Module 03 - Unit 4: Lab

Objectives

- **Write C programs** involving dynamic memory allocation and pointer manipulation.
- **Implement functions** without using array-based operations.
- **Manage dynamic memory** effectively to prevent memory leaks.
- **Identify and correct C compiler error messages and warnings**.

General Instructions

1. **Identification:**
    - Do **not** include your email or user ID in the program.
    - Use your UNC Charlotte 800# to identify your code if necessary.
2. **File Management:**
    - Each program should be in its own source code file with a `.c` extension.
    - Test your code thoroughly before submission.
    - Ensure your program compiles without any errors or warnings to receive full credit. You may comment out lines that have errors to obtain partial credit.
3. **Academic Integrity:**
    - **Do your own work.** Do not use external resources such as the Internet, AI, friends, etc.
    - If you need assistance, contact the instructor, TA/IA, or visit the CCI Tutoring Center.
    - Cheating will result in severe academic consequences, including potential grade reduction.

Program: String Concatenation (100 points)

Objective:

Implement a C program that concatenates two strings with a space in between using dynamic memory allocation and pointer manipulation, without utilizing the `strcpy` or `strcat` functions.

Requirements:

1. **Function Implementation:**

   - Implement the `concatenate()` function with the following prototype:

     ```
     char * concatenate(const char * string1, const char * string2);
     ```

   - **Dynamic Memory Allocation:**
     - Allocate memory dynamically to create a new string that can hold both input strings, an additional space character, and the null terminator (`\0`).
     - Do **not** use arrays for storing the concatenated string.
   - **String Concatenation:**
     - Concatenate `string1` and `string2` with a single space character separating them.
     - **Do not** use `strcpy()`, `strcat()`, or any other standard library string functions for concatenation. Implement the logic manually using pointers.

2. **Main Function:**

   - Use the provided `concat_test.c` as a starting point.
   - The program should handle strings of **any length** without assuming a specific maximum length.

3. **Memory Management:**

   - Ensure that all dynamically allocated memory is properly managed to prevent memory leaks. Use `free()` appropriately.

Sample Runs:

Sample Run 1:

```
First string: The United States
Second string: of America
The two strings concatenated: The United States of America
```

Sample Run 2:

```
First string: The University of North Carolina
Second string: at Charlotte
The two strings concatenated: The University of North Carolina at Charlot
```

Provided Code (`concat_test.c`):

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

char * concatenate(const char * string1, const char * string2);

int main(int argc, const char * argv[]) {
    char str1[] = "The United States";
    char str2[] = "of America";
    printf("First string: %s\n", str1);
    printf("Second string: %s\n", str2);
    char * str3 = concatenate(str1, str2);
    if (str3 != NULL) {
        printf("\nThe two strings concatenated: %s\n", str3);
        free(str3); // Free allocated memory
    } else {
        printf("\nError concatenating strings.\n");
    }

    char str4[] = "The University of North Carolina";
    char str5[] = "at Charlotte";
    printf("\nFirst string: %s\n", str4);
    printf("Second string: %s\n", str5);
    char * str6 = concatenate(str4, str5);
    if (str6 != NULL) {
        printf("\nThe two strings concatenated: %s\n", str6);
        free(str6); // Free allocated memory
    } else {
        printf("\nError concatenating strings.\n");
    }

    return 0;
}
```

Submission Instructions

1. **Prepare Your File:**

- Ensure your implemented `concatenate()` function is correctly added to `concat_test.c`.
- Save your final program as `concat_test.c`.

2. **Submit via Canvas:**
   - Upload the `concat_test.c` source code file to Canvas within the designated assignment submission area.
3. **Organize Your Work:**
   - Keep your lab files organized and ensure only the necessary `.c` file is submitted.

Grading Rubric (100 points total)

Your submission will be evaluated based on the following criteria:

Criteria	Full Credit	Partial Credit	No Credit	Points
Logic and Flow (60 points)	- Compiles without errors or warnings. - Correct implementation of `concatenate()` using dynamic memory allocation. - Proper pointer manipulation without using `strcpy` or `strcat`. - Handles any string lengths correctly.	- Minor logical issues but overall functionality is correct. - May have slight errors in memory allocation or pointer usage. - Partial handling of string lengths.	- Does not compile or has significant logical flaws. - Incorrect implementation of `concatenate()`. - Fails to handle required functionalities.	/60
Output (30 points)	- Output matches the formatting and layout of the	- Output is mostly correct with	- Output has significant differences from	/30

Criteria	Full Credit	Partial Credit	No Credit	Points
	provided sample runs precisely.	minor discrepancies in formatting or content.	the expected results. - Does not produce the correct concatenated string.	
Formatting/Organization (10 points)	- Code is well-organized with clear indentation and appropriate use of whitespace. - Variables and functions are appropriately named. - Comments effectively explain non-trivial parts of the code.	- Code is mostly well-formatted with minor indentation or naming issues. - Some comments are present but may be insufficient.	- Code is disorganized, poorly indented, and lacks clarity. - Minimal or no comments provided.	/10

Additional Deductions:

- **Warnings During Compilation:** Each warning issued by the compiler will result in a **-10%** deduction.
- **Incorrectly Named Files:** Each incorrectly named file will result in a **-2 points** deduction.
- **Academic Dishonesty:** Any form of cheating will result in a reduction of your course grade by one letter grade, regardless of the assignment's point value.

Important Notes

- **Memory Management:** Always free dynamically allocated memory to prevent memory leaks.
- **No Use of Standard String Functions:** The `concatenate()` function must manually handle string concatenation without using `strcpy()`, `strcat()`, or similar functions.

- **Function Prototype Compliance:** Ensure that the `concatenate()` function strictly adheres to the provided prototype.

Good luck with your lab! If you encounter any issues, please reach out to your TA/IA, or