1 Hashing

(a) Imagine we have the following class:

```java
public class Course {
    public final int CCN;
    public final String instructor;
    public Student[] students;
    public int audited; //when the course was last audited
    public Course(int CCN, Student[] initial) {
        this.CCN = CCN;
        this.students = initial;
        this.instructor = "Sohum";
    }
    //implementation
    public void audit() {
        this.audited = System.currentTimeMillis();
        //implementation
    }
    //implementation
    public void addStudent(Student s) {
        //implementation
    }
}
```

Which of the following hashing functions for the Course class are valid?

A) ```java
@override
public int hashCode() {
    return CCN;
}
```  

B) ```java
@override
public int hashCode() {
    return this.students.length;
}
```  

C) ```java
@override
public int hashCode() {
    return this.audited;
}
```
D)  
@override
public int hashCode() {
    return 5;
}

E)  
@override
public int hashCode() {
    return getNumericValue(this.instructor.charAt(0));
}

(b) We have the below external chaining HashSet.

If the load factor is 1.25, how many more insertions can we make before we will resize? Do not include the insertion that will begin with the resize.

() 1
() 2
() 3
() 4
() 5
() 6
(c) Suppose we have the following MonsterHashTable.

The number to the upper left of the monster is their hashCode.

**Part One:**
Suppose we want to insert the element:

Which of the following would correctly mirror the state after inserting the above element?

A)  

B)
Part Two:
Now, after inserting that element, we want to insert a new element (shown below) and resize.

Which of the following correctly mirrors the new state?
B)

C)

( ) A
( ) B
( ) C
2 Unexpected Hashing

Suppose we have the Lamp class below:

```java
class Lamp {
    int brightness;

    Lamp(int brightness) {
        this.brightness = brightness;
    }

    @Override
    public int hashCode() {
        return brightness;
    }

    @Override
    public boolean equals(Object o) {
        return ((Lamp) o).brightness == brightness;
    }
}
```

Assume the HashMap is implemented with external chaining. Assume the size of the internal array of the HashMap is 2 and doesn’t resize. Determine the output of each print line below:

```java
Lamp a = new Lamp(1);
Lamp b = new Lamp(2);

HashMap<Lamp, Integer> map = new HashMap<>();
map.put(b, 0);
map.put(a, 1);
map.put(a, 2);

System.out.println(map.get(a)); // print statement 1
System.out.println(map.get(b)); // print statement 2
map.put(b, 3);
a.brightness = 2;
map.put(b, 4);

System.out.println(map.get(a)); // print statement 3
System.out.println(map.get(b)); // print statement 4
System.out.println(map.get(new Lamp(1))); // print statement 5
```
Print Statement 1:

( ) 0
( ) 1
( ) 2
( ) 3
( ) 4
( ) null

Print Statement 2:

( ) 0
( ) 1
( ) 2
( ) 3
( ) 4
( ) null

Print Statement 3:

( ) 0
( ) 1
( ) 2
( ) 3
( ) 4
( ) null

Print Statement 4:

( ) 0
( ) 1
( ) 2
( ) 3
( ) 4
( ) null

Print Statement 5:

( ) 0
( ) 1
( ) 2
( ) 3
( ) 4
( ) null