CS300 – Spring 2018-2019 - Sabancı University

Homework #1 – The Word Processor

March 12th Tuesday 22:00

In this homework, you will implement a program that acts as a word processor working on the word entered in the console window. This word processor will allow the user to enter a string from the console, and then will operate as described below using the given input string and display the resulting string. The one and **ONLY** data structure allowed in this homework is the **Stack**.

The word processor needs to have the following functions. While reading the input string from left to right,

1. **End Input:** If the current character is “1”, it will be interpreted as “ESC” to end the editing process of the input string and display the output.
2. **Delete:** If the current character is “2”, it will be interpreted as a backspace and will delete the element on the left of the current cursor position.
3. **Move Cursor Left:** If the current character is “3”, it will be interpreted as a “Left Arrow Key” and will move the invisible cursor to the character on the left of its current position. If the cursor is already at the beginning of the string then nothing will change.
4. **Move Cursor Right:** If the current character is “4”, it will be interpreted as a “Right Arrow Key” and will move the invisible cursor to the character on the right of its current position. If the cursor is already at the end of the string then nothing will change.
5. **Reverse Text:** If the current character is “5”, it will be interpreted as a “reverse button”, where it will reverse (flip) the text on the left of the cursor. Example: “Sabanci5” will become “icnabaS”.
6. **Find and Delete:** If the current character is “6x”, it will be interpreted as a “Find and Delete button”. It will search on the left side of the cursor for all occurrences of “x” and delete them. Here “x” can be any character. Example: “I_love_computer_sc6eience” will become “I_lov_computr_science”.

NOTE THAT when a word processor function number such as 1, 2, 3, 4, 5, or 6 is processed, the number character itself will be deleted after the corresponding operation is done. Also note that in any input sequence if the corresponding function cannot be applied (such as an invalid number 7, 8, 9 or two valid numbers one after another such as 61), you may just ignore it as invalid input and not process it.

For an example;

Input: sit3h4_isa_an_33336a33354444444example!1

Our output string will be based on the manipulations from the numbers. Here is a walkthrough. First number we see is ‘3’, so before that we don’t need to do anything. When we see ‘3’ we will move our invisible cursor left by one to be in between the letters ‘i’ and ‘t’ our next input is ‘h’ so it will go in between ‘i’ and ‘t’ to have result of “siht”. When we keep going we see the number ‘4’, which means move cursor right by one. Now our cursor is after the ‘t’. Then we don’t see a number till the next ‘3’ so our resulting string is now “siht_isa_an_”. Then we move our cursor left four times because there are four ‘3’ numbers in the string. Now our cursor is between ‘a’ and ‘_’ in the “isa_” string. Our next number is ‘6’ and it is followed by an ‘a’. This means find all the ‘a’ characters on the left of our cursor. Since the ‘a’ in ‘an_’ is now on the right of our cursor it won’t be deleted, but the ‘a’ in “_isa” will be because it’s in the left of our cursor. Our resulting string will be “siht_is_an_”. Following this, we will keep moving our imaginary cursor left by the amount of ‘3’ numbers we see. After this our cursor will be between ‘t’ and ‘_’. Our next operation is ‘5’ which is the operation that reverses the letters on the left of our cursor. At this moment only “siht” is in the left of our cursor, so when we reverse it the result will be “this” and our resulting whole string will be “this_is_an_”. Our next number is a series of ‘4’. The number ‘4’ means move the cursor right so we will move the cursor right by the count of ‘4’s we see. This will bring us to the end of the string and our cursor will be at the end now. After this our next input is “example!” so our resulting string will be “this_is_an_example!”. Finally the number ‘1’ will mean the string processing will be ended and we will print our resulting string “this_is_an_example!”.

The Stack

You can use any stack implementation you find with the condition that it is a templated class. You can find stack implementation from your slides, textbooks or the web. One such interface is given below:

```
template <class Object>
class Stack {
   private:
      // Internal data representation

   public:
      Stack();

      void push(Object newItem);
      void pop();

      Object top();
      bool isEmpty();

      const static Object noItem;
};
```

To do this homework, you **MUST** implement a **Stack** using whichever augmented data structure you think is appropriate. The interface of the stack class can be found in your lecture slides, and you need to implement the function bodies as necessary. Other than that, **YOU ARE NOT ALLOWED TO USE ANY OTHER DATA STRUCTURE IN YOUR MAIN PROGRAM!!! NOT EVEN VECTOR OR ARRAY.**

Notice that the Stack class is a templated class. Using this templated class you may want to store basic data types, structs or classes in order to implement this homework.

Input and Output

The input will be given as a string, it will have no spaces and no '*' characters in it. You may also assume that the input will **NOT** have any digit characters in it other than the special numbers ('1', '2', '3', '4', '5', '6') with a processing functional meaning. Your program will read the given input and parse it according to the functions specified in the "Functions" section above. After the input is correctly parsed, you need to print the output to the console window.

Important: The one and **ONLY** data structure allowed in this homework is the **Stack**.

HINT: You can use multiple stacks, but all the data structures you need have to be stacks. The use of stacks allows this program to run in O(N) time where "N" is the number of characters in the input.

DEMO EXAMPLES

Example 1:

```
Hakan_Okan222gan_Alpar1
```

Output:

```
Hakan_Ogan_Alpar
```

Example 2:

```
Gulsen_Dmiroz33333e1
```

Output:

```
Gulsen_Demiroz
```

Example 3:

```
napda3351
```

Output:

```
panda
```

Example 4:

```
I_Love_Str3333344Data_444uctures1
```

Output:

```
I_Love_Data_Structures
```

Example 5:

```
Letz’sa/_h6aazzzve_/a/_good_semester6z6/!!!!1
```

Output:

```
Let's_have_a_good_semester!!!!
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS300. Office hours of TAs are here as well as at the syllabus. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Submissions guidelines are below. The submission steps will get natural/easy for later homeworks. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- Name your cpp file that contains your program as follows.
  "**SUCourseUserName_yourLastname_yourName_HWnumber.cpp**"
- Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:
  cago_ozbugsizkodyazaroglu_caglayan_hw1.cpp
- Do not add any other character or phrase to the file name.
- Make sure that this file is the latest version of your homework program.
- You need to submit ALL .cpp and .h files in addition to your main.cpp in your VS solution.

Submission:

- Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).
  1) Click on "Assignments" at CS300 SUCourse.
  2) Click Homework 1 in the assignments list.
  3) Click on "Add Attachments" button.
  4) Click on "Browse" button and select the zip file that you generated.
  5) Now, you have to see your zip file in the "Items to attach" list.
  6) Click on "Continue" button.
  7) Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Resubmission:

- After submission, you will be able to take your homework back and resubmit. In order to resubmit, follow the following steps.
  1) Click on "Assignments" at CS300 SUCourse.
  2) Click Homework 1 in the assignments list.
  3) Click on "Re-submit" button.
  4) Click on "Add/remove Attachments" button
  5) Remove all existing files by clicking on "remove" link. This step is very important. If you don't delete the old files, we get both files and the old one may be graded.
  6) Click on "Browse" button and select the new zip file that you want to resubmit.
  7) Now, you have to see your new cpp files in the "Items to attach" list.
  8) Click on "Continue" button.
  9) Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Grading and Objections

Careful about the semi-automatic grading: Your programs will be graded using **more complex** sample runs different than the ones given in this document Demo section, so make sure you test your program with different combinations of these sample runs.
Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- Late penalty is 10% off the full grade and only one late day is allowed.
- **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your homework from the email address provided in the comment section of your announced homework grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the homework tab to see the problem with your homework.
- Download the .zip file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

Good Luck!
Hakan Ogan Alpar and Gülşen Demiröz