and t2.

Q-3: Construct a C program to add 3X3 matrix.

Answer:

```
#include <stdio.h>

int main() {
```

```
int mat1[3][3], mat2[3][3], sum[3][3];
int i, j;

printf("Enter elements of first 3x3 matrix:\n");
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        scanf("%d", &mat1[i][j]);
    }
}

printf("Enter elements of second 3x3 matrix:\n");
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        scanf("%d", &mat2[i][j]);
    }
}

// Adding two matrices
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        sum[i][j] = mat1[i][j] + mat2[i][j];
    }
}

printf("\nSum of the two matrices:\n");
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
```

```
            printf("%d ", sum[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```

Explanation: This program uses two-dimensional arrays to store the matrices. It first reads the elements of both matrices from the user. Then, it adds corresponding elements from mat1 and mat2 and stores the result in the sum matrix. Finally, it prints the resulting sum matrix.

Q-4: Write a program to store10 elements in array given by user and to find maximum out of those10 elements.

Answer:

```
#include <stdio.h>

int main() {
    int arr[10];
    int i, max;

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

    max = arr[0]; // Assume first element is the maximum
```

```
    for (i = 1; i < 10; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    printf("The maximum element is %d\n", max);

    return 0;
}
```

Explanation: The program stores 10 integers in an array arr. It then initializes a variable max with the first element of the array. It iterates through the rest of the array, comparing each element with the current max. If a larger element is found, max is updated.

Q-5: Write a C program to sort an array in ascending order.

Answer:

```
#include <stdio.h>

int main() {
    int arr[100];
    int n, i, j, temp;

    printf("Enter the number of elements: ");
    scanf("%d", &n);
```

```
    printf("Enter %d integers: ", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Bubble Sort algorithm
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap arr[j] and arr[j+1]
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }

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

    return 0;
}
```

Explanation: This program implements the **Bubble Sort** algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process is repeated until the entire array is sorted. The outer loop controls the number of passes, and the inner loop performs the comparison and swapping.

Q-6: Write a program to check whether entered number is prime or not with the help of user-defined function check-prime ().

Answer:

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

bool check_prime(int n);

int main() {
    int num;

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

    if (check_prime(num)) {
        printf("%d is a prime number.\n", num);
    } else {
        printf("%d is not a prime number.\n", num);
    }

    return 0;
}

bool check_prime(int n) {
    int i;
```

```
    if (n <= 1) {
        return false;
    }

    for (i = 2; i <= n/2; i++) {
        if (n % i == 0) {
            return false;
        }
    }

    return true;
}
```

Explanation: The user-defined function check_prime takes an integer n as an argument. It first checks if the number is less than or equal to 1 (not prime). Then, it checks for divisibility from 2 up to n/2. If any number in this range divides n perfectly, the function returns false (not prime). If no divisors are found, it returns true (prime).

S.L – 2 (CO-3)

Topic: Develop simple programs using appropriate data structures and standard libraries

Q-1: Write a program in c using structure to enter roll no, marks of the three subject for 3 student and find total obtained by each student.

Answer:

```
#include <stdio.h>

struct Student {
    int roll_no;
    int marks_sub1;
    int marks_sub2;
    int marks_sub3;
    int total_marks;
};

int main() {
    struct Student s[3];
    int i;

    // Input data for 3 students
    for (i = 0; i < 3; i++) {
        printf("\nEnter details for Student %d:\n", i + 1);
```

```
        printf("Roll No: ");
        scanf("%d", &s[i].roll_no);
        printf("Marks in Subject 1: ");
        scanf("%d", &s[i].marks_sub1);
        printf("Marks in Subject 2: ");
        scanf("%d", &s[i].marks_sub2);
        printf("Marks in Subject 3: ");
        scanf("%d", &s[i].marks_sub3);

        // Calculate total marks
        s[i].total_marks = s[i].marks_sub1 + s[i].marks_sub2 +
s[i].marks_sub3;
    }

    // Display the total marks for each student
    printf("\n--- Student Total Marks ---\n");
    for (i = 0; i < 3; i++) {
        printf("Roll No: %d, Total Marks: %d\n", s[i].roll_no,
s[i].total_marks);
    }

    return 0;
}
```

Explanation: A structure Student is defined to hold the student's data. An array of this structure is created for three students. A loop is used to input the data for each student, and the total marks are

calculated and stored within the structure itself. Finally, another loop prints the roll number and total marks for each student.

Q-2: Define a structure data type called time_struct containing three member's integer hours, minutes, second.

Develop a program that would assign values to individual member and is play the time in following format :

: HH:MM:SS

Answer:

```
#include <stdio.h>

struct time_struct {
    int hours;
    int minutes;
    int seconds;
};

int main() {
    struct time_struct t;

    // Assign values to individual members
    printf("Enter hours: ");
    scanf("%d", &t.hours);
    printf("Enter minutes: ");
```

```
    scanf("%d", &t.minutes);
    printf("Enter seconds: ");
    scanf("%d", &t.seconds);

    // Display the time in the format HH:MM:SS
    printf("\nThe time is: %02d:%02d:%02d\n", t.hours, t.minutes,
t.seconds);

    return 0;
}
```

Explanation: The structure time_struct is defined with three integer members. The program takes input for each member from the user. The printf function uses %02d to format the output, ensuring that each component (hours, minutes, seconds) is displayed as two digits, with a leading zero if necessary.

Q-3: Define a Structure which contains details of a Cricketer:

- **Name of Player:**
- **Team name:**
- **Total run scored:**

Batting average:

Answer:

```
#include <stdio.h>

struct Cricketer {
```

```
    char player_name[50];
    char team_name[50];
    int total_runs;
    float batting_average;
};

int main() {
    struct Cricketer player;

    // Input details of the cricketer
    printf("Enter Player's Name: ");
    scanf(" %[^\n]%*c", player.player_name); // To read string with spaces
    printf("Enter Team Name: ");
    scanf(" %[^\n]%*c", player.team_name);
    printf("Enter Total Runs Scored: ");
    scanf("%d", &player.total_runs);
    printf("Enter Batting Average: ");
    scanf("%f", &player.batting_average);

    // Display the details of the cricketer
    printf("\n--- Cricketer Details ---\n");
    printf("Player Name: %s\n", player.player_name);
    printf("Team Name: %s\n", player.team_name);
    printf("Total Runs: %d\n", player.total_runs);
```

```
    printf("Batting Average: %.2f\n", player.batting_average);

    return 0;
}
```

Explanation: A structure Cricketer is defined to hold all the required details. The %[^\n]%*c format specifier in scanf is used to read strings that contain spaces (like a full name). After taking all inputs, the program prints the stored information in a formatted manner.

Q-4: Write a C program to copy content of one file to other with the help of file handling functions.

Answer:

```
#include <stdio.h>

int main() {
    FILE *sourceFile, *targetFile;
    char sourceFilename[100], targetFilename[100];
    char ch;

    // Get the source and target file names from the user
    printf("Enter the source file name: ");
    scanf("%s", sourceFilename);
    printf("Enter the target file name: ");
    scanf("%s", targetFilename);
```

```
    // Open the source file in read mode
    sourceFile = fopen(sourceFilename, "r");
    if (sourceFile == NULL) {
        printf("Cannot open source file %s\n", sourceFilename);
        return 1;
    }

    // Open the target file in write mode
    targetFile = fopen(targetFilename, "w");
    if (targetFile == NULL) {
        printf("Cannot open target file %s\n", targetFilename);
        fclose(sourceFile);
        return 1;
    }

    // Copy content character by character
    while ((ch = fgetc(sourceFile)) != EOF) {
        fputc(ch, targetFile);
    }

    printf("File copied successfully from %s to %s.\n",
sourceFilename, targetFilename);

    // Close both files
```

```
    fclose(sourceFile);
    fclose(targetFile);

    return 0;
}
```

Explanation: This program uses standard file handling functions. It opens the source file in read mode ("r") and the target file in write mode ("w"). It checks if the files were opened successfully to avoid errors. It then uses a loop to read each character (fgetc) from the source file until the End Of File (EOF) is reached and immediately writes that character (fputc) to the target file. Finally, it closes both files.