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

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

1. What is hashing good for?
2. Give an example of an inconsistent hashing function.
3. Briefly describe the Pigeon Hole Principle.
4. Describe a situation in which you would prefer to use open addressing over collision chaining.
5. 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.

1. What would be an appropriate way to store 10 integers in a file so they were quickly searchable?
2. How long would it take to perform a binary search of 1,000,000 (sorted) integers?
3. 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.

1. 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.
2. You have a large amount of disk space, and want to quickly locate, insert, and delete items.
3. You want to be able to quickly locate a value in a collection whose elements change often. You have limited memory.
4. 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.