DUKE MIRIAM

DATA STRUCTURES

ASSIGNMENT 2

1. Analysis of Deque simulation:
   Description:
   The deque is implemented using a circular array structure. This implies the the front of the queue may not necessarily be the first element seen in the output of the array as a linear list. In a circular array the back keeps reference of the front. The index of the front continually changes as elements are being removed and added to the deque.

   The simulation is run fifty times for each row of probability and the data is analysed. The probability intervals determine whether additions or removals are to be done. After it is run the average probability of addition and removals is found. After analyzing it was realizes that the average probability was roughly the same as given in the question.

   If the code is run using the main1 function that was coded in line with the first row of probabilities, it is observed that there are more removals from the back of the queue, thus leaving the queue empty. If the code is run using the main2 function that was coded in line with the second row of probabilities, it is observed that there are more removals from the front of the queue, thus leaving the queue empty.

To run the deque:

1. Locate the file in github saved as 'Deque.py'
2. Run the code
3. In the shell, Create an object of the Deck class e.g new_Deque= Deck()
4. Call the main1 or main2 method to run operations on the new deque object.
   e.g new_Deque.main1() or new_Deque.main2()

5. The result would be displayed as appropriate

2. Binary search vs Interpolation search data:

TABLE 1

	BINARY SEARCH RUNTIME					
N(Array size)	**Search 1**	**Search 2**	**Search 3**	**Search 4**	**Search 5**	**AVERAGE**
100	1.19e-05	1.5799999999899228e-05	1.3199999997937084e-05	1.3100000001884756e-05	7.799999991675577e-06	0.000012
1000	1.499999999055035e-05	1.33e-05	1.639999999980546e-05	1.4199999995412327e-05	1.559999999756201e-05	0.0000199
5000	1.7399999990175274e-05	1.589999999396241e-05	1.61e-05	2.0200000001580065e-05	2.320000000111122e-05	0.00001856

TABLE 2

	INTERPOLATION SEARCH RUNTIME					
N(Array size)	**Search 1**	**Search 2**	**Search 3**	**Search 4**	**Search 5**	**AVERAGE**
100	1.0900000000000007e-05	1.3799999997843315e-05	7.899999999949614e-06	7.50000000948603e-06	1.2900000001536682e-05	0.0000106
1000	1.5600000011772863e-05	7.8e-06	9.8999999949001e-06	6.099999993125493e-06	1.0200000005511356e-05	0.00000992
5000	7.600000003549212e-06	9.599999998499698e-06	2.3000000000000017e-06	1.639999999980546e-05	1.629999999863685e-05	0.00001044

TABLE 3

N(Array size)	AVERAGE RUN TIME FOR BINARY SEARCH	AVERAGE RUN TIME FOR INTERPOLATION SEARCH
100	0.000012	0.0000106
1000	0.0000199	0.00000992
5000	0.00001856	0.00001044

Analysis of run time values:

The code was run fifteen times 5 for each given size of the array(100,1000,5000). Out of the 15 runs, Interpolation search's runtime was less than that of Binary search 13 times. That is the probability that interpolation would produce the result faster was 0.87. This shows that Interpolation search is more efficient than Binary search. The average runtime for interpolation search is also less than that of binary search. Hence interpolation search is a quicker search technique.

To Run the Binary vs Interpolation search simulation:

1. Open github and locate the python file named Binary vs "Interpolation search"
2. Open code in any python IDE and run
3. The code would request for the array size N(e.g 100,1000,5000)
4. User should input array size as requested
5. The code will prompt user to eneter target element
6. The user should enter target element
7. The binary and interpolation search methods would both run for the given array and target value
8. The time taken is given as out put
9. The user is informed whether the item was found

To access my Github kindly use the link: https://github.com/Mimi-D/Data-Structures.git

1. You can download or clone the code to your computer.
2. Download by clicking the green "code" button: click on "Download Zip"
3. To clone click on "Open with Git Desktop" if git is already installed on your machine
4. Run with any python IDE of your choice