COMP 282 - Midterm 2 (Fall, 2018) Name:

**Question 1** Provide a short answer to the following questions.

(a) What is hashing good for?

(b) Give an example of an inconsistent hashing function.

(c) Briefly describe the Pigeon Hole Principle.

(d) Describe a situation in which you would prefer to use open addressing over collision chaining.

(e) Give an example of a trivial hashing function.

**Question 2** Given the following set of 2-tuples – where the first element is taken to be the key, and the second is to be the value – provide a hashing function that is minimally perfect. Give the hash table that results from your function.

```
(99801, "Alaska")
(91423, "California")
(83251, "Idaho")
(07801, "New Jersey")
(29418, "South Carolina")
```

**Question 3** Is the following function a "good" candidate for a hashing function? Why or why not?

```
public class Hasher {
  private static int prev = 0;

  public static int hash (int input, int limit) {
    int hash = 0;

    while (int(input /= 2) > 0) {
      hash += input % 2;
    }

    prev = hash + prev;
    return prev % limit;
  }
}
```

**Question 4** Imagine a computer with the following performance characteristics:

| | |
|---|---|
| Time to Load Page into Memory: | 100 ms |
| Time for In-Memory Operations on a Single Value: | 1 ms |
| Capacity of a Single Page: | 100 bytes |
| Size of an Integer: | 4 bytes |

Answer the following questions pertaining to this computer.

(a) What would be an appropriate way to store 10 integers in a file so they were quickly searchable?

(b) How long would it take to perform a binary search of 1,000,000 (sorted) integers?

(c) Suggest a better strategy. How long does your strategy take to search 1,000,000 integers?

**Question 5** Propose the most efficient data structure for the following scenarios. Justify your answer.

(a) You want to be able to quickly locate a value in a collection whose elements do not change often. You have the exact amount of memory required to store these elements.

(b) You have a large amount of disk space, and want to quickly locate, insert, and delete items.

(c) You want to be able to quickly locate a value in a collection whose elements change often. You have limited memory.

(d) You have very limited memory, but a large amount of disk space. You want to be able to search a very large number of values quickly.