Assignment 1
Deadline: 11:59AM Feb 5

Requirements
In this assignment, you are writing a translator. Let's assume that that is a native language, Bcitian, which is strictly used inside BCIT by those born and raised in BCIT. The problem is that BCITians use a variant of English, but because of their lack of societal engagement due to heavy coding workloads, their variant of English a bit different and is called Bcitian. Below are some noticeable differences:

English	Bcitian
I	Aei
er	erre
.	[.]
is	iizz
student	worker
professor	king
skytrain	sloth
class	ccamp
ing	eeng
friend	stranger
tion	ertioner

Your job is to produce a modern version of word processor with translator functionality, which takes in 2 inputs: input file, output file. Now, let's see your code in action. Let's first see a sample input file content:

>>cat intput.txt
I am a student who takes skytrain everyday to class, but my professor does not like it.
He does not like me taking it with my friend because his personality is not friendly.

Note that the input has two lines. After going through the translator, your sentences will read the following:
Aei am a worker who takes sloth everreyday to ccamp, but my king does not like it[.]
He does not like me takeeng it with my stranger because hiizz perresonality iizz not strangerly[.]

Also, there is one more requirement. Every console at BCIT can only print up to 25 characters and the above content must be justified. Below is an example of the justification with a console that can print 10 characters.

>>cat input.txt
I am a student at BCIT CST.

After justification with 10 characters, but for simplicity purposes, I am showing the output without translation here.
>>cat input.txt

I am a
student at
BCIT CST

As you can see, extra spaces are added in between words to make a 10-character line. If there are spaces before or after the line (e.g. I am a_ where _ is a space, it is removed to justify). If there can fit only a single word, then it is left justified as shown. For this assignment, you must implement 25 character length console. Also, note that there will be a new line character as well. Below is an example of the input with a new line character, but for simplicity purposes, I am showing the output without translation here.

>>cat input.txt
I am
a student at
BCIT CST.

After justification with 10 characters
>>cat input.txt
I am
a student
at BCIT
CST

Lastly, a newline break is preserved even after the translation. Here is an example with a line size of 10 characters:
>>cat input.txt
I er CS.
a student

Notice there is a line break after CS. even though there is enough space left? Your justified output after translation should look like the following:
>>cat input.txt
I erre
CS[.]
a worker

In this assignment, you must check corner cases. Corner cases mean that something unexpected occurred in the program. It can actually be from a user input or from program behaviors. In this assignment, we will only assume that corner cases occur from a user input. One example of such behavior (which obviously is not going to be tested for grading) is a file containing non-ASCII characters. The corner cases must be handled with the following:

```
// assume that a corner case is detected here in this line
//replace XXX with an appropriate message. Make sure that the word ERROR are all capitalized.
printf("COMP2510ERROR: XXX");
exit(1);
```

When exit(1) is used, the program is exited immediately. If a program exit any other way such as "segmentation fault" without such message, it is considered that a program has crashed.

Now with these details, your job is first to translate the input, justify it for 25 character line, and output to a file. You now are ready to implement your Bcitian translator.

Detailed specifications are below.

1. The command line argument is in this order <executable> input_file output_file
2. You will be guaranteed that the input file will contain ASCII characters only
3. The output is always 25 characters per line
4. The word will never break. If a group of words cannot fit in a line, then move it to the next line. Do not break a word.
5. User corner case can be anything that is within user's control. User here means the person who is using your executable and providing the input/output file
6. (recommendation) This is going to be a lot of lines of code. I would recommend you to break the code down to modular functions.
7. You will work in a group of 2.

Restrictions

- You CANNOT use strtok function
- For any reason, if your code does not compile, it will result in 0.
- Only use standard libraries listed in Learning Hub
- Adhere to lab/assignment submission guidelines in Edstem announcement.

Grading

Any grading failure due to not following specifications will result in 0. For full marks this week, you must:

- (1 point) Correctly use git/GitHub and the repository following the lab handout including the correct A number format. In the github, there must not be any other files other than submission files listed below. Failure to do so will result in losing marks.
- (6 points) Generate a correct solution to the problem(s)
- (4 points) Gracefully handling corner cases
- (1 point) Participation in Edstem discussion

Submission Files

- You must deliver only one .c file named: **assignment1.c**
- The file that you submit should be a .c file (not .txt, not .cpp or any other type)
- AXXXX.txt (empty file, but with your A number as file name. Make sure to include 0's and match this A number with your A number in learning hub)
- AXXXX.txt (same thing, but your partner's)
- In Github, there must be only assignment.c and 2 AXXX.txt files. Nothing else, so make sure to remove all other files there.
- Github: https://classroom.github.com/a/cpjzE-As