Quiz 2 - Wednesday

July 17, 2024

Instructions

- You have time to finish the quiz before the lab session is over. **No need to submit anything for these quizzes**, but you **must** show your codes and results to your TAs before you leave the class. TAs may ask you questions and request you to work with your codes, e.g. asking you to run and compile, add/remove, and debug your program. TAs will provide you with feedback and your grade right away, but the grades will be released on Avenue a few days later.
- During the quizzes using AI generative models, like ChatGPT, is **NOT** allowed, as I believe quizzes are easy enough to be handled. But you can search on the internet, take a look at the lecture notes, talk to your friends, or even having your TA's help.
- The quiz starts at the beginning of the lab sessions, and only those who are present in the class in person can take the quiz. Please don't be late and make sure you are in class 15 min before the quiz starts.

Programming Questions

No boiler-plate files are available, you can copy the code into `q<question_number>.c` file and then do as directed in the question.

1. **Understanding Implicit and Explicit Type Conversion in C (0.5 points)**

   Consider the following C program snippet:

```c
#include <stdio.h>

int main() {
    int a = 5;
    double b = 12.7;
    char c = 'A';
    double result1;
    int result2;

    // Perform calculations
```

```c
    result1 = a + b / 2;
    result2 = (int) b + c;

    // Print results
    printf("Result1: %f\n", result1);
    printf("Result2: %d\n", result2);

    return 0;
}
```

Questions:

a) **Implicit Conversion:**

- Identify the lines in the program where implicit type conversion occurs.
- Explain what types are being converted and the result of each conversion.

b) **Explicit Conversion:**

- Identify the line in the program where explicit type conversion (casting) occurs.
- Explain the purpose of the cast and what effect it has on the calculation.

c) **Modification:**

- Modify the program to include an additional calculation that demonstrates another example of both implicit and explicit type conversion. Add a new variable `float d = 3.14` and perform a calculation that stores the result in a `double` variable.
- Explain the types involved in the conversions and the results of the calculations.

2. **Mathematical Functions (0.5 points)**

Write a C program that calculates the roots of a quadratic equation $ax^2 + bx + c = 0$ using the quadratic formula. Your program should:

- Prompt the user to enter the coefficients a, b, and c.
- Check if the discriminant $\Delta = b^2 - 4ac$ is non-negative.
- If the discriminant is negative, print a message indicating that there are no real roots.
- If the discriminant is non-negative, compute the two roots using the quadratic formula:

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

- Print the calculated roots.

NOTE: Make use of the functions `sqrt` and `pow` from the `math.h` library where appropriate.

3. **Recursion function** (0.5 points)

   Write a C program that generates the first n numbers in the Fibonacci sequence. The Fibonacci sequence is defined as follows:

   - $F(0) = 0$
   - $F(1) = 1$
   - for $n \geq 2 : F(n) = F(n-1) + F(n-2)$

   Your program should:

   - Prompt the user to enter the value of n (the number of Fibonacci numbers to generate).
   - Validate that n is a positive integer.
   - Use a loop to calculate and print the first n numbers in the Fibonacci sequence.

   NOTE: You must use recursion to calculate the Fibonacci numbers.

4. **Converting from `for` loop to `while` loop** (0.5 points)

   Take a look at the following code. Change the code to use a nested `while` loop instead of `for` loop.

```c
int main()
{
    int row = 5;
    int col = 4;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            printf("(%d, %d) ",i,j);
        }
        printf("\n");
    }
}
```

5. **Understanding Functions and Variable Scope (1 points)**

   Consider the following incomplete C program that demonstrates various concepts related to functions, variable scope, and function declarations:

```c
#include <stdio.h>

// Forward declaration of the function
// (a) Complete the forward declaration to correctly define the
    function signature of ‘isWithinLimit‘
void printArray(int[], int);

int main() {
    // (b) Define an integer array ’numbers’ with 5 elements
       and initialize it with values 1, 2, 3, 4, 5
    // (c) Call the function ’printArray’ passing ’numbers’
       array and 5 as arguments

    const int LIMIT = 100;
    // (d) Write a function ’isWithinLimit’ that accepts an
       integer value and the constant ’LIMIT’
    //     and prints whether the value is within the limit or
       not

    int value = 75;
    // (e) Call the function ’isWithinLimit’ passing ’value’
       and ’LIMIT’ as arguments

    return 0;
}

// (f) Implement the function ’printArray’ to print all
   elements of the array
// (g) Implement the function ’isWithinLimit’ to check if ’
   value’ is within ’LIMIT’ and print appropriate message
```

Complete the program by filling in the blanks marked from (a) to (g). Here are the tasks:

(a) Complete the forward declaration of the function ‘printArray‘ to correctly define its signature.

(b) Define an integer array named ‘numbers‘ with 5 elements and initialize it with values 1, 2, 3, 4, 5.

(c) Call the function ‘printArray‘ from ‘main‘, passing ‘numbers‘ array and 5 as arguments.

(d) Write a function named ‘isWithinLimit‘ that accepts an integer value and the constant ‘LIMIT‘, and prints whether the value is within the limit or not.

(e) Call the ‘isWithinLimit‘ function from ‘main‘, passing ‘value‘ and ‘LIMIT‘ as arguments.

(f) Implement the ‘printArray‘ function to print all elements of the array passed to it.

(g) Implement the ‘isWithinLimit‘ function to check if the ‘value‘ is within the ‘LIMIT‘ and print an appropriate message.

6. **Debugging in C (0.5 points)**

The following C program is intended to calculate and print the factorial of a given number. However, the code contains several bugs. Debug the program using gdb, correct the errors, compile, and run the program to calculate the factorial.

```c
#include <stdio.h>

int factorial(int n) {
    int r;

    for (int i = 1; i < n; ++r) {
        r *= i;
    }

    return r;
}

int main() {
    int num = 5; // Change this value to calculate factorial of
        different numbers
    int fact = factorial(num);

    printf("Factorial of %d is: %d\n", num, fact);

    return 0;
}
```

Explain the steps you took to debug the program, including any changes made to correct the errors. Provide the corrected code, compile and execute it, and include the output showing the factorial calculation for a specific number (e.g., 5).

7. **Operations on arrays (1.5 points)**

Write the following C code:

(a) A function called `void createRandomArray()` that create a random 2D array with n*m dimension and `float` numbers between -50 to +50, while n*m is given by the user.

(b) Use the previous function and create the matrix A (n1*m1) and B (n2*m2).

(c) A function called `void multiply()` to compute the product of $C = A \times B$.