PROG20799: Data Structures and Algorithms in C

Evaluation: **10 points**, **7.5%** of your final grade.
Due date: See SLATE.
Late submission: 10% per day penalty, up to 3 days.

Part 1 (2.5 points).

Write a C program that prints **N numbers** generated by the **recursive function**:

$F_n = 2n + 1 + F_{n-1} \; mod \; 50$

where $F_0 = F_1 = F_2 = 10$, and "mod" is a modulo operator: 5 mod 3 is 2.

Requirements:

1. Number **N** must be provided by the user. Function **fgets()** must be used.
2. Ignore incorrect input and ask for input again (see demo run).
3. You need to declare and use recursive function **Fn(…)** (see extra requirements).
4. To print the series, you need to use an i-loop and call the function **Fn(i)**.

Demo Run (Part 1 only):

```
How many numbers you want to print (1-200)? 210
How many numbers you want to print (1-200)? 7
Series: 10 10 10 17 26 37 50
```

Use **#define MAX_INPUT 200** to limit maximum value of **N**

Ignore incorrect input.

Use **fgets()** function to get value **N**

Part 2 (2.5 points).

Create a string **str** that consists only of **alpha-numerical characters** (uppercase A-Z, lowercase a-z and digits) created from series of **N** numbers in **Part 1**.

Requirements:

1. Do not use any extra functions here (no need!).
2. Create **str** inside **main()** function.

Hints:

1. You can create string **str** as an array of size **MAX_INPUT**+1 (where +1 due to null-terminating character).

2. You can use loop from **Part 1** to print characters in the series, and add only **alpha-numerical** ones to **str**.

3. In string **str** you can ignore values greater than 126 (!).
   Example: for series 60 24 65 102 30 53 **str** is “Af5” since 65 is a numerical value of char ‘A’. Values 60 and 24 are ignored (they are special characters in ASCII table).

Demo Run (Part 1 and Part 2):

```
How many numbers you want to print (1-200)? 16
Series: 10 10 10 17 26 37 50 … more data will be displayed there…
String: 23FL9
```

Part 3 (2.5 points).

Modify **str** in **Part 2** so only upper-case characters left in it.
I.e. remove all digits and lower-case characters from the string **str** and print it.

Requirements:

1. You must create and use function **`void removeCharacters(char*);`**
2. Do NOT create/generate a new string. Modify directly **str** from **Part 2**.

Hints:

1. Modify one of the exercises we discussed in the class.
   For **N==40**, the string **str** will be FLKUPMLMPUKZL.

Demo Run (Part 1, Part 2 and Part 3):

```
How many numbers to print (1-200)? 40
Series: 10 10 10 17 26 37 50 … …
String: 23FL9K ……
Upper-case String: FLKUPMLMPUKZL
```

Part 4 (2.5 points).

Modify **str** in **Part 3** and remove duplicate characters.
I.e. string such as FSKFAFK should become FSKA.

Requirements:

1. You must create and use function **`void removeDuplicates(char*);`**
2. Do NOT create a new string! Remove duplicates directly from **str** from Part 3.

Hints:

1. Problem is similar to **Part 3**, but you need to use an extra embedded loop.
2. Solutions found online usually create a new string. See requirement N2 above.
3. Check in-Class **Exercise5_9**. For N==40, the string **str** will be FLKUPMZ.

Demo Run (Part 1, Part 2, Part 3, Part 4):

```
How many numbers to print (1-200)? -5
How many numbers to print (1-200)? 220
How many numbers to print (1-200)? 40
Series: 10 10 10 17 26 37 50 …
String: 23FL9K…
Upper-case String: FLKUPMLMPUKZL
Unique String: FLKUPMZ
```

Extra Requirements:

1. **Part 1**: you must declare and use a recursive function **Fn(n)**
   You'd have to use this function to generate the series in **Part 1**.
   Example: Fn(0) should return 10, **Fn(3)** should return 17. The series starts with n=0.

   Function **Fn(n)** must use recursive calls and must return **correct data** type.
   You'll get zero grade for **Part 1** if non-recursive function is used.

   Do not declare any other functions in **Part 1**.
   Series output and string **str** generation should be in the function **main().**
   You must use **fgets()** function and then convert string to **N**.

2. In each Part your output should match the **demo run**!

3. Your solution should have optimal time and space complexity (i.e. use proper data types, minimize number of steps, variables, expressions, and function calls).

Submission Requirements:

This is an individual assignment. Even partially copied code will be subject to regulations against academic integrity. Do NOT discuss or share your solution with anybody. Posting this assignment or solution on the Internet is a violation of the Student Code of Conduct.

- Please make sure your code is POSIX-compliant (works in Cygwin)!
- Your submission must be one **text file only**: **assignment1.txt**
- Please save **main.c** as a text file **assignment1.txt** with extension **.txt**
- You must upload **assignment1.txt** file to the Assignment Dropbox by the due date.
- Your submission must be unique or have references.
- **Late submissions are penalized 10% / day (1 point / day), up to 3 days.**
  Submissions won't be accepted after 3 days.

Grading Scheme:

- See **Main requirements** and also **Course_Introduction.pdf**.
  Deductions will be applied if partial functionality is provided.
- You'll get **zero grade** or **minimal grade** if your code doesn't compile.
- Compilation warnings are considered to be major mistakes.
- Comments are not required but recommended. However, "debugging code" or commented out "old code" is a minor mistake (unless it's clearly stated in the comments that you want me to take a look at such code).
- Please check extra and submission requirements for any additional deductions.