LAB02-TASK01: Array Manipulation with Pointers (1.5%)

Scenario:

You have been assigned the task of manipulating an array of integers using pointers. Your goal is to create a program that performs three operations on the array:

a. Calculate the sum of all the elements.
b. Find the maximum element and its index.
c. Reverse the order of the elements in the array.

Instructions:

1. Create an array of integers with a size of 10 and initialize it with random values.
2. Implement a function called calculateSum that takes the array and its size as parameters and returns the sum of all the elements.
3. Implement a function called findMaximum that takes the array and its size as parameters and returns the maximum element and its index.
4. Implement a function called reverseArray that takes the array and its size as parameters and reverses the order of the elements using pointers.
5. In the main function, call each of the three functions and display the results on the console.

Requirements:

1. Write a program that includes user-defined functions.
2. Implement the calculateSum function to calculate the sum of all elements in the array.
3. Implement the findMaximum function to find the maximum element and its index in the array.
4. Implement the reverseArray function to reverse the order of elements in the array using pointers.
5. Ensure the code is properly commented and follows good programming practices.
6. Test the code with different inputs to ensure its correctness.

Sample Output:

Original Array: 10 25 7 32 18 6 15 21 9 12

Sum of all elements: 155
Maximum element: 32, Index: 3

Reversed Array: 12 9 21 15 6 18 32 7 25 10

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	5
	Code initializes the array with random values	5
	Code implements the calculateSum function correctly and returns the sum of all elements	20
	Code implements the findMaximum function correctly and returns the maximum element and its index	20
	Code implements the reverseArray function correctly and reverses the order of elements in the array using pointers	20
	Code displays the results of each operation correctly on the console	10
READABILITY	Code follows appropriate naming conventions (variables)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
TESTING	Code produces correct outputs for different test cases, covering different arrays and sizes	20
	Code demonstrates a clear understanding and effective use of pointers and array manipulation	10

LAB02-TASK02: Student Record Management with Dynamic Memory Allocation (3.5%)

Scenario:

You have been assigned a complex task to develop an advanced student record management system. The program should allow users to add, display, search, update, and delete student records dynamically. Additionally, the program should utilize arrays, pointers, loops, string manipulation, structures, and dynamic memory allocation to achieve these functionalities.

Instructions:

1. Define a structure named Student that contains the following information about each student: name, roll number, age, and GPA.
2. Implement a function called addStudent that allows users to add a new student to the record. This function should dynamically allocate memory for the new student and prompt the user to enter the student's details. Store the information in the dynamically allocated memory.
3. Implement a function called displayStudents that displays the information of all students in the record. This function should iterate over the dynamically allocated memory and print the details of each student.
4. Implement a function called searchStudent that allows users to search for a student by name. This function should prompt the user to enter the name of the student and then search the dynamically allocated memory to find a match. If found, display the student's details; otherwise, display a message indicating that the student was not found.
5. Implement a function called updateStudent that allows users to update a student's information. This function should prompt the user to enter the roll number of the student to be updated. It should then search for the student in the dynamically allocated memory and allow the user to update the student's details if found.
6. Implement a function called deleteStudent that allows users to delete a student's record. This function should prompt the user to enter the roll number of the student to be deleted. It should then search for the student in the dynamically allocated memory, free the memory occupied by the student's record, and remove the student from the record.
7. In the main function, provide a menu-driven interface that allows users to select various operations, including adding a student, displaying all students, searching for a student, updating a student's details, deleting a student's record, and exiting the program.
8. Handle appropriate error cases, such as invalid user inputs.

Requirements:

1. Write a program that includes user-defined functions and utilizes arrays, pointers, loops, string manipulation, structures, and dynamic memory allocation.
2. Ensure the code is properly commented and follows good programming practices.
3. Test the code with different inputs and scenarios to ensure its correctness.

Sample Output:

Front Page

```
Welcome to Student Record Management System

Menu:

1. Add a student
2. Display all students
3. Search for a student
4. Update student details
5. Delete a student record
6. Exit
```

Case 1: Adding record

```
Enter your choice: 1

Enter student details:
Name: John Doe
Matric Number: U123456
Age: 20
GPA: 3.7

Student added successfully!
```

Case 2: Displaying record.

```
Enter your choice: 2

Student Record:
Name: John Doe
Matric Number: U123456
Age: 20
GPA: 3.7
```

Case 3: Search record

```
Enter your choice: 3

Enter the name of the student to search: John Doe

Student found:
Name: John Doe
Matric Number: U123456
Age: 20
GPA: 3.7
```

Case 4: Update record.

```
Enter your choice: 4

Enter the matric number of the student to update: U123456

Enter updated details:
Name: John Smith
Matric Number: U123456
Age: 21
GPA: 3.9

Student details updated successfully!
```

Case 5: Delete record

```
Enter your choice: 5

Enter the matric number of the student to delete: U123456

Student record deleted successfully!
```

Case 6: Check if record really deleted.

```
Enter your choice: 2

No students found in the record.
```

Case 7: Exit program.

```
Enter your choice: 6

Exiting the program...
```

Rubric:

	Criteria	Points
FUNCTIONALITY	Code compiles without errors and warnings	5
	Code implements the addStudent function correctly, including dynamic memory allocation and storage of student details	20
	Code implements the displayStudents function correctly, iterating over the dynamically allocated memory and printing student details	15
	Code implements the searchStudent function correctly, searching for a student by name in the dynamically allocated memory and displaying details if found	20
	Code implements the updateStudent function correctly, allowing the user to update a student's details in the dynamically allocated memory	20
	Code implements the deleteStudent function correctly, removing a student's record from the dynamically allocated memory	20
	Code provides a menu-driven interface and handles user inputs and program flow correctly	15
READABILITY	Code follows appropriate naming conventions (variables, functions etc)	5
	Code includes proper indentation and formatting	5
	Code includes comments to explain the logic and enhance understanding	5
	Code is well-structured and organized with meaningful code blocks	5
	Code does not contain unnecessary or redundant statements	5
MEMORY MGMT	Code dynamically allocates memory for student records as required	10
	Code frees the memory correctly when deleting a student's record	10
TESTING	Code produces correct outputs for different test cases, covering different operations and scenarios	20
	Code demonstrates a clear understanding and effective use of arrays, pointers, loops, string manipulation, structures, and dynamic memory allocation	15

Lab Task Submission:

1. LAB02-TASK01.c
2. LAB02-TASK02.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