Question #1:

File to run: *Q1_linear-binary_search.py*

How to run the code:

- Code has 5 functions (***linear_search***, ***merge***, ***merge_sort***, ***binary_search***, and ***main***)
- In the main function I created a list of elements, ***S***. As well as a list of target values, ***target_values***. There are 2 variables declared ***n*** and ***k***, ***n*** being the size of the list ***S*** and ***k*** being the size of the list ***target_values***.
- The lists were populated using random numbers by importing random. To ensure only half of the elements in ***target_values*** were in list ***S***, the list elements were generated using ***random.randrange(2, 20, 2)*** and the value of the target elements were generated using ***random.randrange(1, 20)***
- In the main function each target value was searched for both using ***linear_search*** and ***binary_search***
- As binary searching requires the list to be in order, the binary search function also calls upon the ***merge_sort*** function
- The main function returns the time taken to complete the linear search and binary search as well as n and k to display test results in a user friendly fashion

Testing (finding smallest possible k value):

- I tested the code under 3 different cases with 3 different list lengths n = 1000, 5000, 10000
- The amount of target values (k) started at 10 and were increased by various increments for each test

Test Results:

	n = 1000	**n = 5000**	**n = 10000**
# of target values (k) = 10	Linear Time: 0.0007 Binary Time: 0.0079	Linear Time: 0.0025 Binary Time: 0.0239	Linear Time: 0.0029 Binary Time: 0.0399
# of target values (k) = 20	Linear Time: 0.0018 Binary Time: 0.0074	Linear Time: 0.0036 Binary Time: 0.0216	Linear Time: 0.0079 Binary Time: 0.0396
# of target values (k) = 30	Linear Time: 0.0021 Binary Time: 0.0081	Linear Time: 0.006 Binary Time: 0.0216	Linear Time: 0.0118 Binary Time: 0.0399
# of target values (k) = 50	Linear Time: 0.0034 Binary Time: 0.0071	Linear Time: 0.008 Binary Time: 0.0211	Linear Time: 0.0175 Binary Time: 0.0369
# of target values (k) = 100	Linear Time: 0.0052 Binary Time: 0.0069	Linear Time: 0.0188 Binary Time: 0.0181	Linear Time: 0.029 Binary Time: 0.0369
# of target values (k) = 125	Linear Time: 0.0074 Binary Time: 0.0075		Linear Time: 0.0341 Binary Time: 0.0342
# of target values (k) = 130	Linear Time: 0.0087 Binary Time: 0.0066		Linear Time: 0.0349 Binary Time: 0.037
# of target values (k) = 135			Linear Time: 0.0374 Binary Time: 0.0371

Per the testing process I came to the conclusion that the smallest value of k that makes binary search faster than linear search is dependent on the size of the list but can be approximate to 100-135 items.

Question #2:

Files to run:

1. *Q2_array-based_stack.py*
2. *Q2_linkedlist_stack.py*

File #1: *Q2_array-based_stack.py*

- This file contains a ***class Stack*** to build stack for array based lists
- Class contains the methods ***size***, ***is_empty***, ***push***, ***pop***, and ***top***
- Main function creates a test stack and calls all of the methods on the stack to ensure proper functionality
  - Note to test functionality, can call methods on ***test_stack***

File #2: *Q2_linkedlist_stack.py*

- This file contains a ***class Node*** to create linked lists
- This file contains a ***class Stack*** to build stacks for linked lists
- Class contains the methods ***size***, ***is_empty***, ***push***, ***pop***, and ***top***
- Main function creates a test stack and calls all of the methods on the stack to ensure proper functionality
  - Note to test functionality, can call methods on ***test_stack***

Question #3:

File to run: *Q3_circular_queue.py*

How to run the code:

- This file contains a ***class CircularQueue*** to build a circular queue (Note requires parameter x, which represents the maximum size of the queue - x must be an integer)
- Class contains the methods ***is_empty***, ***is_full***, ***enqueue***, and ***dequeue***
- Main function creates a test queue and call the ***enqueue*** and ***dequeue*** methods on the queue to ensure proper functionality
  - Note to test functionality, can call methods on ***test_queue***