Study Guide:

You are responsible for all reading, all worksheets, and all homework assignments posted on the course webpage. The worksheets and homework are an excellent study source. Make sure that you understand the solutions that we have sent you via the class e-mail. If you lost some points on homework, check where and why you made the mistake. To prepare well for the midterm exam, make sure you are very comfortable with all data structures, and all implementations of the data structures that we have discussed. For example, it would not be sufficient to only know how to implement a deque using the dynamic array. You also need to know how to implement a deque using singly-, doubly, and circularly-linked lists. You should be able to reimplement any of them, as well as a variation on any of them. You should be familiar with the big-O complexity of all the functions from the worksheets and homeworks. You should also know how to compare various implementations that we have discussed in terms of the big-O complexity. You will be required to write some code. This will be similar to the worksheets and homework, i.e., we will provide some functions that you would need to complete. Below, we list a few example problems for your practice.

1. What is the big – O complexity for:
   a. A method that takes exactly $2n^2 + 5n + 100$ steps
   b. for (i = n; i > 0; i = i/2) {
       //constant time operations
       ...
    }
   c. for (i = n; i > 0; i = i/2) {
       for (j = i; j > 0; j = j/2) {
           //constant time operations
           ...
        }
    }
   d. for (current = previous->next; current != 0; current = current->next) {
       for (previous = current->next; previous != 0; previous = previous->next) {
           //constant time operations
           ...
        }
    }
2. Which data structures require sequential access? Which data structures require random access? Explain your reasoning for each data structure that you mention.

3. Name the important property and interface operations of the following ADTs:
   a. Stack
   b. Bag
   c. Queue
   d. Deque

4. Phil Parker runs a parking lot where cars are stored in six lines holding at most three cars each (see figure below). Patrons leave the keys in their cars so that they can be moved, if necessary. Assuming that no other parking space is available and that Phil cannot predict the time at which patrons will return for their cars, how many spaces must he leave empty to ensure that he can reach any car?
   a. 0
   b. 1
   c. 2
   d. 3
   e. none of the above

5. Suppose you are to develop a software which will help Phil Parker from Problem 4 to park and unpark cars in the aforementioned parking lot. What data structure will be convenient to keep the record of the cars? Is it possible to code the parking lot with only one instance of that data structure, or you would need a number of instances of that data structure? How would you implement that data structure in C by using arrays, dynamic arrays, singly-linked lists, or doubly-linked lists?

6. How does the function _setCapacity change for the dynamic array implementation of a deque, as compared to the dynamic array implementation of a stack?

7. If you were developing an application that heavily involved searching for particular elements in a bag, which of the following implementations of the bag would be most appropriate?
   a. Singly-linked list
   b. Doubly-linked list
   c. Dynamic array
8. Draw the dynamic array structure after the following commands are executed. Show the size, capacity, and contents. You can assume that a resize doubles the capacity. (5 pts)

```c
Struct dynArr s;
initDynArr(&s, 4);
pushDynArr(&s, 6);
pushDynArr(&s, 3);
popDynArr(&s);
pushDynArr(&s, 10);
pushDynArr(&s, 15);
pushDynArr(&s, 2);
pushDynArr(&s, 20);
popDynArr(&s);
```

9. Write the indexing function `TYPE getValueAt()` for the doubly-linked list implementation of a deque that takes an integer index, and returns the value stored at that index assuming index 0 is the link after the front sentinel. What is the big-O execution time of the function that you provided?

```c
struct dlink {
    TYPE value;
    struct dlink * next;
    struct dlink * prev;
};

struct listDeque {
    int size;
    struct dlink *frontSentinel;
    struct dlink *backSentinel;
};

TYPE getValueAt(struct listDeque *q, int idx){
    // FIX ME
}
```