Assignment 3

Adv C Programming Comprehensive Assignment

Title: Mastering Advanced C Programing Concepts

Objective: The objective of this assignment is to reinforce your understanding of advanced C programming and assembly language concepts. You will work on a series of tasks that cover topics such as bit manipulation, function call, stack, union, assembler, and memory allocations.

Assignment Tasks:

Task 1: Bit manipulation

Bit manipulation is a technique used in programming to manipulate individual bits within a binary representation of data. It's often used for tasks such as setting, clearing, or toggling specific bits to achieve certain goals. Implement the missing bit manipulation lines in the below example in C.

Compile the code with your implementation and execute it. You should capture a screenshot of the execution result and generate assembly code (.s) from the C code.

```
#include <stdio.h>

int main() {
    // Initialize a byte with binary value 11101100 (in hexadecimal, 0xec)
    (Implement your code)

    // Display the initial value of 'data' in binary
    printf("Initial data: 0x%02X\n", data);

    // Set the 3rd bit (counting from 0)
    (Implement your code)
    printf("After setting 3rd bit: 0x%02X\n", data);

    // Clear the 5th bit
    (Implement your code)
    printf("After clearing 5th bit: 0x%02X\n", data);

    // Toggle the 7th bit
    (Implement your code)
    printf("After toggling 7th bit: 0x%02X\n", data);

    // Check and print a message if the 7th bit is set
    (Implement your code)

    return 0;
}
```

Task 2: Pointers

In C, arrays and pointers are closely related, and you can use pointers to work with arrays efficiently. Implement the missing matrix multiplication lines using double pointers for 2D array matrices in the below C function.
Please fix, add, or modify any lines if needed.
Compile the code with your implementation and execute it. You should capture a screenshot of the execution result and generate assembly code (.s) from the C code.

```
#include <stdio.h>

#define ROWS_A 9
```

```
#define COLS_A 8
#define ROWS_B 8
#define COLS_B 9

void matrix_multiply(int **A, int **B, int **C) {
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            (Implement your code)// Initialize the result matrix element to 0
            for (int k = 0; k < COLS_A; k++) {
                (Implement your code) // Perform matrix multiplication
            }
        }
    }
}

int main() {
    int matA[ROWS_A][COLS_A] = {
        {1, 2, 3, 4, 5, 6, 7, 8},
        {2, 3, 4, 5, 6, 7, 8, 9},
        {3, 4, 5, 6, 7, 8, 9, 1},
        {4, 5, 6, 7, 8, 9, 1, 2},
        {5, 6, 7, 8, 9, 1, 2, 3},
        {6, 7, 8, 9, 1, 2, 3, 4},
        {7, 8, 9, 1, 2, 3, 4, 5},
        {8, 9, 1, 2, 3, 4, 5, 6},
        {9, 1, 2, 3, 4, 5, 6, 7},
    };

    int matB[ROWS_B][COLS_B] = {
        {1, 2, 3, 4, 5, 6, 7, 8, 9},
        {2, 3, 4, 5, 6, 7, 8, 9, 1},
        {3, 4, 5, 6, 7, 8, 9, 1, 2},
        {4, 5, 6, 7, 8, 9, 1, 2, 3},
        {5, 6, 7, 8, 9, 1, 2, 3, 4},
        {6, 7, 8, 9, 1, 2, 3, 4, 5},
        {7, 8, 9, 1, 2, 3, 4, 5, 6},
        {8, 9, 1, 2, 3, 4, 5, 6, 7},
    };

    int **matOut;

    // Multiply matrices matA and matB
    (Implement your code)

    // Print the result of the matrix multiplication
    (Implement your code)
```

```
    return 0;
}
```

Task 3: Memory allocation

Implement the above matrix multiplication with dynamic memory allocation for the matrices. Increase the matrix dimension to 50x70 (row=50, column =70), load the matrices, matA and matB, from a text file, and print the multiplication result, matOut, as a text file. You can use fopen, fscanf, and fprintf to read integer values from the input file and write the result to the output file.
Please fix, add, or modify any lines if needed.
Compile the code with your implementation and execute it. You should capture a screenshot of the execution result and generate assembly code (.s) from the C code.

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

#define ROWS_A 50
#define COLS_A 70
#define ROWS_B 70
#define COLS_B 50

void matrix_multiply(int **A, int **B, int **C) {
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            (Implement your code)// Initialize the result matrix element to 0
            for (int k = 0; k < COLS_A; k++) {
                (Implement your code) // Perform matrix multiplication
            }
        }
    }
}

int main() {
    File *inFile;
    File *outFile;
    int **matA;
    int **matB;
    int **matOut;

    // Allocate memory for matA, matB, and matOut
    (Implement your code)
```

```
    // Open the file for reading and load integers to matA and matB
    (Implement your code)

    // Multiply matrices matA and matB
    (Implement your code)

    // Write the result of the matrix multiplication
    (Implement your code)

    // Free the dynamically allocated memories
    (Implement your code)

    // Close the file
    fclose(inFile);
    fclose(outFile);

    return 0;
}
```

Task 4: Struct, union, and function call by reference

Fix or add any missing lines in the source code to exercise struct, union, and function call by reference.
Compile the code with your implementation and execute it. You should capture a screenshot of the execution result and generate assembly code (.s) from the C code.

```
#include <stdio.h>

// Define a union named "Data" to hold different data types
union Data {
    int intData;
    double doubleData;
    char *stringData;
};

// Define a structure named "SNN" to represent a spiking neuron network
struct SNN {
    char *neuronName;
    int neuronNumber;
    union Data commonData;
};

// Use typedef to create aliases for the structure and union
```

```
typedef struct SNN SNN;
typedef union Data Data;

// Function to modify a neuron name of SNN by reference
void modifyName(SNN *snn, char *newName) {
    snn->neuronName = newName;
}

void modifyNumber(SNN *snn, int newNumber) {
    snn->neuronNumber = newNumber;
}

// Function to modify the common data of a SNN by reference
void modifyData(SNN *snn, Data *data) {
    snn->commonData.intData = data->intData;
    snn->commonData.doubleData = data->doubleData;
    snn->commonData.stringData = data->stringData;
}

int main() {
    // Create an instance of the SNN structure
    SNN snnA;
    strcpy(snnA.neuronName, “LIF Neuron");
    neuronA.neuronNum = 100;

    // Create an instance of the Data union
    Data dataA;
    dataA.intData = 150;
    dataA.doubleData = 12.7;
    dataA.stringData = “Number of synapses and average weight”

    // Call the modifyName function by reference to change the name of snnA
    modifyName(&snnA, “Hodgkin-Huxley”);

    // Call the modifyAge function by reference to change the number of snnA
    modifyNumber(&snnA, 20);

    // Call the modifyData function by reference to change the common data of
snnA
    modifyData(&snnA, &dataA);

    // Display the modified snnA information
    (Implement your code)

    return 0;
}
```

Task 5: Integration and Documentation

1. Combine all the tasks into a single, well-organized C program.
2. Document your code thoroughly with comments explaining each section.
3. Prepare a report summarizing everything such as each step or functions that you implemented, challenges faced, and what you've learned during this assignment.

Submission Guidelines:

- Submit the consolidated source code (a zip file) for the entire assignment.
- Include a README file with instructions for compiling and running your program.
- Ensure your code is well-documented with comments.
- Submit your report in a separate PDF document.

Assessment Criteria:

Your assignment will be assessed based on the following criteria:

- Submit your own work (80% in points) even though it doesn’t execute as expected.
- Proper use of C data types, pointers, arrays, structures, and dynamic memory allocation.
- Efficiency and readability of the code.
- Effective use of memory, functions, and function pointers.
- Proper error handling and validation.
- Utilization of macros for code optimization.

- Utilization of C standard library functions.
- Quality and completeness of documentation.

Important Note: **Plagiarism will not be tolerated, and students are expected to produce their work independently.** Please perform this assignment by yourself and don't copy any solutions from any Generative AI applications like ChatGPT, your friends or online. If I find any things that imply plagiarism, you will lose the whole points and be reported.