C PROGRAMMING TEST CODES:

Below are some codes you can try to run to see how they work. These codes cover arrays, pointers, memory allocation and deallocation.

```c
#include <stdio.h>

int main() {
    // Creating an array
    int my_array[5] = {1, 2, 3, 4, 5};

    // Accessing elements in the array
    printf("First element: %d\n", my_array[0]);  // Prints: 1
    printf("Second element: %d\n", my_array[1]);  // Prints: 2
    printf("Third element: %d\n", my_array[2]);   // Prints: 3

    // Modifying elements in the array
    my_array[0] = 10;
    printf("Modified first element: %d\n", my_array[0]);  // Prints: 10

    // Iterating through the array
    printf("Array elements:\n");
    for (int i = 0; i < 5; i++) {
        printf("%d\n", my_array[i]);
    }

    return 0;
}
```

OUTPUT:

First element: 1

Second element: 2

Third element: 3

Modified first element: 10

Array elements:

10

2

3

4

5

This code creates an array my_array with elements [1, 2, 3, 4, 5], then it accesses and modifies elements of the array and iterates through all elements printing them out.

```c
#include <stdio.h>

int main() {
    int num = 10;  // Declare an integer variable
    int *ptr;      // Declare a pointer to an integer

    ptr = &num;    // Assign the address of num to the pointer

    // Accessing variable through pointer
    printf("Value of num: %d\n", *ptr);  // Prints: 10

    // Modifying variable through pointer
    *ptr = 20;
    printf("Modified value of num: %d\n", num);  // Prints: 20

    return 0;
}
```

OUTPUT:

Value of num: 10

Modified value of num: 20

In this code, ptr is a pointer to an integer. We assign the address of the integer variable num to the pointer using the address-of operator &. Then, we access the value of num through the pointer using the dereference operator *, and modify the value of num through the pointer as well. Finally, we print out the values to verify the changes.

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

int main() {
    int *ptr;

    // Allocate memory dynamically for an integer
    ptr = (int *)malloc(sizeof(int));
    if (ptr == NULL) {
        printf("Memory allocation failed\n");
        return 1;
    }

    // Assign a value to the dynamically allocated memory
    *ptr = 10;

    // Access the value
    printf("Value of dynamically allocated memory: %d\n", *ptr);

    // Deallocate the dynamically allocated memory
    free(ptr);

    return 0;
}
```

OUTPUT:

Value of dynamically allocated memory: 10

In this code:

- We declare a pointer ptr to an integer.
- We use malloc() function to allocate memory dynamically for an integer. The sizeof(int) is used to specify the size of memory to be allocated for one integer.
- We check if the memory allocation was successful or not.
- We assign a value to the dynamically allocated memory using the pointer ptr.
- We access and print the value stored in the dynamically allocated memory.
- Finally, we deallocate the dynamically allocated memory using the free() function to avoid memory leaks.

LAB02-TASK01: Sales Tracking System (5%)

You are tasked with developing a program for a small retail store that sells electronic gadgets. The store manager wants to keep track of the sales made by each employee during the week. The program should allow the manager to input the sales data for each employee and then display the total sales for each employee, the overall total sales for the week, and the employee with the highest sales.

Write a C program to implement the functionality of the sales tracking system described above using functions, arrays, and pointers. Your program should include the following:

- Define a structure named Employee with the following fields: id (integer), name (string), and sales (array of floats to store daily sales for the week).
- Define a function to input sales data for each employee (inputSales) that takes a pointer to an array of Employee structures as a parameter.
- Define a function to calculate the total sales for each employee (calculateTotalSales) that takes a pointer to an array of Employee structures as a parameter.
- Define a function to calculate the overall total sales for the week (calculateOverallTotalSales) that takes a pointer to an array of Employee structures as a parameter.
- Define a function to find and return a pointer to the employee with the highest sales (findEmployeeWithHighestSales) that takes a pointer to an array of Employee structures as a parameter.
- Display the total sales for each employee, the overall total sales for the week, and the employee with the highest sales.

Your program should adhere to the following specifications:

- Use dynamic memory allocation to manage employee data.
- Use pointers to access and manipulate the data.
- Display meaningful messages and ensure that the output is well-formatted.
- Ensure proper memory deallocation at the end of the program.

Sample Output:

```
Welcome to the Sales Tracking System!

Enter the number of employees: 3

Enter details for Employee 1:
ID: 101
Name: John Doe
Enter sales for each day of the week:
```

Day 1: 100
Day 2: 150
Day 3: 200
Day 4: 180
Day 5: 250
Day 6: 300
Day 7: 220

Enter details for Employee 2:
ID: 102
Name: Jane Smith
Enter sales for each day of the week:
Day 1: 120
Day 2: 180
Day 3: 210
Day 4: 190
Day 5: 270
Day 6: 320
Day 7: 240

Enter details for Employee 3:
ID: 103
Name: David Lee
Enter sales for each day of the week:
Day 1: 90
Day 2: 130
Day 3: 180
Day 4: 160
Day 5: 220
Day 6: 260
Day 7: 200

Total Sales for Employee 1 (John Doe): RM1400
Total Sales for Employee 2 (Jane Smith): RM1530
Total Sales for Employee 3 (David Lee): RM1230

Overall Total Sales for the Week: RM4160

Employee with the Highest Sales:

ID: 102

Name: Jane Smith

Total Sales: RM1530

Thank you for using our Sales Tracking System!

Rubric:

	Criteria	Points
MEMORY ALLOCATION/ DEALLOCATION	Successfully allocates memory for the array of Employee structures based on the user input.	5
	Properly handles scenarios where memory allocation fails due to insufficient memory.	5
	Properly deallocates memory for the sales array of each employee.	5
	Properly deallocates memory for the array of Employee structures.	5
FUNCTIONALITY	Function inputSales correctly inputs sales data for each employee.	5
	Other operations performed by the program are correct and functioning as expected.	5
	Handles errors and invalid inputs gracefully, providing appropriate error messages.	5
DOCUMENTATION	Provides sufficient comments to explain the purpose of each function and major code sections.	5
	Uses meaningful variable names that enhance code readability.	5

Lab Task Submission:

1. LAB02-TASK01.c

Zip all your files and save it as LAB02_<GroupName>_<Day>.zip

Example: LAB02_Group1_WED.zip

Group Name
Group1 – Group15

Day
TUES
WED
THURS
SAT