2023S Datastruct Midterm A Name: _____

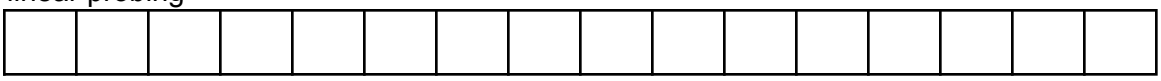
1. (5 points) Given the data here, show the passes of insertion sort
    a. How many passes does it take to sort n elements?
    b. What is the worst-case complexity of insertion sort?
    c. What is the best-case complexity of insertion sort?
    d. What is the complexity of insertion sort if all elements are sorted except for one element?
    e. What is the complexity of insertion sort

pass	3	2	1	5	6	4	8	7
1								
2								
3								
4								
5								

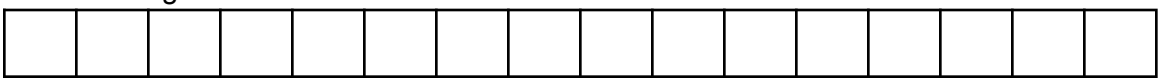
2. (5 points) Given the data here, show how the data is partitioned for the first pass of quicksort using a random pivot (you may assume the random number is 3, selecting the number 5).
    a. What is the worst-case complexity of quicksort?
    b. What is the best-case complexity of insertion sort?
    c. What is the complexity of quicksort using the pivot choice p = x[LEFT], the first element in each partition, given that the data is sorted?
    d. What is the complexity of quicksort using the pivot choice p = x[RIGHT], the last element in each partition, given that the data is sorted?

pass	3	2	1	5	6	4	8	7

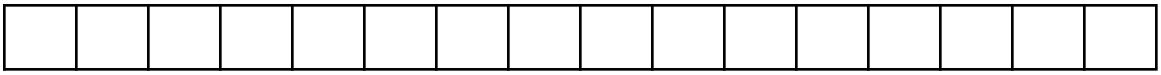
3. (10 points) Show results of the following hash algorithms:
   Given the hash algorithm hash(x) = x mod n
   insert (51, 48, 17, 16, 15, 31)
   n=16, linear probing



n=16, linear chaining



n=16, perfect hashing, use the same function for both levels



4. (10 points) Write pseudocode for binary search using iteration (not recursion).
   Return the position of the target within the list x or -1 if not found.
   binarySearch(x[], int target)

5. (30 points) Write a singly linked list class with a head and tail. ***Implement all the methods*** of the following main. Code may be in C++, Java or pseudocode but express the algorithm clearly.
   Write the complexity of each method.
   Determine the complexity of each line of the code below for your implementation.

```
int main() {
  LinkedList a;
  for (int i = 0; i < n; i++) {      //O(_____)
    a.addFirst(i);                   //O(_____)
    a.addLast(i);                    //O(_____)
  }
  for (int i = 0; i < a.getSize(); i++)    // O(_____)
    if (a.get(i) % 2 == 0)                 // O(_____)
      a.remove(i);                         // O(_____)
}
```

6. (20 points) Write a class HashMapLinearProbing. You may assume that a method hash() exists that returns a hashed integer value from 0 to maxint, but you must restrict the resulting number to the size of the table. Each node should contain a key and corresponding value.
   For every key in the table, you must also have a value. This is usually done by defining a Node that contains both:

```
class Node {
  String key;
  int value;
}
```

Write ***3 methods and state complexity***:

void insert(key, val)	inserts a key,value pair, (maybe new, or may replace existing key)
int get(key)	return value corresponding to key ***if the value is not found, return null***
void printHistogram()	display how many times add was able to work with 0 collisions, 1 collision, and so on. The last bin in the histogram is for all adds in which 9 or more collisions occurred.

For example (you do not have to implement a templated class, pseudocode is fine as usual):

```
HashMap<string, int> quote;
quote.add("AAPL", 206);          //first inserts the symbol AAPL into hashmap with val 206
quote.add("AAPL", 214);          // replaces the entry with val 214, does not create new one
int quote.get("AAPL");           // in C++, this code cannot handle returning null
                                 // you do not have to worry about that
                                 // just pretend it would work or write real code if you wish
quote.printHistogram();
```

7. (20 points) Write pseudocode for the insert and containsWord methods of the trie shown, and state complexity. C++, Java, and pseudocode are all acceptable.

```
class trie {
  class node {
  public:
    node* next[26];
    bool isword;
  };
  node root;
public:
  void insert(const string& word)




  bool containsWord(const string& word)
```

9. (2 points extra credit) What is the space complexity of a trie in terms of n, the number of symbols, k the size of the alphabet, and r the maximum size of a word? If you need more variables, define them and explain your answer.

10. (1 points extra credit) What is the amortized complexity of the following code?

```
scramble(x[], int n, y[]) {
  out ← 0
  for i ← 0 to length(x)
    do
      pick ← random(0,n-1)          // pick a random element in x
    while x[pick] < 0
    y[out] ← x[pick]                // place in the output
    x[pick] ← -1                    // replace by -1 so it is not found again
    out ←out + 1
  end
end
```

11. (2 points extra credit) Given an unsorted list of n elements, define an efficient algorithm to find the median element. Note that you have to be more efficient than sorting the list and picking the middle one, that's obvious.