Beginner C problems for application

Achraf KHABAR

August 7, 2023

1 Problem 1 - *Prime Number generator*

Create a program that generates a list of prime numbers within a given range. Using a simple program with ***main()*** or *function* with the name of ***is-prime()***.

1. Using only the ***stdio.h***.
2. The user should enter the **start** and the **end** .
3. Create a function named ***void swap(int *start, int *end)*** which swap the start and the end if the start is bigger than the end.

2 Problem 2 - *Factorial Calculator*

Create a program that calculates the factorial of a given non-negative integer.

1. Using only the ***stdio.h***.
2. Create a function named ***int factorial-it(int num)*** to create a function calculating factorial in iterative way.
3. Create other function named ***int factorial-rec(int num*** to create a function calculating factorial in recursive way.

3 Problem 3 - *Table of Powers*

Create a program that takes a number and a range as input and prints its powers within that range.

1. Using only the ***stdio.h***.
2. Create a function named ***int power(int base, int exponent)*** in order to calculate the ***power***.

4 Problem 4 - *Square Root Function*

Implement a function to calculate the ***square root*** of a given positive number using the ***Newton-Raphson*** method or any other suitable approach.

$$\sqrt{1+x} = \sum_{n=0}^{\infty} \frac{(-1)^n \cdot (2n)! \cdot x^n}{(1-2n) \cdot (n!)^2 \cdot (4^n)} \tag{1}$$

1. Using only ***stdio.h***.
2. Create a function named ***double square-root(double num, int range)*** in order to calculate the square root.
3. Evaluate the $\sqrt{2} = 1.41421$ and $\sqrt{11} = 3.3166$ for i range with $i \in \{4, 5, 6\}$.

5 Problem 5 - *Logarithmic Equation Solver*

Design a function that solves simple logarithmic equations, such as $log(x) = a$, for a given value of a.

1. Using only ***stdio.h***.
2. Create a help function ***double my-exp(double x, int n)*** in order to calculate the exponential of given x and in range of n .

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \cdots + \frac{x^n}{n!} + o(x^n) \tag{2}$$

3. Create a functiona named ***double solve-log-equation(double a, int iterations)*** in order to solve the ***logarithmic equation*** .

6 Problem 6 - *Grading System*

Write a program that reads a student's marks and prints their corresponding grade according to a grading system.

1. Using only ***stdio.h***.
2. The tree types of grades we have : ***validate or V / Non Validate or NV /Resit Examination or R***, and you are allowed to choose the norms.

7 Problem 7 - *Palindrome Checker*

Create a program that checks whether a given string is a palindrome or not. A palindrome is a string that reads the same forwards and backwards, ignoring spaces, punctuation, and capitalization.

1. Using only ***stdio.h.***
2. Create a function named ***int isPalindrome(const char* str)*** to check if a string is a palindrome.

8 Problem 9 - *Fibonacci Series Printer*

Create a program that prints the ***Fibonacci series*** up to a given number of terms.

1. Using only ***stdio.h.***
2. Create a function named ***void fibonacciSeries(int n)*** to print the Fibonacci series up to the nth term.

9 Challenge Problem - *Word Search Puzzle Solver*

9.1 Problem Statement

Create a program that searches for a given word in a ***2D grid*** of characters ***(word search puzzle)***. The word can be placed horizontally, vertically, or diagonally in any direction.

1. Using only ***stdio.h***.
2. Create a function named ***int searchWord(char puzzle[][MAX-SIZE], int rows, int cols, const char* word)*** that searches for the given word in the puzzle.
3. The function should return the starting coordinates ***(row*** and ***column)*** of the word if found, otherwise return ***(-1, -1)***.
4. Test your function with various puzzles and words.

9.2 Challenge

Solving this problem involves complex string matching and grid traversal. You'll need to check various directions for the word's presence while handling edge cases and boundaries properly. Additionally, implementing a robust algorithm that efficiently handles different orientations can be quite challenging for beginners.

Solving this problem will enhance your skills in grid manipulation, string processing, and logical problem-solving. It's a great opportunity to explore multidimensional arrays and algorithmic thinking.