CS201 – Fall 2021-2022

Take-Home Exam 3
– Palindrome Search in Reverse Order –
Due November 24, Wednesday, 23:55 (Sharp Deadline)

Introduction

The aim of this take-home exam is to make students comfortable with loops (while and for) and strings (operating on strings using member functions) in C++ as well as using computational approaches to solve analytical questions.

Description

In this homework, you will write a program that will parse a text to sentences and perform a reverse operation on them. You will take the inputs from the console, identify each sentence in the input and process each sentence from the last to the first and starting from the last word to the first word. You will prompt the word itself if the word you process is a palindrome, otherwise you will display a message indicating the word is not a palindrome. A palindrome is a word that reads the same backwards as forwards (Example: ata, radar, refer, ege etc.).

You **cannot** write all of your code under the main function, i.e. you **must** write user-defined functions and use them.

Your take-home exams will be automatically graded using GradeChecker, so it is very important to satisfy the exact same output given in the sample runs. You can utilize GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check whether your implementation is working in the expected way. To be able to use GradeChecker, you should upload all of your files used in the take-home exam (**only** *your_main_cpp* file for this take-home exam). Additionally, you should submit all of your files to SUCourse (**only** *your_main_cpp* file for this take-home exam) **without zipping** them. **Just a reminder, you will see a character ¶ which refers to a newline in your expected output.**

The name of your main source (cpp) file should be in the expected format: "SUCourseUsername_THEnumber.cpp" (all lowercase letters, e.g. gulsend_THE3.cpp). Please check the submission procedures of the take-home exam, which are listed at the end of this document.

Inputs and Input Checks

You only take the input text at the beginning of your program. For this input you should follow the rules below:

- All inputs are strings.
- There can be one or more spaces or tabs between these inputs (Hint: use a loop to take inputs).
- You can assume every sentence ends with a dot (‘.’) character and there will be no words that contains a dot (‘.’) inside it (i.e. "ber.ker" is not a valid word and **you do not need to check it**. However, "berker." is valid because it indicates that the sentence ends).
- Input words can contain non-alphabetical characters but they cannot consist of only non-alphabetical characters without any letters (i.e. "a12@341?*&BC" is a valid word but "123" is not).
- The input should contain at least one sentence.
- Empty string is not a valid input.
- The word "@" is used to notify the program that the input has ended.

Important! You can assume that each sentence ends with a dot in the given inputs. After each dot, there will be a space or a new line before the next sentence or "@".

So, you must implement all the necessary checks for the inputs described above. If there is an error in the input, your program should display an appropriate message and ask for it again.

Processing, Program Flow and Outputs

Your program should start with a prompt that asks for the input sentences. After a correct input is entered, your program should calculate how many sentences the input contains and display this information to the user. After that, your program will process the input and eliminate non-alphabetical characters in words and convert alphabetical characters to lowercase letters. For example if the input string is "12Gu@LsE*n, B4aRiS8!, BErk1432eR." you need to convert it to "gulsen baris berker.". Then, starting from the last sentence (Hint: dots indicate the end of the sentences, so you should use this information to parse the string) and processing each sentence in reverse order, you need to display the word itself if it is a palindrome, otherwise it should prompt the word "notpalindrome". For example; if the sentence you process is "ata ege berker radar." your program should display "radar notpalindrome ege ata". Comparing the strings, the input check and the other operations in the program require using some of the string member functions (like length, find, rfind, substr, at etc.) that are covered in class.

The inputs and the order of the program are explained above. It is extremely important to follow this order with the same characters since we automatically process your programs. ***Thus, your work will be graded as 0 unless the order is entirely correct***. Please see the "Sample Runs" section for some examples.

Important Remarks

Unlike the second homework, we will not specify any functions here. But you are expected to use functions to avoid code duplication and improve the modularity of your program. **If your main function or any user-defined function is too long and if you do everything in main or in another user-defined function, your grade may be lowered. Please do not write everything in main and then try to split the task into some functions just to have some functions other than mail. This is totally against the idea of functional design and nothing but a dirty trick to get some points. Instead please design your program by considering the necessary functions at the beginning.**

Try to use parametric and non-void functions wherever appropriate. Do NOT use any global variables (variables defined outside the functions) to avoid parameter use.
In this homework (and in the coming ones) you are not allowed to use instructions such as “exit” and “goto”. These cause difficulties to control the flow of your programs. Thus, we do not approve of using them. You are also not encouraged to use “break” and “continue”. The use of “break” and “continue” prevent you from forming good readable loop conditions and hence prevent you from learning how to form good loops. Think cleverly in order not to use any of these instructions. If you don't know these commands, do not even try to learn them (we will explain “break” in class).

IMPORTANT!

If your code does not compile, then you will get **zero**. Please be careful about this and double check your code before submission.

VERY IMPORTANT!

Your programs will be compiled, executed and evaluated automatically; therefore you should definitely follow the rules for prompts, inputs and outputs. See **Sample Runs** section for some examples.

- **Order of inputs and outputs** must be in the mentioned format.

Following these rules is crucial for grading, otherwise our software will not be able to process your outputs and you will lose some points in the best scenario.

Sample Runs

Below, we provide some sample runs of the program that you will develop. The *italic* and **bold** phrases are inputs taken from the user. You have to display the required information in the same order and with the same words and characters as below.

Sample Run 1

```
Please enter the input sentences: @
Input string should not be empty.

Please enter the input sentences: . @
There should be no words without alphabetical characters.

Please enter the input sentences: berker defne @
There should be at least one sentence.

Please enter the input sentences: berker 123 emre. @
There should be no words without alphabetical characters.

Please enter the input sentences: berker de1n2i3z efe. @

Sentence (1/1):
efe notpalindrome notpalindrome
```

Sample Run 2

Please enter the input sentences: **555 @**
There should be at least one sentence.

Please enter the input sentences: ***555. berker. @***
There should be no words without alphabetical characters.

Please enter the input sentences: **a. b. c d. e f g. notpalindrome. yes it is:). @**

Sentence (6/6):
notpalindrome notpalindrome notpalindrome
Sentence (5/6):
notpalindrome
Sentence (4/6):
g f e
Sentence (3/6):
d c
Sentence (2/6):
b
Sentence (1/6):
a

Sample Run 3

Please enter the input sentences: **gulsen baris berker ali. . . @**
There should be no words without alphabetical characters.

Please enter the input sentences: ***radar giray. refer abba abcba a bb. cs201sc is not hard. A9a#L132aA AB BA CC D. @***

Sentence (4/4):
d cc notpalindrome notpalindrome aalaa
Sentence (3/4):
notpalindrome notpalindrome notpalindrome cssc
Sentence (2/4):
bb a abcba abba refer
Sentence (1/4):
notpalindrome radar

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all take-home exams, unless otherwise noted.

How to get help?

You can use GradeChecker (http://learnt.sabanciuniv.edu/GradeChecker/) to check your expected grade. Just a reminder, you will see a character ¶ which refers to a newline in your expected output.

You may ask questions to TAs (Teaching Assistants) or LAs (Learning Assistants) of CS201. Office hours of TAs/LAs are at the course website.

What and Where to Submit

You should prepare (or at least test) your program using MS Visual Studio 2012 C++ (Windows users) or using Xcode (macOS users).

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). If your name and last name is "İnanç Arın", and if you want to write it as comment; then you must type it as follows:

// Baris Altop

Submission guidelines are below. Since the grading process will be automatic, students are expected to strictly follow these guidelines. If you do not follow these guidelines, your grade will be 0.

- Name your submission file as follows:
    - 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_THEnumber.cpp"

    - Your SUCourse user name is actually your SUNet username, which is used for checking sabanciuniv emails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name (**use only lowercase letters**). For example, if your SUCourse username is "**altop**", then the file name should be: **altop_THE3*.cpp*** (please only use lowercase letters).
    - Do not add any other character or phrase to the file name.
- Please make sure that this file is the latest version of your take-home exam program.
- Submit your work **through SUCourse only**! You can use GradeChecker only to see if your program can produce the correct outputs both in the correct order and in the correct format. It will not be considered as the official submission. You must submit your work to SUCourse. You will receive no credits if you submit by any other means (email, paper, etc.).
- If you would like to resubmit your work, you should first remove the existing file(s). This step is very important. If you do not delete the old file(s), we will receive both files and the old one may be graded.

Grading, Review and Objections

Be careful about the automatic grading: Your programs will be graded using an automated system. Therefore, you should follow the guidelines on the input and output order. Moreover, you should also use the same text as given in the "Sample Runs" section. Otherwise, the automated grading process will fail for your take-home exam, and you may get a zero, or in the best scenario, you will lose points.

Grading:

- There is NO late submission. You need to submit your take-home exam before the deadline. Please be careful that SUCourse time and your computer time may have 1-2 minutes differences. You need to take this time difference into consideration.
- Successful submission is one of the requirements of the take-home exam. If, for some reason, you cannot successfully submit your take-home exam and we cannot grade it, your grade will be 0.
- If your code does not work because of a syntax error, then we cannot grade it; and thus, your grade will be 0.
- Please submit your **own** work only. It is really easy to find "similar" programs!
- Plagiarism will not be tolerated. Please check our plagiarism policy given in the Syllabus or on the course website.

Plagiarism will not be tolerated!

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 take-home exam from the email address provided in the comment section of your announced take-home exam grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the take-home exam tab to see the problem with your take-home exam.
- Download the 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!
Berker Demirel & Gülşen Demiröz & Barış Altop