Assignment: Swift Programming Exercises on Loops, Conditions, and Collections

Objective: Assess your understanding of fundamental programming concepts in Swift, including loops, conditional statements, and collections such as arrays and dictionaries.

Instructions

- Complete the following problems using Swift.
- Include comments in your code to explain your logic.
- Test your code with various inputs to ensure correctness.
- Submit your Swift source files (.swift) with appropriate filenames for each problem.

Problems:

Problem 1: FizzBuzz
Write a Swift program that prints the numbers from **1** to **100**. For multiples of three, print **"Fizz"** instead of the number. For multiples of five, print **"Buzz"**. For numbers that are multiples of both three and five, print **"FizzBuzz"**.
Requirements:

- Use a loop to iterate through the numbers.
- Use conditional statements (if, else if, else) to determine what to print.

Problem 2: Prime Numbers
Create a function isPrime(_ number: Int) -> Bool that determines whether a given number is prime. Then, use this function to print all prime numbers between **1** and **100**.
Requirements:

- Use loops and conditional statements within your function.
- Call the function in your main program to display the prime numbers

Problem 3: Temperature Converter
Write a Swift program that converts temperatures between **Celsius**, **Fahrenheit**, and **Kelvin**.
Program Specifications:

- Prompt the user to input a temperature value.
- Ask the user to specify the unit of the entered temperature (C, F, or K).
- Convert the temperature to the other two units.
- Display the results with appropriate labels.

Requirements:

- Use conditional statements to handle different units.
- Use functions to perform the conversions.

Problem 4: Shopping List Manager
Develop a simple shopping list application.

Program Features:

- Allow the user to **add** items to the shopping list.
- Allow the user to **remove** items from the shopping list.
- Display the current shopping list.
- Provide an option to **exit** the application.

Requirements:

- Use an array to store the shopping list items.
- Use loops and conditional statements to manage user input.
- Display a user-friendly menu for interaction.

Problem 5: Word Frequency Counter

Write a Swift program that counts the frequency of each word in a given sentence.

Program Specifications:

- Prompt the user to enter a sentence.
- Count how many times each word appears.
- Display the words and their corresponding frequencies.

Requirements:

- Use a dictionary to store word-frequency pairs.
- Ignore punctuation and make the word count case-insensitive.
- Use loops to iterate through the words.

Problem 6: Fibonacci Sequence

Implement a function fibonacci(_ n: Int) -> [Int] that returns an array containing the first **n** numbers of the Fibonacci sequence.

Requirements:

- Use loops to calculate the sequence.
- Handle cases where **n** is less than or equal to zero by returning an empty array.

Problem 7: Grade Calculator

Create a program that processes student test scores.

Program Features:

- Accept a list of student names and their corresponding test scores.
- Calculate the **average score**.
- Determine the **highest** and **lowest** scores.
- Display each student's name with their score and indicate if they are **above** or **below** the average.

Requirements:

- Use arrays or dictionaries to store student data.
- Use loops to process the data.
- Use conditional statements to compare scores.

Problem 8: Palindrome Checker
Write a function isPalindrome(_ text: String) -> Bool that checks whether a given string is a palindrome.
Requirements:

- Ignore spaces, punctuation, and make the check case-insensitive.
- Return true if the string is a palindrome, false otherwise.

Problem 9: Simple Calculator
Create a program that functions as a simple calculator.
Program Features:

- Prompt the user to enter two numbers.
- Ask the user to choose an operation: addition (+), subtraction (-), multiplication (*), or division (/).
- Perform the calculation and display the result.
- Allow the user to perform multiple calculations until they choose to exit.

Requirements:

- Use functions to perform each operation.
- Handle division by zero with an appropriate error message.
- Use a loop to continue the program based on user input.

Problem 10: Unique Characters
Write a function hasUniqueCharacters(_ text: String) -> Bool that determines if a string has all unique characters.
Requirements:

- Consider letters case-sensitively (e.g., 'A' and 'a' are different).
- Return true if all characters are unique, false otherwise.

Submission Guidelines

- Ensure your code compiles and runs without errors.
- Include comments explaining your code logic.
- Organize your code for readability (use proper indentation and spacing).
- Submit all .swift files in a git repository

Grading Criteria

- **Correctness**: Solutions meet the problem requirements and produce the correct output.
- **Code Quality**: Code is clean, well-organized, and follows Swift naming conventions.
- **Comments**: Code includes comments that explain the logic and flow.
- **Functionality**: Programs handle user input gracefully and perform necessary error checking.

Additional Resources

- Swift Programming Language Guide
- Swift Standard Library Reference
- Swift Control Flow
- Swift Collections

If you have any questions or need clarification on the assignment, please feel free to reach out before the submission deadline.

Good luck!