Discussion Prompt Questions
Module 4
Brian Loughran
Johns Hopkins Data Structures

1. A lot of things have a LIFO characteristic. What is something that does not have a LIFO characteristic that you could still effectively manage with a queue?
2. What are some of the key distinguishing features of a list? Under what conditions would these be useful?
3. Should a list ADT include hints or references to the list implementation?
4. What might you consider to be a nice-to-have in a list ADT (not necessary, but potentially useful)?
5. When might the implementation details of a list be an important consideration?
6. List some pros and cons of an array implementation of a list.
7. List some pros and cons of a linked implementation of a list.

ANSWERS:

1. Queues are more useful in dealing with things with FIFO characteristics. One classic example is the milk at the grocery store. The first milk is put in and slides to the front of the fridge. If more milk arrives at the grocery store, it is put in the back of the slider. Customers can only select from the front of the slider, which is the back of the queue. Therefore, the first milk carton that is put in the slider is the first milk carton that is bought .This is done to prevent milk hiding in the back of the slider and going bad. This is a good, real life system which has a FIFO characteristic (not LIFO) which can be effectively tracked using a queue.
2. A list is an ordered collection of data, where all locations are available for insertion or deletion. There is no limit on the number or nature of items. A list may be useful in tracking students in a class, where some students may drop halfway through, or transfer from another class. Another useful implementation would be for keeping track of groceries. Whenever you need an item, you can put it in the list, and once it is in your cart, you can take it off the list.
3. An ADT should not include hints and references to the list implementation. It would be odd if you could tell if one was using the linked list or array implementation for a list because an ADT should only have information about input, precondition, postcondition, process and output. The ADT should only have information about the functionality, the implementation should be entirely separate.
4. Some functions that would be nice to have may include sort() (which would have to be clear if that is sortNumerical(), sortAlphabetical() or some other type of sort). Perhaps if you had a sorted list and you wanted to find a specific value, you could use a binarySearch() function. This would help to search the list in O(lgn) time.
5. Implementation details of a list may be important when you are discussing which type of list you want to have. Will it be a long list that you have to find an index in quite often? You may want to use an array to take advantage of advanced search algorithms. Are you worried about memory? You may want a linked list implementation to take advantage of the efficiency of dynamically allocating memory. Once you have an ADT and the functionality is laid out, you should consider the different use cases your list may face. Then you should start to think about the details of the implementation for you list.
6. For the array implementation, the size of the list is static. This is a disadvantage, and can lead to stack overflow issues with the list. Items are required to be homogeneus, so you cannot have an array implementation that includes, say, strings and ints. One advantage to array implementation is that you can access indices randomly, so you can potentially use more advanced searching algorithms to reduce your search time.
7. For the linked list implementation of a list, there are no size limits. You can have pointers point to other values of a list to infinity (if you have that much memory). A disadvantage to this is that you must access the list sequentially. Each index of the list only has one pointer, so you must go through each pointer to access the next index of the list. Advanced search algorithms like binarySearch(), therefore, are not well suited to the linked list implementation of the list.