Name: Kaan Kaya
Student number: 218189415

A. Array
(a) Slow, because it has to go through every element.
(b) Fast, because size of an array is already decided so n/2 would take us to the middle element immediately.
(c) Fast, because an array's first element is always at array[0].
(d) Slow, because list's end element is not known, we have to go through all the array to find the last element.
(e) Fast, because an array's first element is always at array[0].
(f) Slow, because list's end element is not known, we have to go through all the array to find the last element.

B. Singly-linked list, with "next" only, and with "head" only.
(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have "head" element which is the beginning of the list.
(d) Slow, because we don't have the "tail" element of the list, so we have to go through all the list to reach the end element.
(e) Fast, because we already have "head" element which is the beginning of the list.
(f) Slow, because we don't have the "tail" element of the list, so we have to go through all the list to reach the end element.

C. Singly-linked list, with "next" only, and with "head" and "tail".
(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have "head" element which is the beginning of the list.
(d) Fast, because we already have "tail" element which is the end of the list.
(e) Fast, because we already have "head" element which is the beginning of the list.
(f) Fast, because we already have "tail" element which is the end of the list.

D. Reverse singly-linked list, with "prev" only, and with "tail" only.
(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have "tail" element which is the beginning of the reverse linked-list.
(d) Slow, because we don't have the "head" element of the list, so we have to go through all the list to reach the end element.
(e) Fast, because we already have "tail" element which is the beginning of the reverse linked-list.
(f) Slow, because we don't have the "head" element of the list, so we have to go through all the list to reach the end element.

E. Doubly-linked list with "next" and "prev", and with "head" and "tail".
(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have "head" element which is the beginning of the list.
(d) Fast, because we already have "tail" element which is the end of the list.
(e) Fast, because we already have "head" element which is the beginning of the list.

(f) Fast, because we already have “tail” element which is the end of the list.

F. Circular singly-linked list, with “next” only, and with “tail” only.

(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have “tail” element which is the end of the list. Beginning of the list is always one next element of “tail” in circular linked lists.
(d) Fast, because we already have “tail” element which is the end of the list.
(e) Fast, because we already have “tail” element which is the end of the list. Beginning of the list is always one next element of “tail” in circular linked lists.
(f) Fast, because we already have “tail” element which is the end of the list.

G. Circular doubly-linked list, with “next” and “prev”, and with “tail” only.

(a) Slow, because it has to go through every element.
(b) Slow, because it has to go through every element to find size, so it can go to the middle element.
(c) Fast, because we already have “tail” element which is the end of the list. Beginning of the list is always one next element of “tail” in circular linked lists.
(d) Fast, because we already have “tail” element which is the end of the list.
(e) Fast, because we already have “tail” element which is the end of the list. Beginning of the list is always one next element of “tail” in circular linked lists.
(f) Fast, because we already have “tail” element which is the end of the list.