Assignment 2

1. Q1
   1. Decimal
      1. 1100112 = 5110
   2. Decimal(Signed)
      1. 001100+1=001101=13
      2. 110011signed integer binary = -13
   3. Hexadecimal
      1. 1100112 = 3316
   4. ASCII
      1. 1100112 -> 5110 -> 3ASCII
2. Q2
   1. 86712 – 530912 = 123110 - 908110 = -785010 = -466212
   2. 111001102 – 000101112 -> -(00011010) – 00010111 -> -26 – 23 = -4910 = 11001111
   3. 0xDEAD16 + 0xBEEF16
      1. 5704110 + 6116710 = 11820810
      2. = 1CDC016
   4. 15810 – 134710
      1. 00000000100111102 – 0000000101010000112
      2. 11111011010110112
3. Q3
   1. 1610\*4210
      1. = 100002 \* 1010102
      2. = 10101000002
   2. 21210 / 1810
      1. = 110101002 / 100102
      2. 1011 with a remainder 1110
      3. 11 with a remainder 14
      4. 11 + (14/18)
      5. 11 + (7/9)
4. Q4
   1. A)

|  |  |  |  |
| --- | --- | --- | --- |
| * + 1. A | * + 1. B | * + 1. C | * + 1. OUTPUT |
| * + 1. 0 | * + 1. 0 | * + 1. 0 | * + 1. 0 |
| * + 1. 0 | * + 1. 0 | * + 1. 1 | * + 1. 0 |
| * + 1. 0 | * + 1. 1 | * + 1. 0 | * + 1. 0 |
| * + 1. 0 | * + 1. 1 | * + 1. 1 | * + 1. 1 |
| * + 1. 1 | * + 1. 0 | * + 1. 0 | * + 1. 0 |
| * + 1. 1 | * + 1. 0 | * + 1. 1 | * + 1. 1 |
| * + 1. 1 | * + 1. 1 | * + 1. 0 | * + 1. 1 |
| * + 1. 1 | * + 1. 1 | * + 1. 1 | * + 1. 0 |

* 1. B) This expression essentially says that check if only two of the three inputs are true, if all three are true return false.

1. Q5
   1. The first proof is that you could use a truth table. Both expressions have the following truth table

|  |  |  |  |
| --- | --- | --- | --- |
| * + 1. X | * + 1. Y | * + 1. Z | * + 1. OUTPUT |
| * + 1. 0 | * + 1. 0 | * + 1. 0 | * + 1. 1 |
| * + 1. 0 | * + 1. 0 | * + 1. 1 | * + 1. 1 |
| * + 1. 0 | * + 1. 1 | * + 1. 0 | * + 1. 1 |
| * + 1. 0 | * + 1. 1 | * + 1. 1 | * + 1. 0 |
| * + 1. 1 | * + 1. 0 | * + 1. 0 | * + 1. 0 |
| * + 1. 1 | * + 1. 0 | * + 1. 1 | * + 1. 1 |
| * + 1. 1 | * + 1. 1 | * + 1. 0 | * + 1. 1 |
| * + 1. 1 | * + 1. 1 | * + 1. 1 | * + 1. 1 |

* 1. You can also then prove this by using basic logic
     1. If you evaluate each part in the second expression you can find that (~X+Y)\*(X+~Y) has an equivalent truth table to an XNOR expression. So it can be rewritten as this ~(X⊕Y). So we have simplified to ~(X⊕Y)+((Z\*~Y)+(~Z\*Y))
        1. XNOR and (~X+Y)\*(X+~Y) truth table

|  |  |  |
| --- | --- | --- |
| * + - 1. X | * + - 1. Y | * + - 1. Output |
| * + - 1. 0 | * + - 1. 0 | * + - 1. 1 |
| * + - 1. 0 | * + - 1. 1 | * + - 1. 0 |
| * + - 1. 1 | * + - 1. 0 | * + - 1. 0 |
| * + - 1. 1 | * + - 1. 1 | * + - 1. 1 |

* + 1. Then (Z\*~Y)+(~Z\*Y) can be put into a truth table and it is discovered that it simulates an XOR Gate. So it can thus be rewritten as (Z⊕Y).

|  |  |  |
| --- | --- | --- |
| * + - 1. Z | * + - 1. Y | * + - 1. Output |
| * + - 1. 0 | * + - 1. 0 | * + - 1. 0 |
| * + - 1. 0 | * + - 1. 1 | * + - 1. 1 |
| * + - 1. 1 | * + - 1. 0 | * + - 1. 1 |
| * + - 1. 1 | * + - 1. 1 | * + - 1. 0 |

* + 1. We now have an expression ~(X⊕Y)+(Z⊕Y) but we need ~((X⊕Y)\*~(Z⊕Y)). Demorgans theorem states that ~(AB) is equal to ~A + ~B. So if we apply this theorem ~(X⊕Y)+(Z⊕Y) becomes ~((X⊕Y) \* ~(Z⊕Y)). The not inside the brackets gets flipped around because that is required by demorgans theorem. This can be proven because if the brackets were to be open and distributed out the not would be applied to the (X⊕Y) making it not and to the ~(Z⊕Y) which would make it a double not and effectively cancel out.