Assignment 1

CS 1037
Computer Science Fundamentals II

Due Date: 11:59 PM, Sep 23, 2024

Learning Outcomes

In this assignment, you will get practice with:

- Writing basic C functions.
- Compiling and testing C programs.
- Working with function prototypes and implementation.
- Conducting basic testing of programs.

Introduction

In this assignment, you will work on a series of C functions that cover basic computations and operations, such as converting units, calculating geometric properties, and summing sequences. Each function will be implemented in C, with corresponding test cases provided by you to ensure correctness.

You are required to implement, test, and submit these functions as part of an Eclipse/CDT project. It is crucial that you thoroughly test your code to ensure that all functions perform correctly under a variety of conditions.

Assignment Instructions

Your assignment submission must follow the guidelines:

- Create an Eclipse/CDT project
- Name your Eclipse project using your Western network login, followed by _a01. For example, if your login is *pop1234*, the project should be named *pop1234_a01*.
- Place the given files (*a01_start.zip*) after decompressing them into the project 'src' folder, then complete and test these functions specified below.
- All programs and tests must be part of a single Eclipse project.
- Export the project as a .zip file. The exported file must be named *login_a01.zip* (replace login with your actual login).
- Remember to include a *testing.txt* file detailing your test results in your project.
- Any explanations or discussions should be done through the assignment 1 discussion forum on the course website on OWL Brightspace.

Failure to adhere to these requirements may result in significant penalties or a grade of zero, as this assignment may be auto-graded.

Function Specifications

You are required to implement the following functions:

1. **feet_to_acres**
   Converts square footage to acres.

2. **mow_lawn**
   Calculates how long it takes to mow a rectangular lawn.
3. **date_convert**
   Converts a date from `DDMMYYYY` format to `YYYYMMDD` format.
   - **Note**: This function cannot use strings—only math operations.
4. **falling_time**
   Calculates the time it takes for an object to fall a certain distance due to gravity.
5. **hypotenuse**
   Calculates the hypotenuse of a right triangle given the two other sides.
   - **Note**: Do not use `hypot`, `hypotf`, or `hypotl` from the C math library.
6. **sum_even**
   Sums and returns all even numbers from 2 to `n` (inclusive).
7. **sum_partial_harmonic**
   Sums and returns the total of a partial harmonic series: $H_n = 1 + \frac{1}{2} + \frac{1}{3} + .. + \frac{1}{n}$.

Provided files

The following files are provided in the archive a01_start.zip:

- functions.h: Contains documentation and prototypes for the required C functions.
- functions.c: Contains your implementation of the functions described in functions.h.
- main.c: A test file that will allow you to test your implemented functions.
- testing.txt: A text file where you will place the output from your program's tests.

Implementation and Testing

Do not modify the provided functions.h or main.c files. Modify and implement the logic within functions.c to meet the specifications in this assignment. Ensure that the output of your program after execution is stored in the testing.txt.

Compilation Instructions

After implementing your functions in functions.c, you need to compile your program. Use the following command in your terminal or command prompt to compile your code:

```
gcc -o a1 main.c functions.c
```

Here's what each part of this command means:

- `gcc`: This is the compiler we're using. If you're using GCC, you can replace this with gcc.
- `-o a1`: This specifies the output file name. Your executable will be named assignment1.
- `main.c functions.c`: These are the source files to compile.
- `-lm`: This links the math library, which is necessary for mathematical functions like sqrt().

After compilation, you can run your program with: ./a1 (Mac OS)

Note: If you're using Windows and MinGW, you might need to use assignment1.exe to run the program.

Important: When compiling, make sure all your source files (**main.c, functions.c, functions.h**) are in the same directory. If you encounter any compilation errors, double-check your function implementations and ensure that all functions are properly declared in **functions.h** and defined in **functions.c**.

Remember to compile and test your program frequently as you implement each function. This will help you catch and fix errors early in the development process.

Submission Instructions

1. Complete the implementation of the functions in **functions.c**.
2. Ensure that your project follows the submission requirements, including naming conventions and file structure.
3. Test your project thoroughly and include the results in **testing.txt**.
4. Export your project as a .zip file (named login_a01.zip) and submit it through OWL Brightspace.

Grading

- Functionality (15 marks): Correct implementation of each function. Proper handling of input and output. Proper use of control structures, loops, and function prototypes.
- Testing (3 marks): Thorough and clear testing of all functions.
- Code Style and Readability (2 marks): Meaningful variable names. Appropriate use of comments to explain complex logic. Proper formatting and indentation.

Remember **you must do** all the work on your own. **Do not copy** or even look at the work of another student. All submitted code will be run through similarity-detection software.

Good luck and remember to thoroughly test your code before submission!

Appendix A

Unit Conversion

- **Feet to Acres Conversion**:
    - 1 acre = 43,560 square feet. You'll need to divide the input (in square feet) by this constant to get the result in acres.
- **Mowing Time Calculation**:
    - The area of a rectangular lawn can be calculated by multiplying its width by its length. To find the time required to mow it, divide the area by the mowing speed (area covered per minute).

Working with Dates and Numbers

- **Integer Division & Modulus for Date Conversion**:
    - Date conversion (from DDMMYYYY to YYYYMMDD) requires breaking down an integer into its day, month, and year components using division and modulus.
    - For example, to extract DD from 25101962, you would perform:

```
int day = date / 1000000; // Extract day (25)
int month = (date / 10000) % 100; // Extract month (10)
int year = date % 10000; // Extract year (1962)
```

    - This logic can be applied to rearrange the date components as required.

Geometry and the Pythagorean Theorem

- **Pythagorean Theorem**: The hypotenuse of a right triangle can be calculated using the formula: $c = \sqrt{a^2 + b^2}$
- In C, the square root can be computed using the sqrt function from the <math.h> library.

Physics: Free Fall

- **Falling Time Calculation**: The time t it takes for an object to fall a certain distance d due to gravity can be calculated using the formula:

$$t = \sqrt{\frac{2d}{g}}$$

Where g is the acceleration due to gravity (approximately 9.8 m/x^2)

Summation Algorithms

- **Summing Even Numbers**: To sum even numbers from 2 to n, use a loop that checks whether a number is divisible by 2.
- **Harmonic Series**: A partial harmonic series is a sum of reciprocals from 1 to n:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + .. + \frac{1}{n}$$