Lab Assessment 2 (10%)

This is an open book, open internet individual in-class coursework (No AI Bots and No Discussion).

There are 3 questions in this sheet.

Duration: 2 hours (120 minutes)

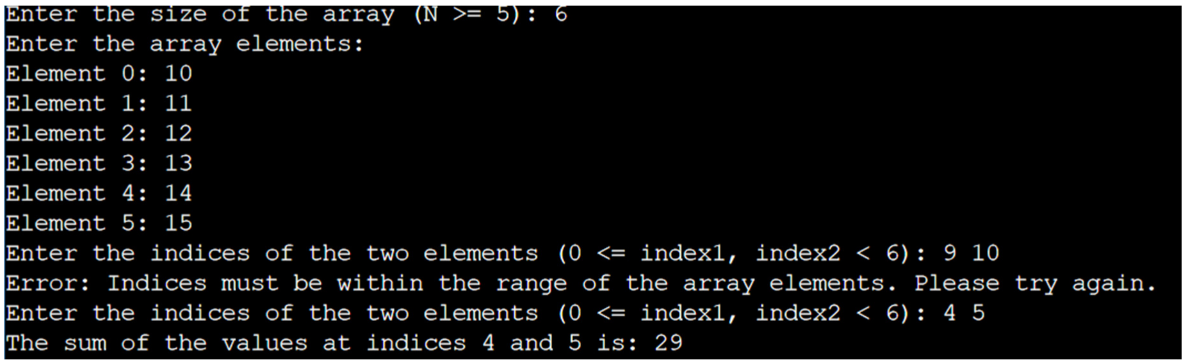
Question 1 (3 marks)

Write a C program that takes in an array of integers and two target indices of the two elements in the array from user. Compute and print the sum of the two target elements. The program should use **pointers** to efficiently access and manipulate the array elements.

The program should have the following functionality:

1) Prompt the user to enter the size N of the array. Validate if N ≥ 5.
2) Dynamically allocate memory for the array using pointers.
3) Prompt the user to enter each element of the array one by one.
4) Prompt the user to enter the two indices (positions) of the array elements of their choice. Validate if the indices are within the range of the array elements (0 <= index1, index2 < N). Prompt the user for input again until the indices entered are valid.
5) Use a pointer-based approach to find the sum of the values of the two array elements.
6) Print the output for the sum.

A sample program output is as shown as follows (make sure the output logic is as close as possible to the sample):



```
Enter the size of the array (N >= 5): 6
Enter the array elements:
Element 0: 10
Element 1: 11
Element 2: 12
Element 3: 13
Element 4: 14
Element 5: 15
Enter the indices of the two elements (0 <= index1, index2 < 6): 9 10
Error: Indices must be within the range of the array elements. Please try again.
Enter the indices of the two elements (0 <= index1, index2 < 6): 4 5
The sum of the values at indices 4 and 5 is: 29
```

Note: For this question, writing modular codes with function(s) to perform the task is allowed, but not necessary. Write comments to explain your code as necessary.

Marking Scheme:

- Student will not be able to get more than half of the marks (1.5/3) if the program compile with syntax error.
- Correctly get and validate user inputs: 1 mark
- Correctly access array element: 1 mark
- Correctly calculate and print sum: 1 mark

Sample Answer:

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

int main() {
    int N, i, index1, index2;
    int* arr;

    // 1. Prompt the user to enter the size N of the array
    printf("Enter the size of the array (N >= 5): ");
    scanf("%d", &N);

    // Validate if N >= 5
    while (N < 5) {
        printf("Error: N must be greater than or equal to 5.
Please enter again: ");
        scanf("%d", &N);
    }

    // 2. Dynamically allocate memory for the array using pointers
    arr = (int*)malloc(N * sizeof(int));
    if (arr == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    // 3. Prompt the user to enter each element of the array one
by one
    printf("Enter the array elements:\n");
    for (i = 0; i < N; i++) {
        printf("Element %d: ", i);
        scanf("%d", arr + i);
    }

    // 4. Prompt the user to enter the two indices (positions) of
the array elements of their choice
    do {
        printf("Enter the indices of the two elements (0 <=
index1, index2 < %d): ", N);
        scanf("%d %d", &index1, &index2);

        // Validate indices
        if (index1 < 0 || index1 >= N || index2 < 0 || index2 >=
N) {
            printf("Error: Indices must be within the range of the
array elements. Please try again.\n");
        }
    } while (index1 < 0 || index1 >= N || index2 < 0 || index2 >=
N);

    // 5. Use a pointer-based approach to find the sum of the
values of the two array elements
    int sum = *(arr + index1) + *(arr + index2);

    // 6. Output the sum
    printf("The sum of the values at indices %d and %d is: %d\n",
index1, index2, sum);

    // Free the dynamically allocated memory
    free(arr);

    return 0;
}
```

Question 2 (3 marks)

Write a C program that prompts the user to enter three student names and marks one by one. After the data is entered, the program will display the names and marks of all the students, as well as calculate and display the average mark. Use **array** as the data structure to store student names and marks. The use of a struct is allowed but not necessary.

Write C program to do the following:

1) Prompt the user to enter the values of the three student names and marks one by one. (Hint: use "`scanf(" %[^\n]", names[i]);`" instead of `fgets` to get student names if `fgets` causes problem in the program flow).
2) After all data is entered, calculate the average mark, and
3) Print the student names and marks of the students, and the average mark.

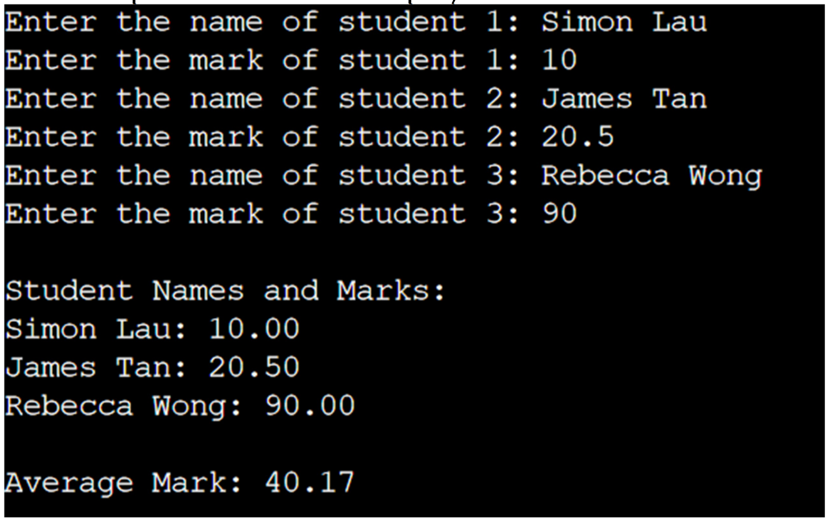
Perform the following tasks with functions:

1) Calculate the average mark.
2) Display student names and marks and the average mark.

Use the following function prototypes as guide for you to create the functions:

```
float calculateAverage(const float marks[], int count);
void displayStudentData(const char names[][NAME_LENGTH], const
float marks[]);  // NAME_LENGTH 50
```

A sample program output is as shown as follows (make sure the output logic is as close as possible to the sample):



```
Enter the name of student 1: Simon Lau
Enter the mark of student 1: 10
Enter the name of student 2: James Tan
Enter the mark of student 2: 20.5
Enter the name of student 3: Rebecca Wong
Enter the mark of student 3: 90

Student Names and Marks:
Simon Lau: 10.00
James Tan: 20.50
Rebecca Wong: 90.00

Average Mark: 40.17
```

Note: Write comments to explain your code as necessary.

Marking scheme:

- Student will not be able to get more than half of the marks (1.5/3) if the program compile with syntax error.
- Correct implementation of `displayStudentData` function: 1 mark
- Correct implementation of `calculateAverage` function: 1 mark
- Correct implementation for main program logic: 1 mark

Sample Answer:

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

#define NAME_LENGTH 50
#define NUM_STUDENTS 3

// Function to calculate the average mark
float calculateAverage(const float marks[], int count) {
    float sum = 0.0f;
    for (int i = 0; i < count; i++) {
        sum += marks[i];
    }
    return sum / count;
}

// Function to display student names, marks, and average mark
void displayStudentData(const char names[][NAME_LENGTH], const
float marks[]) {
    // float avgMark = calculateAverage(marks, count);

    printf("\nStudent Names and Marks:\n");
    for (int i = 0; i < NUM_STUDENTS; i++) {
        printf("%s: %.2f\n", names[i], marks[i]);
    }

    // printf("\nAverage Mark: %.2f\n", avgMark);
}

int main() {
    char studentNames[NUM_STUDENTS][NAME_LENGTH];
    float studentMarks[NUM_STUDENTS];

    // 1. Prompt the user to enter the student names and marks
    for (int i = 0; i < NUM_STUDENTS; i++) {
        printf("Enter the name of student %d: ", i + 1);
        scanf(" %[^\n]", studentNames[i]);
        printf("Enter the mark of student %d: ", i + 1);
        scanf("%f", &studentMarks[i]);
    }
    // 2. Print the student names and marks, and the average mark
    displayStudentData(studentNames, studentMarks);

    // 3. Calculate the average mark
    float averageMark = calculateAverage(studentMarks,
NUM_STUDENTS);
    printf("\nAverage Mark: %.2f\n", averageMark);


    return 0;
}
```

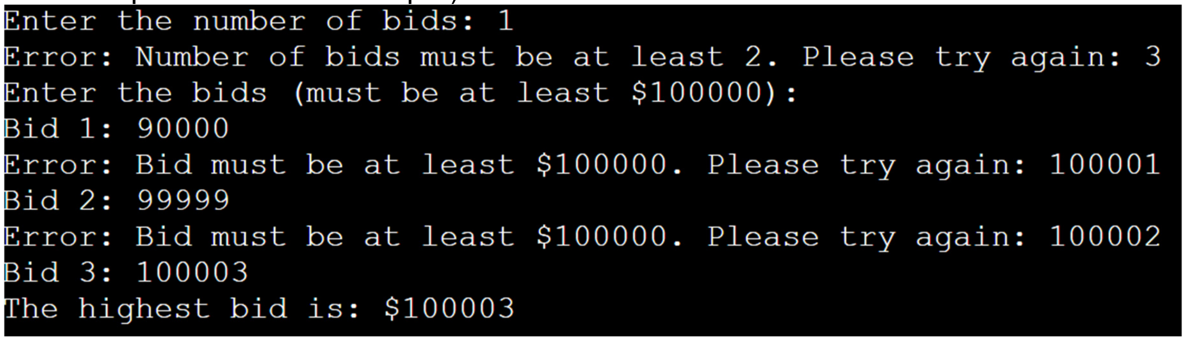
Question 3 (4 marks)

In an antique auction bidding system, multiple bidders can submit their bids for an antique item. The auction will always be awarded to the *highest bidder*. However, to ensure the bids are serious, each bid must be at least $100,000. Use the appropriate **array** data structure to manage the bids.

Write C program to do the following:

1) The user should be able to enter the number of bids (N). Perform validation that N ≥ 2. If there are fewer than two valid bids, the program should display an appropriate message indicating that a highest bid cannot be determined.
2) The user should then enter the values of the N bid one by one. Perform validation of user inputs to ensure each bid is at least $100,000.
3) The program will determine the highest bid from the entered values.
4) The program should output the value of the highest bid.

A sample program output is shown as follows (make sure the output logic is as close as possible to the sample):



```
Enter the number of bids: 1
Error: Number of bids must be at least 2. Please try again: 3
Enter the bids (must be at least $100000):
Bid 1: 90000
Error: Bid must be at least $100000. Please try again: 100001
Bid 2: 99999
Error: Bid must be at least $100000. Please try again: 100002
Bid 3: 100003
The highest bid is: $100003
```

Note: For this question, writing modular codes with function(s) to perform the task is allowed, but not necessary. Write comments to explain your code as necessary.

Marking scheme:

- Student will not be able to get more than half of the marks (2/4) if the program compile with syntax error.
- Correct implementation to get and validate user inputs: 2 marks
- Correct implementation to calculate and output the highest bid: 2 marks

Sample Answer 1:

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

#define MIN_BID 100000
```

```
int main() {
    int numBids, i;
    double* bids;
    double highestBid;

    // 1. Prompt the user to enter the number of bids (N)
    printf("Enter the number of bids: ");
    scanf("%d", &numBids);

    // Validate that N >= 2
    while (numBids < 2) {
        printf("Error: Number of bids must be at least 2. Please 
try again: ");
        scanf("%d", &numBids);
    }

    // 2. Dynamically allocate memory for the bid array
    bids = (double*)malloc(numBids * sizeof(double));
    if (bids == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    // Prompt the user to enter the bids one by one
    printf("Enter the bids (must be at least $%d):\n", MIN_BID);
    for (i = 0; i < numBids; i++) {
        printf("Bid %d: ", i + 1);
        scanf("%lf", &bids[i]);

        // Validate that each bid is at least $100,000
        while (bids[i] < MIN_BID) {
            printf("Error: Bid must be at least $%d. Please try 
again: ", MIN_BID);
            scanf("%lf", &bids[i]);
        }
    }

    // 3. Determine the highest bid
    highestBid = bids[0];
    for (i = 1; i < numBids; i++) {
        if (bids[i] > highestBid) {
            highestBid = bids[i];
        }
    }

    // 4. Output the highest bid
    printf("The highest bid is: $%.0f\n", highestBid);
    // Free the dynamically allocated memory
    free(bids);

    return 0;
}
```

Sample Answer 2:

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

#define MAX_BIDS 100
#define MIN_BID 100000

int main() {
    int numBids, i;
    double bids[MAX_BIDS];
    double highestBid;

    // 1. Prompt the user to enter the number of bids (N)
    printf("Enter the number of bids (max %d): ", MAX_BIDS);
    scanf("%d", &numBids);

    // Validate that N is between 2 and MAX_BIDS
    while (numBids < 2 || numBids > MAX_BIDS) {
        printf("Error: Number of bids must be between 2 and %d.
Please try again: ", MAX_BIDS);
        scanf("%d", &numBids);
    }

    // Prompt the user to enter the bids one by one
    printf("Enter the bids (must be at least $%d):\n", MIN_BID);
    for (i = 0; i < numBids; i++) {
        printf("Bid %d: ", i + 1);
        scanf("%lf", &bids[i]);

        // Validate that each bid is at least $100,000
        while (bids[i] < MIN_BID) {
            printf("Error: Bid must be at least $%d. Please try
again: ", MIN_BID);
            scanf("%lf", &bids[i]);
        }
    }

    // 3. Determine the highest bid
    highestBid = bids[0];
    for (i = 1; i < numBids; i++) {
        if (bids[i] > highestBid) {
            highestBid = bids[i];
        }
    }

    // 4. Output the highest bid
    printf("The highest bid is: $%.0f\n", highestBid);

    return 0;
}
```

1.0 Instructions

1. For each question you have to provide programming solutions as separate source code files (for example: Q1.c, Q2.c and Q3.c).
2. **Submission on Moodle:** Submit you code separately one-by-one separately (for example: Q1.c, Q2.c and Q3.c) to Moodle. Make sure you have uploaded all the files successfully before you click the “Submit” button.

2.0 Evaluation Criteria

1. The evaluation is based on the following criteria:
   - Successful execution of the program – program runnable with correct inputs and outputs.
   - Correctness of functionalities as stated in each question.
   - Code quality (organization of codes)

2. Compatibility to standard C11 or C17. (If your program does not compile in such an environment, marks may be deducted.)

3.0 Plagiarism and Integrity

1. Codes copied and pasted directly and exactly from AI chatbots (e.g. Chatgpt, Claude, Copilot, Poe, Gemini etc.), and/or friends or other acquaintances without problem solving and coding effort will be considered as plagiarism and will not get any mark once proven.

2. You should have written every line of code yourself and should be able to explain each line fully when asked to do so (by the lab examiner).

3. Do not share your code with any other students.

End of Question