Brock Boehler

[bvb2@illinois.edu](mailto:bvb2@illinois.edu)

ECE 374 B Spring 2019

Homework 0

**Problem 1**

**(a)** Use proof by induction:

*Inductive Hypothesis:* Assume that

is true for all arbitrary strings x such that 0 ≤ |x| < |w| and any arbitrary string y, and that w is some arbitrary string

*Base Case:* prove for x = :

|  |  |
| --- | --- |
|  |  |
|  | Replace x with |
|  | Definition of digsum |
|  | Additive Identity |

*Inductive Step:* Let :

|  |  |
| --- | --- |
|  |  |
|  | Replace w with ax |
|  | Associative property of concatenation |
|  | Definition of digsum |
|  | Induction Hypothesis |

∴ It has now been shown that for all strings x and y.

**(b)** Use proof by induction:

*Inductive Hypothesis:* Assume that is true for all arbitrary strings x such that 0 ≤ |x| < |w|, with w being some arbitrary string.

*Base Case:* prove for x =

|  |  |
| --- | --- |
|  |  |
|  | Replace x with |
|  | Definition of reversal |
|  | Subtract from both sides |

*Inductive Step:* Let :

|  |  |
| --- | --- |
|  |  |
|  | Replace w with ax |
|  | Definition of reversal |
|  | from problem 1A. |
|  | Definition of digsum |
|  | Inductive hypothesis |

∴ It has now been shown that for all strings x.

**Problem 2**

**(a)** ‘374’ is a 3-digit string, therefore it can only be represented in the form ‘ax’ or ‘axb’:

* Let ‘374’ be represented in the form axb with a = 3, b = 4, and x = 7. We know that ‘374’ is not in L­odd because, by definition, must be true but .
* Let ‘374’ be represented in the form ax then, with a = 3 and x = 74. It is not in Lodd in this case either, because as necessary. And because if we represent ’74’ in the same ax form

**(b)** Prove by induction:

*Inductive Hypothesis:* Suppose that for any , is odd for any arbitrary string x such that 0 < |x| < |w|, with w being some arbitrary string.

*Base Case:* Let x = a where . Since x is only one digit long, that means the result of is simply equal to a. Because a is limited to 1, 3, 5, 7, or 9 (all odd numbers), the result of will also be odd.

*Inductive Step:* Let w be any arbitrary string in Lodd.

W can be composed in the following two ways:

1.)

1. . This is true by the theorem of problem 1A

2. We know that and are both odd, because a and b are both odd-number one-character strings. We know that is odd by our inductive hypothesis.

3. is an even number, because an odd number plus an odd number is an even number.

4. is an even number plus an odd number, which results in an odd number (see below proof).

5. is then an odd number from (1.)

2.) :

1. . This is true from the theorem of problem 1A.

2. We know that is even because a is a one-character string of an even number, therefore is even because a is even

3. We know that is an odd number by our inductive hypothesis

4. is an even number plus an odd number, which results in an odd number.

4. is then an odd number from (1.)

∴ In both cases it is clearly shown then that for any , is odd for all strings x.

**Problem 3**

Define “bar” operation as follows on

Define ­L­bad as follows:

* for
* for () and (, for any string z) and ()
* for () and (, for any string z) and ()

Justification:

We can define Lbad recursively. We start off with our base case, either 0 or 1, to which we can build onto. Assume that Z is any string in Lbad. We then can make a new string from Z by either adding a character to the front (second bullet) or to the back (third bullet).

To add a character to the front three conditions must be satisfied: First, the string we are building onto must be a member of Lbad. Second, the character must be 0 or 1. Third, the string Z cannot have the character you wish to add twice in a row at the beginning. This means that if you wish to add a 0, Z cannot be of the form 001x, and if you wish to add a 1, Z cannot be of the form 110x, where x is any string. This is because adding the character in this case would result in 3 1’s or 0’s in a row, thereby eliminating it from Lbad.

The third bullet point about appending a character to the end uses the same logic as appending a character to the front, but just with the directions reversed.