1 Assignment 4 Introduction

This assignment is due *before* 11:59pm on the listed date. Which means, submissions made *on or after* 11:59pm will be counted as a late submissions. Late submissions *before* 11:59pm on the following day will receive a 25% point deduction as penalty. Submissions made *on or after* 11:59pm on the same day will *not* be graded. We strongly recommend completing the assignment before then.

It is imperative that you meticulously follow the submission process outlined at the end of this assignment. Incorrectly structured submissions will receive a 10% point deduction as a penalty.

Assignments due on Mondays generally involve materials covered in lectures from the previous week, whereas assignments due on Thursdays involve materials covered in lectures from the running week. So be sure to watch the lectures and go over the reading materials before attempting the assignments.

Good luck!

2 Word Scrambler (50 points)

Write a program that generates obfuscated texts by jumbling the middle letters of words. You will do this by reading from standard input a sequence whitespace separated lowercase-only words. Your program should output these words (8 words per line separated by spaces) scrambled. A scrambled word is a word that has its first and last letters unchanged, but has its middle letters reversed. In your solution, you must use a `void` function named `ScrambleText`, which takes the whole text as a string as the function parameter, and outputs the scrambled text based on the problem specification.

Example Input:

```
typoglycemia a portmanteau of typo and hypoglycemia is a neologism for a purported
discovery about the cognitive processes involved in reading text the principle is
that readers can comprehend text despite spelling errors and misplaced letters in
the words it is an urban legend and internet meme that only appears to be correct
```

Expected Output:

```
timecylgopya a paetnamtrou of tpyo and himecylgopya is
a nsigoloem for a petroprud drevocsiy auobt the
cvitingoe pessecors ievlovnd in rnidaeg txet the plpicnire
is taht rredaes can cneherpmod txet dtipsee snillepg
erorrs and mecalpsid lrettes in the wdros it
is an uabrn lneged and ienretnt mmee taht
olny araepps to be ccerrot
```

Note: all words should be followed by a space, unless they are the 8th word on the line.

General Instructions:

- When writing your program, you may only use concepts you have learned in the course thus far.
- Your program should compile and run.
- The formatting must match exactly as shown in the example above, including whitespaces.

Hints and Suggestions:

- You may find it useful to break this problem into smaller problems. The function `ScrambleText` does two things: (1) scramble each word, and (2) output them as 8 words per line.
- You can use another function called `ScrambleWord` which simply takes a word as a string as the function parameter, and returns the scrambled word. You can test this function using the words in the given text. You will obviously need to call `ScrambleWord` from `ScrambleText` for each word.

3 ASCII Neighbors (50 points)

Write a program that reads from standard input a word, filters the word by only keeping adjacent characters that are equal or adjacent in the ASCII table, and outputs the filtered word. For example, the character `i` has an ASCII value of 105, and it should be included in the output only if it is next to an `h` (104), an `i` (105), or a `j` (106). In your solution, you must use a `void` function named `FilterNeighbors`, which takes a string as the function parameter, and changes it based on the problem specification.

Input:

```
ahidkkrtsraaav
```

Output:

```
hikktsraaa
```

The first `a` is not in the output because its only neighbor `h` is not adjacent to it in the ASCII table. The `h` and the `i` are adjacent in the table so they are in the output. The `k`'s are equal to each other, so they are also in the output. And so on.

If a word contains no letters that are adjacent in the word and in the table, then the output should be empty.

Input:

```
gbsyfbjlbsadfoebw
```

Output:

```

```

There are many characters in ASCII beyond just letters. Your program should work for all ASCII printable characters found in the second table here: `https://www.ascii-code.com`. Some more test cases are provided below:

Input:

```
23321
```

Output:

```
23321
```

Input:

```
./-$$!&%aAJ
```

Output:

```
./$$&%
```

Input:

```
{|}:<:>=k?
```

Output:

```
{|}>=
```

General Instructions:

- When writing your program, you may only use concepts you have learned in the course thus far.
- Your program should compile and run.
- The formatting must match exactly as shown in the example above.

4 Assignment 4 Submission Process

- Create a folder, name it `your_msu_id4`. For example, if your MSU email is `johndoe@msu.edu`, then you should name the folder `johndoe4`.
- For each programming task, create a sub-folder inside your `your_msu_id4` folder, and name it as the number that corresponds to the programming task number. For this assignment, there should be two sub-folders named '2' and '3'.
- Inside each sub-folder, put the `main.cpp` for the appropriate solution.
- Compress/Zip `your_msu_id4` folder and name it `your_msu_id4.zip`. For example, if the name of your folder is `johndoe4`, then you need to create a zip file named `johndoe4.zip`. Zip file guide: `https://copyrightservice.co.uk/reg/creating-zip-files`.
- Submit the zip file through D2L.