UTEK

University of Toronto Engineering Kompetition

January 11-12th, 2020

PROGRAMMING COMPETITOR PACKAGE

Director:

Dhruv Warrier

UTEK 2020 Sponsors:

HUAWEI

Welcome to the UTEK Programming Competition for 2020! Over the next 7 hours, you will solve a programming challenge, compete with other teams, and end with a presentation on your solution. Let the games begin!

Rules

1. You must collaborate only with members on your team.
2. You can use any language and its **standard libraries**.

Problem Statement

The Librarian's Dilemma

Astrid works at the Library of Babel, arranging books for minimum wage to put herself through a Computer Science degree. Recently, the library's catalog was updated and the head librarian is looking for reasons to fire Astrid! She gives her the task of rearranging and restocking all the shelves. But it's a daunting job! There's thousands of new books to be added, and thousands more to remove or replace.

Astrid knows the current arrangement of the books, and is given a new arrangement for each shelf. Your goal is to generate an optimal set of moves to rearrange the shelves. Astrid's future is at risk, and she's counting on you!

You can assume that the shelves are infinite in size. Solutions will be judged on the optimality of the moves generated.

!	***Boxes like these provide tips for 1st years/frosh! Skip them if you wish.***

Input/Output Syntax

Input test files have extension **.in**. They will be posted on Discord.

You must generate output files with extension **.out**.
Output files contain a series of commands that modify the arrangement of the books.

Following is the syntax for all valid commands (<*start*> and <*end*> are 0-indexed):

`Insert <start>-<end>, "<book1>" "<book2>"...`
Inserts a list of books starting from index *start* to index *end*. Expects *start - end + 1* book names to follow.
`Delete <start>-<end>`
Removes books from index *start* to index *end*.
`Replace <start>-<end>, "<book1>" "<book2>"...`
Replaces the books from indices *start* to *end* with new books. Expects *start - end + 1* book names to follow.

A modification **starts** at the `<start>` index and **ends** at the `<end>` index *(i.e. an* `Insert 0-0` *means that the new element takes the 0th position, pushing the previous 0th element down)*.

Here is a sample **input file:**

```
Original: 4
Ancient Egypt, unknown
Tales of Horror, Bram Stoker
Foundation, Isaac Asimov
Intro to Algorithms, Thomas Cormen
Desired: 5
Religion and Mythology, Neil Potts
Ancient Egypt, unknown
Intro to Algorithms, Thomas Cormen
Foundation, Isaac Asimov
Intro to Algorithms, Thomas Cormen
```

a) The old arrangement is provided first, followed by the desired arrangement.
b) The number of elements in each arrangement is given at the start of the arrangement.

Following is a sample **output file** for the above input:

```
Insert 0-0, "Religion and Mythology, Neil Potts"
Delete 2-3
Insert 3-4, "Foundation, Isaac Asimov" "Intro to Algorithms, Thomas Cormen"
```

a) Each line is a **command** that performs operations on the original arrangement to manipulate it into the desired arrangement.
b) The commands are **executed sequentially** and **immediately.** The indices of elements are thus modified each time a command is executed. Inserting an element at index 0, for example, will push the remaining elements down by 1.

In the example output file, is the choice of moves optimal?

Here is an example of better output:

```
Insert 0-0, "Religion and Mythology, Neil Potts"
Replace 2-2, "Intro to Algorithms, Thomas Cormen"
```

a) Fewer **changes** were made to the shelves *(1 item inserted, 1 item replaced)*
b) Remember that fewer moves **may not necessarily be more optimal!**
Consider the trivial example:

```
Delete 0-<end>
Insert 0-<end> “<book1>” “<book2>” “<book3>” ...
```

In 2 moves the entire arrangement can be replaced with the desired one. But this is the least optimal output!

Requirements

Your submission **must** be able to take the path of an input file as a command line argument *(i.e. when the executable/program is called from the command line, it should accept a path to an input file).*

Examples for Linux and Windows *(on Terminal and Command Prompt respectively)*:

```
./<name_of_your_executable> </path/to/input/file>.in
<name_of_your_executable>.exe </path/to/input/file>.in
```

!	Most environments will already do this for you, or will have libraries to do this work for you! Some examples: a) In C/C++, `int main(int argc, char ** argv)`

    i) `argc`: number of arguments
    ii) `argv`: an array of string arguments *(Even if no arguments are provided on the command line, the system always provides one argument, which is the executable's file path)*.

b) In Python, you can use the `argparse` library:

```
import argparse
parser = argparse.ArgumentParser(description='Some description')
args = parser.parse_args()
```

Deliverables

1. *Code:* Code and documentation for all challenges, and a README with build/run instructions. **Due at 5:30 pm.**
2. *Presentation:* A presentation outlining the key aspects of the problem, your approach, and your solution. **Due at 6:30 pm.**
    a. Due to time constraints, only the **top 5 teams** will present on Sunday. You will be notified of your selection by email/Slack on Saturday night.

	Time	**Details**
Team Presentation	10 min	Please keep your presentation to 10 min. If you go over time, you will get a **30s** grace period before being cut off.
Judges Q&A	5 min	Q&A for all members of the team.

!

1. Start your presentation early! If you don't finish all challenges by the deadline, you can still present if you describe your solution in the presentation.
2. Start with the command line input parser, since this is **mandatory!**
3. If you want to quickly find an element common to two arrays, you can use a **hash table** for much better search performance than an array (*O(1) as compared to* $O(n^2)$). Most environments should already provide this in a library (*for example, on C++, `std::unordered_map` is a hash table*)

How to submit

a) Submit your code and input/output files zipped to programming@utek.skule.ca, with the subject line **[UTEK 2020 Submission] Team N**, where N is your team number.
b) CC your teammates and include your names in the body of the email.
c) Email your presentation to the same email address with the subject **[UTEK 2020 Presentation] Team N**, where N is your team number.

Part 1: Input and output 20 pts

Astrid has a massive number of shelf arrangements to go through! She figures that a good place to start is to get the input and output formats right.

> a) *Write a parser function that reads in an input file, and loads the book names in each arrangement into separate collections/arrays.*
>
> b) *Write a generator function to produce output with valid syntax* ***(see Input/Output Syntax)****. For simplicity, all the test inputs for this part can be solved with just one move.*

For this part, you need only submit output files for b).

Part 2: A simpler, related problem 100 pts

Astrid now starts thinking about the core of the problem. She quickly realizes it is very similar to one she solved in puzzle books as a kid!

She rips out her old notebook and finds one of those puzzles:

```
What is the least number of edits required to convert the string “KATE”
to "CATS"? An edit can be one of:
   -  Insertion of a single character (for eg. “TABLE” -> “TABLES”)
   -  Deletion of a single character (for eg. “TABLES” -> “ABLES”)
   -  Replacement of a single character (for eg. “ABLES” -> “ABLED”)
If you’re done, try “SATURDAY” -> “SUNDAY”, “KITTEN” -> “SITTING”, and
“CARTHORSE” -> “ORCHESTRA” too!
Answers: 2 edits, 3 edits, 3 edits, 8 edits
```

By trial and error, she solves one of them:

“SATURDAY” → “STURDAY” → “SURDAY” → “SUNDAY” (delete, delete, replace)

She jots down some ideas:

1. An insertion followed by a deletion from the same location in the string, is equivalent to replacement. But the former is two edits, while the latter **is only one**!
2. Insertion and deletion both modify the length of the string, while replacement does not. If the length of the original and target string vary, some insertions or deletions **must** have taken place.
3. **Multiple solutions exist** to the problem. For eg. more than one solution exists to convert “OPERATING” to “CONSTANTINE” in 6 edits.

Develop a solution to this problem.
Hints: *a) The problem is best expressed in matrix form.* *b) Solutions can be reached in $\Theta(M \cdot N)$, where M and N are the lengths of the original and target strings, respectively.* *c) For this simpler version of the problem, solutions that take the same number of edits are graded the same. In Part 3, edits can be grouped together to produce a smaller number of moves!*

!	a) You can get partial points if your implementation generates valid solutions, even if they take more edits than required. b) Solve this on paper first - it really helps to look at this problem visually.

For this part only, the input and output format is slightly different. Following are the output commands (<*index*> is 0-indexed):

`Insert <index>, ‘<char>’`
Inserts <*char*> at the given <*index*>. For eg. *Insert 1, ‘c’* on “brain” gives “bcrain”.
`Delete <index>`
Deletes the character at the given <*index*>.
`Replace <index>, ‘<new-char>’`
Replaces character at <*index*> with <*new-char*>.

Each input file contains two strings, the first is the original and the second the target string. For example:

```
saturday
sunday
```

Example output for this input:

```
Delete 1
Delete 1
Replace 2, ‘n’
```

Part 3: Astrid becomes a bookworm 200 pts

Astrid puts the final touches on her code and hits “Compile”. Woohoo! It works without a hitch. She’s almost there.

She begins to tackle the full book problem. She jots down some notes:

1. Moves/edits can be made in batch (i.e. **Insert**s, **Delete**s or **Replace**s to contiguous locations can be combined into one command).

```
Insert 0
Delete 1
Insert 1
```

```
Insert 0
Insert 1
Delete 2
```

   a. In the previous part, the above solutions would have been treated the same. But when moves can be combined, **solution 2 (on the right)** would be the better choice since its **Insert**s are contiguous!

Develop a solution to this problem.

Hints:

- *Keep in mind the solution here is heuristic, so there is no “good” solution. Your solutions will be graded on their optimality by how close they are to reference solutions.*

Grading

80% of the points are awarded based on how many test files are passed.

The grading is done automatically by an autotester. For solutions that are not optimal but close to it, partial points will be awarded.

20% is awarded for code style + readability.

Good luck!

Presentation Rubric

Category	Points	Details
Well-justified	50	Explanations are detailed, and your reasoning is backed with examples.
Presentation	30	The presentation should be easy to follow and flow naturally. Competitors should be prepared to answer questions on their solution.
Choice of visuals	20	Your choice of visuals should make a meaningful contribution to the presentation.
Total	100	