Overview:

- textReplace will be an application that allows a user to attach a text file, give input that specifies what strings they want to replace, and output a text file with all of the strings replaced with the specified ones.

Input:

- The name of a text file ("example.txt") that will contain the text to be operated on
- An integer *n* representing the number of replacement input entries
    - The next *n* lines will be in the form "i" "j" where *i* is the string to find and *j* is the string to replace it with. The quotes are necessary

Output:

- A new text file called "replaced_text.txt" will be created which will be the input text file but with all of the specified replacements made

Design - Console application:

- First prompts the user to enter the name of a text file and takes their input
- Prompts the user to enter the name of a file where the output will be saved
- Next prompts the user to enter the number of replacement specifications they will make and takes their input
- Lastly, takes the next *n* lines as the input for the replacement specifications
- Check for errors:
    - Check that the file exists
    - Check that file is the right format
    - Check that each line is written in the right format

Design - GUI:

- User can select a file from their computer or drag and drop it
- There is a row of 2 text boxes. The first one is where the user types the string they want to find. The second one is where the user types the string they want to replace it with
- Under the text boxes is a + button that allows the user to add more rows
- Each additional row can be deleted with the X button
- Check for errors:
    - Check that file is a text file
    - Check that no row is left blank

Application Specifications:

- All replacements will be done by reading the result of the last string being replaced

    - This means if the output of one replacement is the input of another replacement, the new output will still allow the replacement of any instances of the other replacement
    - For example, if the text reads “Hello, my name is Jesse.” and you have two replacements (“name” “nickname”) and (“nick” “nicolas”) in this order, the output would be “Hello, my nicolasname is Jesse.”
- For each substring of the document, only one replacement can be applied
    - For example, if the text document reads “Hello, my name is Jesse.” and you have two replacements (“my name” “your name”) and (“my” “his”) in this order, the output would be “Hello, your name is Jesse.”
- If one replacement input string is the substring of another replacement output string, the replacements will happen in the order they are entered
    - For example, if the text document reads “Hello, my name is Jesse.” and you have two replacements (“my name” “my middle name”) and (“my” “his”) in this order, the output would be “Hello, his middle name is Jesse.”

Classes:

- Replacement (struct)
    - Attributes:
        - string readString
        - string writeString
- Replacer
    - Attributes:
        - Replacement [] replacements
        - string infile
        - string outfile
        - string outputString
    - Methods:
        - void add_replacement(string text, string replace)
            - takes 2 strings and creates a Replacement object with those strings and adds the Replacement object into the Replacer’s replacements
        - void replace_text()
            - Begins the replacement operation

Replacement operation breakdown:

- The entirety of the infile is read and saved as the outputString
- For each replacement in replacements, we find and replace the desired strings and save them to a temporary string. At the end of each iteration, we replace outputString with whatever the temporary string is

- When the replacements are all done, we save the new text into a text file that will be named “replaced_text.txt”