CS19003 : Programming and Data Structures Laboratory

Assignment 4 : One Dimensional Arrays in C

Date: 27-April-2021

Problem Statement – A:

In complicated cases, one which Detective Byomkesh Bakshi and Ajit Banerjee were investigating, the number of suspects keep on increasing with time. It has been noted in this case that, such increments of influential suspects follow a generalized Fibonacci sequence where the number of new suspects to be freshly/newly interrogated becomes the sum of the number of all previous suspects interrogated in k number of earlier interrogation phases (k may vary with the complicacy of the case).
To explain the progression in details, suppose Mr. Bakshi started the murder investigation by interrogating a number of $0, 1, 1, 2$ chosen suspects in previous $k = 4$ phases, respectively. Now, the number of new suspects to interrogate in 5-th phase will be $0 + 1 + 1 + 2 = 4$; in 6-th phase will be $1 + 1 + 2 + 4 = 8$; and so on. This indicates a generalized Fibonacci sequence of $0, 1, 1, 2, 4, 8, 15, 29, 56, 108, \ldots$ with $k = 4$ – indicating each term (from 5-th position onwards) will be the sum of the previous $k = 4$ terms.

Mathematically, the generalized Fibonacci sequences are expressed as:

$$a_1, a_2, a_3, \cdots, a_k, \quad a_{k+1}\Big(=\sum_{i=1}^{k} a_i\Big), \quad a_{k+2}\Big(=\sum_{i=2}^{k+1} a_i\Big), \quad \cdots, \quad a_n\Big(=\sum_{i=(n-k)}^{n-1} a_i\Big), \quad \cdots$$

Here, $a_1, a_2, \cdots, a_k$ are called the *base* (or initial) terms, and a_n (for all $n > k$) are called the *derived* terms. The number of initial terms provided is sometimes called the *order* of the generalized Fibonacci sequence.

Can you write a C-program to automate this process of computing upto n-th terms (i.e. including base and derived ones, $a_1, a_2, \cdots, a_k, a_{k+1}, \cdots, a_n$ for a given n and order k)?
In particular, your program will do the following:

- Take from user the order k of generalized Fibonacci sequence. Assume, $1 \leq k \leq 100$.
- Next, take all k base/initial terms from user and keep it in an array.
- Then, take from user total number of terms (n) to be computed (including base terms). Assume, $n \leq 10^6$.
- Compute j-th term, $a_j = \sum_{i=(j-k)}^{j-1} a_i$ $(k < j \leq n)$, from previous k terms and finally print all n terms, i.e. k base terms and $(n - k)$ derived terms.

Note: *You are* **not** *allowed to use any additional array other than the array that stores the initial k terms.*

Example Execution Details – A:

Sample-1:

```
++ Enter Order of Fibonnaci Sequence: 4
++ Enter Initial/Base Terms (space-separated): 0 1 1 2
++ Enter Number of Terms in Sequence to View: 10

** The Fibonacci Sequence upto 10-th Term:
 0  1  1  2  4  8  15  29  56  108
```

Sample-2:

```
++ Enter Order of Fibonnaci Sequence: 2
++ Enter Initial/Base Terms (space-separated): 1 1
++ Enter Number of Terms in Sequence to View: 20

** The Fibonacci Sequence upto 20-th Term:
 1  1  2  3  5  8  13  21  34  55  89  144  233  377  610  987  1597  2584  4181  6765
```

Submit a single C source file. Do not use global/static variables.

CS19003 : Programming and Data Structures Laboratory

Assignment 4 : One Dimensional Arrays in C

Date: 27-April-2021

Problem Statement – B:

Suddenly, Detective Mr. Bakshi and Ajit found a certain mathematical clue and similarity between the two murders that they were investigating. Especially, if they write the number of all the phase-wise interrogated suspects one after another (without space or comma separated manner individually for two murder cases), then the total common number of suspects may get identified by calculating the GCD (greatest common divisor) of these two *huge* number-sequences. For example, let the number of interrogations for the first murder (phase-wise) was $1, 2, 3, 5, 8, 13, 21$, which forms the huge number $N_1 = 123581321$. Again, the number of interrogations for the second murder (phase-wise) was $7, 2, 2, 8, 15, 2$, which forms the huge number $N_2 = 7228152$. So, the overall number of common suspects will be $= \texttt{GCD}(\texttt{N}_1, \texttt{N}_2) = \texttt{GCD}(123581321, 7228152) = 100391$. Please note that, the length of N_1 and N_2 (say n_1 and n_2) can be arbitrarily large containing a maximum of 10^9 digits.

Can you write a C-program to automatically compute the *greatest common divisor* (GCD) of two *huge* numbers? In particular, your program will do the following:

- Take from user the (positive) lengths ($1 \leq n_1, n_2 \leq 10^9$) of two numbers. Subsequently, scan in these two positive numbers as sequence of characters and keep it in two separate character arrays, say `A[ ]` and `B[ ]`.
- Compute the GCD of these two huge numbers by operating over the two character arrays, `A[ ]` and `B[ ]`.
- Finally, print the GCD as an array of characters resulted from the computation in above step.

Example Execution Details – B:

Sample-1:

```
++ Enter Length of First Number: 20
++ Enter First Number: 11223344556677889900
++ Enter Length of Second Number: 3
++ Enter Second Number: 666

** The GCD is: 18
```

Sample-2:

```
++ Enter Length of First Number: 10
++ Enter First Number: 1000000000
++ Enter Length of Second Number: 5
++ Enter Second Number: 99999

** The GCD is: 1
```

Sample-3:

```
++ Enter Length of First Number: 10
++ Enter First Number: 1234567890
++ Enter Length of Second Number: 100
++ Enter Second Number: 1111111111222222222233333333334444444444555555555566666666667777777777888888888899999999990000000000

** The GCD is: 90
```

Sample-4:

```
++ Enter Length of First Number: 40
++ Enter First Number: 1234567890123456789012345678901234567890
++ Enter Length of Second Number: 40
++ Enter Second Number: 1234567890123456789012345678901234567890

** The GCD is: 1234567890123456789012345678901234567890
```

Submit a single C source file. Do not use global/static variables.