Hackathon 2 (Jan 25, 2022)

General Instructions:

Rules:

- The allowed libraries are stdio.h and stdlib.h (only for malloc, calloc, free).
- Your program should be modular. Do not write your entire program in main. Create suitable functions.
- For each function you create, leave a short comment above it describing what the function does.
- You are not allowed to use variable length arrays (VLA). All dynamic memory allocation must be on the heap.
- Your program should not have memory leaks. Free all heap memory used.
- Your program should accept input till EOF. You can detect this by checking if scanf returned -1.

Deadline: 1800hrs on Thursday 27th Jan,

Problem 1

- Input: Each instance will have $n \in \mathbb{N}$ followed by $a_1, a_2, \ldots, a_n \in \mathbb{N}$, and a number $k \in \mathbb{N}$.
  - Format:
    n `\n`
    a_1 a_2 $\ldots$ a_n `\n`
    k `\n`
- Goal: Check if there exists a contiguous subarray that has sum k. More precisely, check if there exists $1 \leq s \leq t \leq n$ such that $\sum_{i=s}^{t} a_i = k$.
- Output:
  - If s, t as above exist, then print `1\n`
  - Else print `0\n`

Remark: Every 3 lines of the input forms one input instance. Process each input instance as it is received. Stop at EOF.
See public test cases for examples.

Problem 2

- Input:
  - The starting four elements of a sequence that follows the pattern:
    $a, (ar + d), (ar^2 + d), (ar^3 + d), \ldots$
  - Format: Four numbers each separated by space. Terminated by `\n`.
- Goal: Find **integers** a, r, and d with $a \neq 0$ and $r \neq 0$ that satisfy the input pattern.
- Output: Print the values of a, r, and d, each separated by a space. Terminate the line with a `\n`.
- Example:
  - Input: 1 5 7 11 `\n`
  - Output: 1 2 3 `\n`

Remark: Every line of the input forms an input instance. Process each input instance as it is received. Stop at EOF.

See public test cases for more examples.

Problem 3

- Input: $n \in \mathbb{N}$ followed by a string s consisting of characters from the set $\{A, C, G, T\}$ of length at most n. This is followed by series of "pattern" strings $pat_1, pat_2, \ldots$, each of length at most n.
    - Format:
      n `\n`
      s `\n`
      pat_1 `\n`
      pat_2 `\n`
      $\ldots$
- Goal: For each pattern string pat_i given as input, find every occurence of pat_i in s.
- Output: Print the starting position of every occurence in s separated by a space. If pat_i does not exist, print -1. Terminate all output lines with `\n`

Remark: The first two lines of input define the main string s. This does not change for the rest of the input. The pattern strings are given as input in separate lines. Process each pattern string as it is received. Stop at EOF.

See public test cases for examples.