Quiz 2 - Monday

July 15, 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 num1 = 10;
    float num2 = 5.5;
    double result;

    result = num1 + num2; // Line A
    printf("Result (implicit conversion): %lf\n", result);
```

```
    result = (double)num1 + num2; // Line B
    printf("Result (explicit conversion): %lf\n", result);

    return 0;
}
```

Questions:

1. **Implicit Conversion (Line A):**
   - Describe what happens during the expression `result = num1 + num2;`.
   - Explain why the compiler allows this operation without explicit type casting.
2. **Explicit Conversion (Line B):**
   - Explain the purpose of `(double)num1` in the expression `result = (double)num1 + num2;`.
   - What is the difference in behavior between Line A and Line B regarding type conversion?
3. **Discussion:**
   - When is it necessary to use explicit type casting in C? Provide examples where implicit conversion might not be sufficient or appropriate.

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

Consider the following C program snippet that demonstrates the usage of various mathematical functions from the math library:

```
#include <stdio.h>

int main() {
    double angle = 45.0; // Angle in degrees
    double radians = angle * M_PI / 180.0; // Convert degrees
       to radians
    double result;

    // Use math library functions
    result = cos(radians); // Cosine function
    printf("Cosine of %.1f degrees: %.2f\n", angle, result);

    result = exp(2.0); // Exponential function (e^x)
    printf("Exponential function e^%.1f: %.2f\n", 2.0, result);
```

```
    result = log(10.0); // Natural logarithm function (log base
        e)
    printf("Natural logarithm of %.1f: %.2f\n", 10.0, result);

    result = fabs(-5.5); // Absolute value function
    printf("Absolute value of %.1f: %.2f\n", -5.5, result);

    return 0;
}
```

Questions:

- Explain why including the `math.h` header file is necessary for using these mathematical functions in C programs.
- Highlight any specific compiler option(s) that may be needed to successfully compile and link programs using math library functions.

3. **Control Structures: Converting from `if-else` to `switch`** (0.5 points)

   Consider the following C program that uses `if-else` statements to determine the action based on user input:

```
#include <stdio.h>

int main() {
    int choice;
    printf("Enter a number between 1 and 3: ");
    scanf("%d", &choice);

    // Original if-else version
    if (choice == 1) {
        printf("You entered 1.\n");
    } else if (choice == 2) {
        printf("You entered 2.\n");
    } else if (choice == 3) {
        printf("You entered 3.\n");
    } else {
        printf("Invalid choice.\n");
    }
```

```
    // Re-write the above if-else section to use switch
       statements

    return 0;
}
```

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.

```
            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:

```
#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 a C code with following function functions:

   (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 the user

   (b) A function called `void findMax()` to find and print the index of maximum value within the array and the corresponding value

   (c) A function called `void findMin()` to find and print the index of minimum value within the array and the corresponding value

   (d) A function called `float findAverage()` to calculate and print the average of the values in the array. This function return a `float` number which is the average value.

   (e) A function called `void transposeMatrix()` to find and print the transpose of a matrix