SET A

United International University

Department of Computer Science and Engineering

Final Examination **Spring 2024**

Course Code: **CSE 1112** Course Title: **Structured Programming Language Laboratory**

Date: May 19, 2024 Time: 09:00 AM – 10:00 AM (1 hour) Full marks: 25

Name: Student ID:

Write down C programs for the following problems in Code Blocks (or any C compiler you prefer), and present the code to your instructor after the time is up. You can make rough calculations in this paper.

Problem 1 (Marks: 13)

A word is a **Dragon Word** if it starts with '?', ends with '#' and contains only lowercase or uppercase letters in between. A string is a **Dragon String** if the middle word is a Dragon Word. Your task is to check if a given string is a Dragon String.

For this task, you have to implement the following functions: **[Cannot use string.h functions]**

1. **`int countWords(char *str)`**: Takes a string as a parameter, and counts how many words are in it. *You can safely assume that two words are separated by a space only.*
2. **`void getMiddleWord(char str[])`**: Takes the string and finds the starting index of the middle word. You can assume that there will always be odd number of words. You should use the `countWords()` function.
3. **`int isDragon(char str[])`**: Checks if the middle word is a Dragon word, if it is then the function returns 1, otherwise returns 0. You need to use `getMiddleWord()` function to find the starting index of the middle word.

In the main function, you have to take as input a string, and print if the input string is a Dragon String.

Sample input	Sample output	Explanation
`fly me to the moon`	No	The middle word 'to' is not a Dragon Word
`I love ?SpL# and UIU`	Yes	The middle word '?SpL#' is a Dragon Word
`I ?love# SpL and UIU`	No	There is a Dragon Word, but it's not the middle one

Problem 2 (Marks: 12)

You are tasked with creating a program to evaluate the "luck" of new players in a video game. Each player has allocated stat points in vitality, magic, defense, and attack. The program should determine the luck value of a player based on their stat points. First, create a structure to hold player data, including their username and stat points:

```
struct Player {
    char username[50];
    int vitality;
    int magic;
    int defense;
    int attack;
};
```

The luck value calculation involves two scenarios:

1. If the summation of the player's stat points forms a Spectacular number, the luck value is the product of the Spectacular number and 10.
2. If the summation doesn't form a Spectacular number, the luck value is just the sum of the stat points.

A Spectacular number n must meet these criteria:

- n is a prime number.
- If s is the sum of the digits of n, then $1 + 2 + 3 + \cdots + s$ is divisible by the last digit of n.

Example: 167 is a spectacular number because –

- It is a prime number.
- The sum of its digits are $1 + 6 + 7 = 14$, and $1 + 2 + 3 + \cdots + 14 = 105$, which is divisible by 7, the last digit of 167.

In this problem, you have to implement the following functions:

1. **`int isPrime(int num)`**: Checks if a given num is prime or not. Returns 1 if prime, and 0 otherwise.
2. **`int digitSum(int num)`**: Calculates the sum of digits of a given number num. **You must use recursion**.
3. **`int summation(int n)`**: Calculates the summation of numbers from 1 to num.

Sample input (bold ones are inputs from user, regular ones are prompts)	**Sample output**

`Enter player username:` **`Joey231`**
`Enter vitality:` **`47`**
`Enter magic:` **`40`**
`Enter defense:` **`40`**
`Enter attack:` **`40`**
	`167 is a spectacular number`
	`Luck value for player Joey231is: 1670`

`Enter player username:` **`Ty42`**
`Enter vitality:` **`88`**
`Enter magic:` **`89`**
`Enter defense:` **`87`**
`Enter attack:` **`34`**
	`298 is not a spectacular number`
	`Luck value for player Ty42 is: 298`