Homework 8
CS 3385

1. For a stack, if we push $1, 2, 3, 4, 5$ and then pop five times, give the order of the elements popped off the stack.

2. For a queue, if we enqueue $1, 2, 3, 4, 5$ and then dequeue five times, give the order of the elements dequeued from the queue.

3. Using figure 10.1 as a model, illustrate the result of each operation in the sequence `push(4), push(1), push(3), pop(), push(8), pop()` on an initially empty stack stored in an array $S[1..6]$.

4. Using figure 10.2 as a model, illustrate the result of each operation in the sequence `enqueue(4), enqueue(1), enqueue(3), dequeue(), enqueue(8), dequeue()` on an initially empty queue stored in an array $Q[1..6]$.

5. Recall that a set has no duplicate elements. Suppose we use a linked list to store a set. The operation `union` takes two sets and combines them into a single set.

   (a) Describe the running time of `union` if we know the two sets are disjoint (they don't share any elements).

   (b) Describe the running time of `union` if we *don't* know that the two sets are disjoint.

6. In a doubly-linked list, each node has a pointer to both the next node and the previous node (see figure 10.3 in the textbook). For each of the four types of lists in the following table, what is the asymptotic worst-case running time for each operation listed? $L$ is the list, $k$ is a key, and $x$ is a node. `Successor(L,x)` finds the next ordered node and `Predecessor(L,x)` finds the previous ordered node. Assume in all cases that $x$ is a node with pointer(s). Assume that sorted means sorted in ascending order.

| | unsorted, singly linked | sorted, singly linked | unsorted, doubly linked | sorted, doubly linked |
|---|---|---|---|---|
| `Search(L,k)` | | | | |
| `Insert(L,x)` | | | | |
| `Delete(L,x)` | | | | |
| `Successor(L,x)` | | | | |
| `Predecessor(L,x)` | | | | |
| `Minimum(L)` | | | | |
| `Maximum(L)` | | | | |

7. In a direct-address table or hash table, "satellite data" is extra data that is stored along with the key. For example, in the video, "Joe" and "Jill" would be satellite data, where 385 and 741 would be the keys.

   A bit vector is an array of bits (0s and 1s). A bit vector of length $m$ takes much less space than an array of $m$ pointers (which so far has been what we use to implement a hash table). Describe how to use a bit vector to represent a dynamic set of elements with no satellite data which each have distinct hash values. Operations `Insert`, `Delete`, and `Search` should run in O(1) time.

8. Show the resulting chained hash table after inserting the keys $5, 28, 19, 15, 20, 33, 12, 17, 10$. Let the table have 9 slots and let the hash function be $h(k) = k \bmod 9$. Elements in each chain can be given in any order.

9. Professor Plum hypothesizes that he can obtain substantial performance gains by modifying the chaining scheme to keep each list in sorted order. How does the professor's modification affect the running time for successful searches, unsuccessful searches, insertions, and deletions (which take the node to be deleted, not the key)?

10. Suppose we wish to search a linked list of length n for a string k. Say that each element in the list contains the string k along with a hash value for the string $h(k)$. If we know that the string at each element is a very long character string, how might we use the hash values to speed up searching the list for a particular string k?

11. Consider a hash table of size $m = 1000$ and a corresponding hash function $h(k) = \lfloor m(kA \bmod 1) \rfloor$ for $A = (\sqrt{5} - 1)/2$. Compute the locations to which the keys $61, 62, 63, 64$, and $65$ are mapped.

12. Consider inserting the keys $10, 22, 31, 4, 15, 28, 17, 59$ into a hash table of length $m = 11$ using open addressing with the auxiliary hash function $h'(k) = k$. Show the hash table after inserting the keys using:

    (a) linear probing.
    (b) quadratic probing with $c_1 = 1$ and $c_2 = 3$.
    (c) double hashing with $h_1(k) = k$ and $h_2(k) = 1 + (k \bmod (m - 1))$.

    Show also the number of collisions for each key and the total number of collisions. For example, *a portion* of the answer for part (a) is the following:

```
0  1  2  3  4  5  6  7  8  9  10  index
               15       59    10
               1        4     0
6 total collisions
```