**Homework 01 -- Questions 1 - 5**

1. What is a dynamic array? How does it work?

A dynamic array is a data structure which allows the programmer to store data in an array with alterable size. The dynamic array includes an array as instance data as well as variables that keep track of the size of the array (the actual length of the array) and the count of the array (how many elements are currently being stored in the array). When the program attempts to add an element to the array, it checks if the array is already full (that is, count == size). If it is not full, the program simply appends the element to the array. If it is full, the program calls another method of the dynamic array which grows the size of the array. To do this, it creates a new array with double the size of the old array and manually copies the data from the old array to the new array. This functionality allows the array to expand in size when necessary, although it is a costly operation.

2. How is an Array different from a Linked List? Discuss in terms of the runtime and space for different operations.

The largest difference between arrays and linked lists comes from how the data is arranged and accessed in memory. With an array, the data is indexed, and each element of the array is stored consecutively in memory. With a linked list, however, the data is stored in nodes, each of which has a reference to the next node (and optionally a reference to the previous node in doubly-linked lists). These nodes do not need to appear consecutively in memory.

These differences create disparities in the efficiencies of arrays and linked lists for performing various operations. For example, the array is much more efficient at accessing elements than the linked list; accessing an element of an array simply requires the index of that element, whereas the linked list must be traversed one node at a time until the desired node is found, requiring a much longer runtime. Inserting and deleting nodes, however, is relatively cheap for the linked list, as it only requires updating the links of the list to either skip a node (in which case the node has no reference in the code its memory will be recycled) or include a new node. Inserting and deleting elements at the end of an array is similarly cheap, but inserting or deleting and the beginning or middle of an array requires the program to shift every element after the inserted or deleted element, which is very expensive.

The linked list does require more memory space than the array for storing data, since the list must keep track of each node’s instance data as well as the reference to the next node. Despite this, adding elements to a linked list is relatively cheap in terms of memory space, whereas adding a new element to a dynamic array when the array is full requires creating an entirely new array in memory. The data of the original array has to be copied to the new array, which also makes this operation very costly in terms of runtime compared to adding a new node to a linked list.

3. Explain the scenarios where you can use linked lists and arrays.

Linked lists and arrays are useful for any situation where the program needs to store a set of similar data in a particular order. Linked lists are used for playlists of music or videos, for example. Arrays are particularly useful for storing ordered data since array elements are accessed by index. Arrays can also be multi-dimensional, which makes them useful for representing tables of data in a logical way. Since it is much cheaper to insert and delete elements from a linked list than it is for arrays, linked lists are more useful for data sets that require frequent insertion and deletion.

4. What is a Queue, and how is it different from a Stack? How can the Stack and Queue be implemented?

5. Which data structures are used for implementing the LRU cache?

**Programming Problems (6 – 10)**

Question 6:

Discussion:

Verification:

Question 7:

Discussion:

Verification:

Question 8:

Discussion:

Verification:

Question 9:

Discussion:

Verification:

Question 10:

Discussion:

Verification: