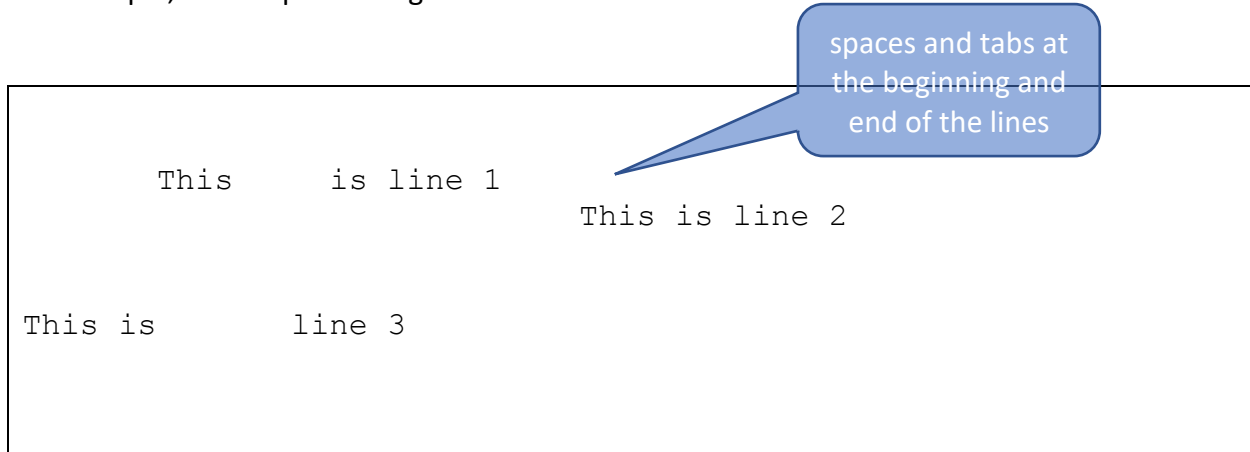


Objective:

Use input/output files, strings, and command line arguments.

Write a program that processes a text file by removing all blank lines (including lines that only contain white spaces), all spaces/tabs before the beginning of the line, and all spaces/tabs at the end of the line. The file must be saved under a different name with all the lines numbered and a single blank line added at the end of the file.

For example, if the input file is given as follows:



The diagram shows a text box representing an input file with the following content:

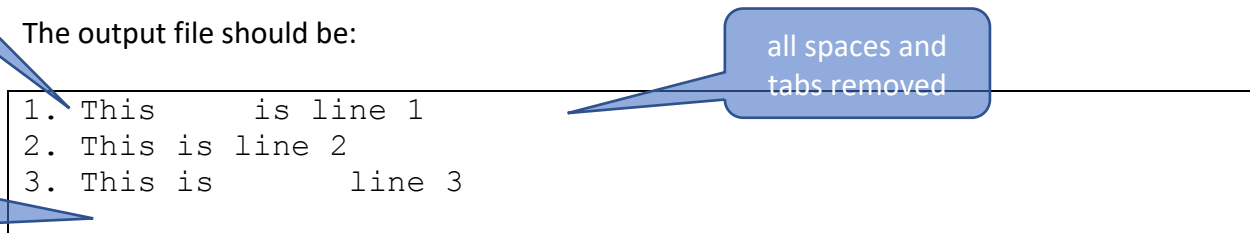
```
    This    is line 1
                This is line 2

This is      line 3
```

Annotations with callouts:

- A callout pointing to the leading spaces in "This is line 1" says "spaces and tabs at the beginning and end of the lines".
- A callout pointing to the single space between "This" and "is" in "This is line 3" says "single space".

The output file should be:



The diagram shows a text box representing an output file with the following content:

```
1. This    is line 1
2. This is line 2
3. This is      line 3
```

Annotations with callouts:

- A callout pointing to the trailing spaces in "This is line 1" says "all spaces and tabs removed".
- A callout pointing to the blank line between lines 2 and 3 says "blank line".

The file names must be provided at the command line (see "**Parsing command line arguments**" below). To use your program, a user would type:

```
./a.out first.txt second.txt
```

Where "first.txt" is the input file name and "second.txt" is the output file name.

Your program must print usage/help message and quit if any of the following occurs:

- Wrong number of arguments at the command line.
- The input file fails to open
- The output file fails to open

Your program must include, at least, the following functions:

- A function to display a usage message to the user.
- A function that removes all white spaces from the beginning and end of the line. Should take a string and return a string.

Parsing command line arguments

In C++ you can input data (in the form of strings) into your program on the command line (command line arguments). You can capture these data by adding two parameters to your main program as follows:

```
int main (int argc, char *argv[])
```

argc: (argument count) number of arguments on the line, including the name of the program

argv: (argument vector) list of c-style strings that represent all the arguments including the name of the program.

For example, executing the command:

```
./a.out file1.txt file2.txt
```

Assigns:

```
argc = 3  
argv[0] will be "./a.out"  
argv[1] will be "file1.txt"  
argv[2] will be "file2.txt"
```

To get the file name into your program you would write the following code:

```
string inputFileName = argv[1];  
string outputFileName = argv[2];
```

Grading:

Programs that contain syntax errors will earn zero points.

Programs that do not include the above functions will also earn zero points.

Programs that use global variables other than constants will earn zero points.

Your grade will be determined using the following criteria:

- Correctness (25 points)
 - 5 usage function.
 - 15 points: removing the white spaces function
 - 5 points: displaying error messages
- Style & Documentation (5 points)

Follow the coding style outline on GitHub:

<https://github.com/nasseef/cs2400/blob/master/docs/coding-style.md>