Assignment 4 – Eric Wan – ezw23@drexel.edu

Question 1:

$\sum_{i=1}^{n} ba_i = b\sum_{i=1}^{n} a_i$

1. Checking if k=1 case is true
   $ba_1 = ba_1$
2. Assume k=n case is true
3. Prove k=n+1 is true
   $\sum_{i=1}^{n+1} ba_i = b\sum_{i=1}^{n+1} a_i$
   $\sum_{i=1}^{n} ba_i + (ba_{n+1}) = b\sum_{i=1}^{n} a_i + b(a_{n+1})$
   - b/c we assumed the k=n case is true:
   $ba_{n+1} = ba_{n+1}$

$\sum_{i=1}^{n}(a_i + b_i) = \sum_{i=1}^{n} a_i + \sum_{i=1}^{n} b_i$

1. Checking if k=1 case is true
   $(a_1 + b_1) = (a_1) + (b_1)$
2. Assume k=n case is true
3. Prove k=n+1 case is true
   $\sum_{i=1}^{n+1}(a_i + b_i) = \sum_{i=1}^{n+1} a_i + \sum_{i=1}^{n+1} b_i$
   $\sum_{i=1}^{n}(a_i + b_i) + (a_{n+1} + b_{n+1}) = \sum_{i=1}^{n} a_i + (a_{n+1}) + \sum_{i=1}^{n} b_i + (b_{n+1})$
   - b/c we assumed the k=n case is true:
   $(a_{n+1} + b_{n+1}) = (a_{n+1}) + (b_{n+1})$

Question 2:

$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

1. Checking if k=1 case is true
   $(1)^2 = \frac{(1)[(1)+1][2(1)+1]}{6} = \frac{6}{6}$
2. Assume k=n case is true
3. Prove k=n+1 case is true
   $\sum_{i=1}^{n+1} i^2 = \frac{(n+1)[(n+1)+1][2(n+1)+1]}{6}$
   $\sum_{i=1}^{n} i^2 + (n+1)^2 = \frac{(n+1)(n+2)(2n+3)}{6}$
   $\frac{n(n+1)(2n+1)}{6} + \frac{6(n^2+2n+1)}{6} = \frac{(n+1)(n+2)(2n+3)}{6}$
   $2n^3 + 9n^2 + 13n + 6 = (n+1)(n+2)(2n+3)$
   $2n^3 + 9n^2 + 13n + 6 = 2n^3 + 9n^2 + 13n + 6$

Question 3:

$\sum_{i=1}^{n} 2^i = 2^{n+1} - 2$

1. Checking k=1 case is true
   $2^1 = 2^2 - 2$
2. Assuming k=n case is true

3. Prove k=n+1 case is true
   $\sum_{i=1}^{n+1} 2^i = 2^{n+2} - 2$
   $\sum_{i=1}^{n} 2^i + 2^{n+1} = 2^{n+2} - 2$
   $2^{n+1} - 2 + 2^{n+1} = 2^{n+2} - 2$
   $2(2^{n+1}) - 2 = 2^{n+2} - 2$

Question 4:

N	Output
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

Base Cases:
XOR of 00 => replaced by 0
XOR of 01 => replaced by 1
XOR of 10 => replaced by 1
XOR of 11 => replaced by 0
- XOR of pairs recursively replaced until last pair is reached, and if the numbers of the XOR are the same, outputs a 0, else a 1
- XOR of an even number of 1s will eventually be simplified to 0 from the base case
e.g.
0101 => (01)01 => 101 => (10)1 => 11 => 0

1. Prove k=1, 2, 3 are true (assuming values of x are 1s)
   $parity(x_1) = x_1 = 1$
   even # of 1s: $parity(x_1, x_2) = x_1 \oplus x_2 = 0$
   odd # of 1s: $parity(x_1, x_2, x_3) = x_1 \oplus x_2 \oplus x_3 = 1$
2. Assuming k=n is true
3. Prove k=n+1 is true
   $parity(x, \dots, x_{n+1}) = x_1 \oplus \dots \oplus x_{n+1}$
   $parity(x, \dots, x_{n+1}) = x_1 \oplus \dots \oplus x_n \oplus x_{n+1}$
   - we know that the parity up to n will output the correct number indicating the number of 1s (1 is odd, 0 if even)
   - XORing the n+1 term will result in the same result making it true (if n was even, the n+1 term is odd and 0 XOR 1 is 1 staying true to the parity function; if n was odd, the n+1 term is even and 1 XOR 1 is 0 staying true to the parity function)

Question 5:
$G_1 = [0,1]; G_n = [0G_{n-1}, 1G'_{n-1}]$

1. Prove that k=1 is true
   $G_1 = [0(G_0), 1(G'_0)]$ $G_2 = [0(G_1), 1(G'_1)]$
   $G_1 = [0(), 1()]$ $G_2 = [0([0,1]), 1([0,1]')]$
   $G_1 = [0,1]$ $G_2 = [00,01,11,10])]$
2. Assuming k=n is true
   $G_n = [0,1,2, \dots, 2^n - 1]$
3. Prove that k=n+1 is true
   $G_{n+1} = [0G_n, 1G'_n]$
   $G_{n+1} = [0[G_n], 1[G_n]']$
   $G_{n+1} = [0[G_1, \dots, G_n], 1[G_n, \dots, G_1]]$
   $G_{n+1} = [0G_1, \dots, 0G_n, 1G_n, \dots, 1G_1]$
   - G_n iterates all the way to G_0 to create a list of all binary numbers of n bits
   - The G_{n+1} case follow suit with the G_n case with the addition of another bit up to $[0,1,2, \dots, 2^{n+1} - 1]$ which is adding a bit to all the G_n bit values $[0,1,2, \dots, 2^n - 1]$
   - The G_{n+1} case adds a leading 1 or 0 to all G_n cases
   - b/c $G_n$is true and includes all possible binary representations of n bits, then G_{n+1} is true due to adding a 0 to the front of all G_n values and 1 to the front of all G'_n values, all bit representations are accounted for in the G_{n+1} case
   - G_n is grey code by assumption, and heading it with a 0 for the G_{n+1} case will still result in grey code as there is still a change of only 1 bit
   - G'_n is grey code by assumption and heading it with a 1, for the G_{n+1} case will still result in grey code as there is still a change of only 1 bit
   - going from $0G_n$ to $1G_n$ is a change of the first, n+1, bit and nothing else meaning it stays consistent with the definition of grey code
   - going to $0G_1$ from $1G_1$ is a change of the first, n+1, bit and nothing else meaning it stays consistent with the definition of grey code
   - Thus G_{n+1} is still grey code

Question 6:

The append function always adds the second input to the end of the first input. The program will append the reverse function on the rest of the list with the first value of the list. Once it iterates to the final value of the list, it will append that to a null list and then append every subsequent value before it.

e.g. (reverse '()) will do the following
'()

e.g. (reverse '(1)) will do the following
(append (reverse '()) (cons '(1) null))
(append null '(1)) => '(1)

e.g. (reverse '(1 0)) will do the following
(append (reverse '(0)) (cons '(1) null))

(append ((append (reverse '()) (cons '(0) null))) (cons '(1) null))
(append ((append null '(0))) '(1))
(append '(0) '(1)) => '(0 1)

if L is null, (reverse L) outputs null
assuming (reverse '(L)) is true
e.g (reverse '(M) will do the following: (where M is '(x L))
(append (reverse '(L)) (cons '(x) null))
b/c (reverse '(L)) is true, reverse will output x at the end of reverse L
the input is '(x L) and the order is reversed, thus the function is valid

Extra Credit:

$G_n(i) = i\hat{}(i \gg 1)$

1. Prove that k=2 (i=0, 1, 2, 3) is true

   showing k=3

   $G_2(0) = 00\hat{}00 = 00$ 000^000 = 000 100^010 = 110
   $G_2(1) = 01\hat{}00 = 01$ 001^000 = 001 101^010 = 111
   $G_2(2) = 10\hat{}01 = 11$ 010^001 = 011 110^011 = 101
   $G_2(3) = 11\hat{}01 = 10$ 011^001 = 010 111^011 = 100

   k=2 is a binary-reflected Grey code, as it includes all binary representations of 2 bits
2. Assuming k=n is true

   $G_n(i) = i\hat{}(i \gg 1)$

   i = 0, 1, …, (2^n)-1

   the G_n case will produce the values, i, and the 2^n value will move into the n+1 bit place
3. Prove that k=n+1 is true

   $G_{n+1}(i) = (i)\hat{}(i \gg 1)$

   i = 0, 1, …, 2^(n+1)-1 = 0, 1, …, 2(2^n)-1

   - The G_{n+1} case follows similar steps to the G_n case and thus will produce a list i up to the n+1 bit, but subtracting the 1 indicates that it does not go into the n+2 bit
   - The G_{n+1} case adds an additional preceding bit to all G_n cases for evaluation
   - for all the G_{n+1} cases with a 0 as the leading bit, the calculations of the values of i will be no different from the calculations of G_n just with a leading bit 0. This makes it a grey-code.
   - for all the G_{n+1} cases with a 1 as the leading bit, the calculations of the values of i will be similar to the G_{n+1} cases with a 0 as the leading bit as the leading one will output the last G_{n+1} 0 cases with a leading one instead of a 0 from the xor between the bits. This makes it a grey code.
   - the G_{n+1} cases with a leading 1 produces the output of question 5: $1G_n, \ldots, 1G_1$ and thus the last G_{n+1} case with a leading 0 and the first G_{n+1} cases with a leading 1 differs by the leading bit making it comply with being a grey code and reflecting. The final G_{n+1} cases with a leading 1 produces a 1 followed by all 0s which will lead back to the first G_{n+1} cases with a leading 0 making it still comply with being a reflected grey-code.
   - Because all the criteria are met, $G_n(i) = (i)\hat{}(i \gg 1)$ does indeed produce a binary-reflected grey-code