Assignment 1: 'Revising' C Concepts

CS2.201a Computer Systems Organization (CSO)

January 30, 2023

1. (10 marks) Write a C program that implements a doubly linked list for heap allocated strings. This task will be graded automatically, so please follow the input/output format carefully.

   The first line of input will contain a single integer q, $1 \leq q \leq 10^5$ the number of queries you will have to process. Each query will be in one of the following 4 types.

   - 1 pos str - This means, insert the string str at position pos in the current list. It is guaranteed that the first query will have $pos = 0$. To clarify, pos is the number of elements in the list from the left after which you insert str. For example, if the current list was

     $$"abc" \iff "def" \iff "ghi"$$

     an insert to $pos = 2$ would result in the following list:

     $$"abc" \iff "def" \iff "new\ string" \iff "ghi"$$

   - 2 id - Delete the string at position pos in the current list. It is guaranteed that this query will always correspond to a valid position in the current list. id here is the **0-indexed** position of the element in the list.
   - 3 l r $(l \leq r)$ - Print all the strings (space separated) that satisfy $l \leq id \leq r$ in increasing order of pos (from left to right)
   - 4 l r $(l \leq r)$ - Print all the strings (space separated) that satisfy $l \leq id \leq r$ in decreasing order of pos (from right to left)

     Sample input

     ```
     7
     1 0 hello
     1 0 world
     1 2 pointers
     4 0 2
     3 2 2
     2 0
     3 0 1
     ```

     Sample output

     ```
     pointers hello world
     pointers
     hello pointers
     ```

Note: It is guaranteed that the sum of the characters you'll have to read and print will be $\leq 10^5$.

2. (10 marks) Shoreki and Ramsri get access to a treasure vault that has n boxes in it. Each box contains a_i amount of coins in it. They both decide to play a game to decide who gets how many coins. The rules of the game are as follows:

   The game occurs in turns and Shoreki has the first turn. During their turn, the participant can either choose to **keep** the next box's contents to themselves and let the other person decide for the next turn **OR** They can **give** the next box to the other person and have the decision power for the next turn too. The first turn is played on box a_1, the next on box a_2, and so on.

   Each of them want to maximize the amount of coins gained and hence they make decisions optimally. How much coins will they each have at the end?

   Input format

   The first line of the input will contain an integer $n(1 \leq n \leq 10^4)$. The next line of input will contains n space separated integers $a_i(1 \leq i \leq n, 1 \leq a_i \leq 10^5)$.

   Output format

   Print two space separated integers, s and k. The number of coins Ramsri and Shoreki gain respectively if they play optimally.

   Sample input

   15
   3026 3027 4599 4854 7086 29504 38709 40467 40663 58674 61008 70794 77517 85547 87320

   Sample output

   306375 306420

3. (3 marks) **Note:** The following code contains undefined behavior and *may not* show the **exact** behavior described when run on your system.

   The code we're referring to can be found here.

   This code contains some undefined behavior due to which the following effects were observed on our target machine.

   - The code was compiled as given using the command *gcc printf.c -o printf* and run with the input:

     ```
     input.txt
     ```

     ```
     fname:WhyC
     lname:Again
     age:21
     ```

     It executes without any issue and provides the expected output:

     ```
     #> ./printf < input.txt
     Age: 21
     First name: WhyC
     Last name: Again
     ```

- Now, the *printf* on line **23** is commented / removed and the program is run again with the exact same compilation options and input as before. However, this time the program segfaults!

We are extremely puzzled by this bizarre behavior and want to know the reason behind this voodoo. There are two parts to this problem, first try to identify the bug in the program that is causing the undefined behavior. Second, try to reason why the bug does not cause a segfault when the *printf* statement is present in the code.

Hints:

(a) *The bug is related to the memory allocation and subsequent accesses that are taking place in the program.*

(b) *To figure out why printf "fixes" the bug, think about how memory is usually allocated to a program and about the internal workings and requirements of printf. You do not need to read printfs library code, just reason about what printf requires to work.*