College of Computing and Informatics, Sungkyunkwan University

Problem Solving 2022 Fall
Assignment 1

Instructor: Jaemin Jo (jmjo@skku.edu)

September 25, 2022

1 Introduction

In this assignment, you will implement an interactive console program, `SKKU-SEQ`; it takes a command from the user as text input through the console, executes the command, and prints the result to the console until an exit command is entered. `SKKU-SEQ` manages multiple integer sequences. Let N be the number of sequences ($N \leq 300$). Let S_i and S_{ij} denote the i-th sequence and the j-th term of S_i, respectively ($j \leq 300$ and $S_{ij} \leq 10,000$). Initially, there is no sequence on `SKKU-SEQ` ($N = 0$). The user can create/modify/remove a sequence or term using the following commands:

- `new`: The user can add a new arithmetic sequence (등차수열) from the given first term (초항) and common difference (공차).
- `insert`: The user can insert a term in a specific position of a sequence.
- `rseq`: The user can remove a sequence.
- `rrange`: The user can remove the terms of a sequence specified by a range.
- `rone`: The user can remove a specific term from all sequences.
- `sum`: The user can compute the sum of the terms in each sequence.
- `exit`: The user can exit the program.

2 The `SKKU-SEQ` Commands

A command is a single-line string that ends with "\r\n" and consists of *command_type* and *arguments* as follows:

$$<command> := <command\_type> <arg_1:type_1> <arg_2:type_2> \cdots <arg_n:type_n>$$

command_type represents the type of the command and is one of: `new`, `insert`, `rseq`, `rrange`, `rone`, `sum`, and `exit`.

arg_i and $type_i$ are the name and type of the i-th argument in the command, respectively. The type of an argument is one of:

- *size*: a positive integer ranging from 1 to 300 (inclusive),
- *index*: a non-negative integer ranging from 0 to 300 (inclusive),
- *number*: a non-negative integer less than 10,000.

IMPORTANT: It is guaranteed that an exact number of arguments are given (no insufficient or superfluous arguments). An argument is guaranteed to have the type that is specified in this document (no wrong type).

2.1 The `new` Command

Usage: `new <first:`*number*`> <diff:`*number*`> <m:`*size*`>`

The `new` command creates a new arithmetic sequence A of length m, where $A_i = first + (k-1) \cdot diff$ $(0 \leq k < m)$, and adds it as the last sequence, incrementing the number of sequences in the system, N, by one.

Example: Suppose the user enters "`new 2 3 4`" and then "`new 1 0 3`" (on separate lines) after the program starts. The first command will add $S_1 = [2, 5, 8, 11]$. Then, the second will add $S_2 = [1, 1, 1]$.

2.2 The `insert` Command

Usage: `insert <i:`*size*`> <j:`*index*`> <num:`*number*`>`

The `insert` command inserts a term *num* to the position after the j-th term of the i-th sequence. Note that *num* is inserted after the j-th term. If j is 0, *num* is inserted before the first element of the sequence, becoming the first term.

Example: Suppose there are two sequences $S_1 = [2, 5, 8, 11]$ and $S_2 = [1, 1, 1]$. "`insert 1 2 3`" will change S_1 to $[2, 5, 3, 8, 11]$, and "`insert 2 0 7`" will change S_2 to $[7, 1, 1, 1]$.

2.3 The `rseq` Command

Usage: `rseq <i:`*size*`>`

The `rseq` command removes the i-th sequence, S_i. Note that once S_i is removed, the sequences after S_i should replace the one before, eventually decrementing N by one. See the example below.

Example: Suppose there are two sequences $S_1 = [2, 5, 8, 11]$ and $S_2 = [1, 1, 1]$. "`rseq 1`" will remove S_1. Then, there is only one sequence remaining, $S_1 = [1, 1, 1]$ (S_2 becomes S_1 since the previous S_1 is gone). If the user enters the same command "`rseq 1`" again, the only sequence remaining $S_1 = [1, 1, 1]$ will be removed as well. So after the two commands, there will be no sequences.

2.4 The `rrange` Command

Usage: `rrange <i:`*size*`> <start:`*size*`> <end:`*size*`>`

The `rrange` command removes the terms S_{ij} ($start \leq j \leq end$). Note that S_i can be completely removed when $start = 1$ and $end = len(S_i)$. In this case, as in the `rseq` command, the sequences after the i-th sequence should replace the one before.

Example: Suppose there are two sequences $S_1 = [2, 5, 8, 11]$ and $S_2 = [1, 1, 1]$. "`rrange 1 2 3`" will change S_1 to $[2, 11]$ (removes the second and third terms of S_1). Then, "`rrange 1 1 2`" will completely remove S_1, making S_2 be the first sequence.

2.5 The `rone` Command

Usage: `rone <num:`*number*`>`

The `rone` command removes the term *num* from all sequence. Note that there can be sequences completely removed. In this case, as in the `rseq` command, the sequences after those completely removed sequences must be shifted so that there is no "empty" sequence.

Example: Suppose there are two sequences $S_1 = [1, 1, 2]$ and $S_2 = [3, 1]$. "`rone 1`" will change S_1 to $[2]$ and S_2 to $[3]$. Then, "`rone 2`" will completely remove S_1. After the two commands, there will be only one sequence $S_1 = [3]$.

2.6 The `sum` Command

Usage: `sum`

The `sum` command computes the sum of the terms of S_i and prints out the sum, one on a line. This is the only command that prints something to the console. Print out nothing if

there is no sequence ($N = 0$).

Example: Suppose there are two sequences $S_1 = [1, 1, 2]$ and $S_2 = [1, 2]$. "`sum`" will print out 4 and 3 on separate lines.

2.7 The `exit` Command

Usage: `exit`

The `exit` command terminates `SKKU-SEQ`.

3 Input and Output Examples

Input 1

```
new 1 0 3
new 2 2 3
new 3 1 4
sum
exit
```

Output 1

```
3
12
18
```

In this example, $S_1 = [1, 1, 1]$, $S_2 = [2, 4, 6]$, and $S_1 = [3, 4, 5, 6]$.

Input 2

```
new 1 0 3
new 2 2 3
new 3 1 4
new 2 1 3
rseq 2
insert 2 0 2
insert 3 3 1
rone 1
sum
exit
```

Output 2

```
20
9
```

After the first four `new` commands and `rseq` commands, we get three sequences: $[1, 1, 1]$, $[3, 4, 5, 6]$, and $[2, 3, 4]$. After the `insert` commands, we have $[1, 1, 1]$, $[2, 3, 4, 5, 6]$, and $[2, 3, 4, 1]$. After the `rone` command, we have $[2, 3, 4, 5, 6]$ and $[2, 3, 4]$ that sum up to 20 and 9, respectively.

4 Rules and Tips (READ THIS CAREFULLY!)

- Your code must be in a single source file, **main.c**. Do not create any separate header or source files. A skeleton code file will be given in iCampus. Feel free to modify it.
- Your code will be compiled by gcc 11.1. You must test your implementation on Goorm.
- You can only use the standard functions of C. See the list of standard header files: `https://en.cppreference.com/w/c/header`
- Your program will be automatically graded by a program. Therefore, be careful about the input and output formats. Do not print extra characters/spaces/newlines that are not specified in this document, such as "Welcome", "Enter a command>", or "Available commands are...".
- The right behavior of your program is **undefined** for cases that are not specified in this document; for the wrong command name, wrong argument number and types, or invalid ranges, you can do anything. You can print an error message, ignore it, or even exit the program. You can assume that testcases will be syntactically and semantically correct.
- A testcase consists of up to 100 commands.
- Your program will be given **1** second and 2 GB of memory for each testcase.
- The program must exit with code 0, i.e., returning 0 at the end of the main function.
- Use only `scanf("%s", ...)` to read a string input. Do not use `"%c"` or `getchar()`. Do not use non-standard, non-cross-platform, should-be-avoided, and I-didn't-teach-you statements such as `scanf("%[^\n]*c", ...)`.
- Any array that exceeds 1 KB must be defined globally or allocated dynamically. No `int main() { int a[300][300]; ... }`.
- Do not use variable length arrays. No `int main() { int n; ... int a[n]; ... }`.
- Double-check the range for N, $len(S_i)$, and S_{ij}.

5 Submission

The deadline is **October 14th 23:55 KST**. You need to submit one C file **main.c** to iCampus.

6 Copyright

You will hold the copyright of your work. I will not copy/distribute/modify your work except for the grading.

7 Plagiarism

Do NOT copy others' source codes. Do NOT transfer any part of your code through email, messenger, text, etc. Both cheater and code-giver will get a penalty if their codes are similar enough.