Lab 6 – Stacks and Queues

Ember Roberts, Maggie Lyon, and Nyla Spencer

The objectives/concepts explored in this lab are stack and queue data structures. The specific task involves reversing the letter within each word while maintaining the original order of the words. Stacks and queues are fundamental data structures and understanding their composure builds skills for developing more complex data structures. Learning to use them in conjunction with one another also builds necessary skills.

Contributions:

Maggie – Task 1, Task 3, debugging

Ember – Task 2, Task 3

Nyla – Task 2, lab report

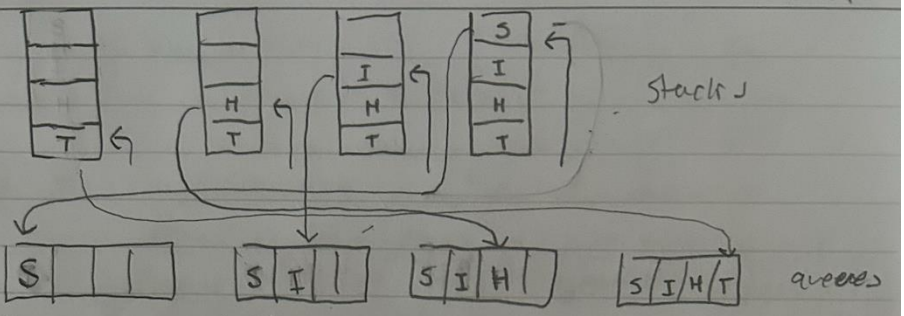
In task one, we created a template stack class utilizing an array. This stack has several functions including push, pop, top, length, empty. Exceptions such as overflow or underflow throw a custom class error.

In task two, A template queue class utilizing an array was created. This included standard queue functions (enqueue, dequeue, peek, is Empty, size).

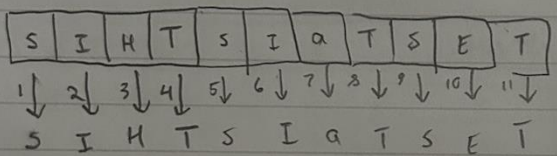
Finally, in task three, we started working with the text. Our program reads a series of strings from the above file, reverses the order of the text in each word while keeping the words in the same order, then prints the result to the screen. This includes unique error messages when over/underflow and options to end/continue running. Shown below is a screenshot of the output of our code, as well as a diagram of our programs functionality using both stacks and queues. First, the input string is separated into individual words. Each word is added to stack and then removed in last in first out order, reversing the word. Then, this word is added to a queue. The next word is added and removed from the stack and then added to the queue, until all words are in the queue. At this point, the entire queue is dequeued to form a string of reversed words while still maintaining the order of the original words.

```
Choose an option:
1. File name:
2. User input:
3. Exit
1
Enter the file name:
example.txt
sdrowkcaB txeT
Would you like to continue? (y/n)
y
Choose an option:
1. File name:
2. User input:
3. Exit
2
Enter a string:
I love Kitty Cats
I evol yttiK staC
Would you like to continue? (y/n)
n
```

step 1 - letters in individual words are added to stack



step 2 - letters are removed from stacks LIFO and added to queue



letters are removed FIFO from queue to get reversed words