**IN0013 week 2 exercise**

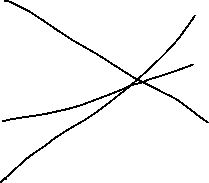
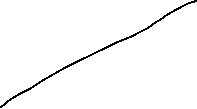
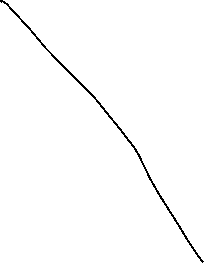
Logic gates

1. The diagram below shows 5 logic gates and 5 names. Draw a line matching the logic gates with the respective names



Diagram

Description automatically generated



1. Complete the truth table for the NOR gate

A picture containing diagram

Description automatically generated



|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  | 0 |
|  |  | 0 |
|  |  | 0 |
|  |  | 1 |

1. Write a logic statement that corresponds to the following logic circuit

Diagram

Description automatically generated

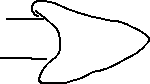
(A AND B) AND (NOT B OR C)

1. One way to think of the basic logic gate types (all but the XOR and XNOR gates) is to consider what single input state guarantees a certain output state. For example, we could describe the function of an OR gate as such:

*Any high input guarantees a high output.*

Identify what type of gate is represented by each of the following phrases:

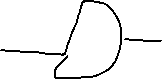
* Any high input guarantees a low output: NOR



* Any low input guarantees a high output: NAND



* Any low input guarantees a low output: AND



1. Design a logic circuit to model the requirement for a gym membership. To become a member of this prestigious gym (X = 1), the 4 criteria requirements are listed below

Table

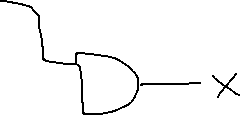
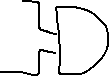
Description automatically generated

Membership is approved (X = 1) if the person:

* is over the age of 18 and has been recommended by a pre-existing member and
* either is working full-time or is retired, but not both.

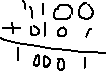
Draw a logic circuit to represent the membership requirement and draw the truth table showing the logic results. X = (A and B) and (C xor D)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | C | D | A AND B | C XOR D | (A AND B) AND (C XOR D) |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

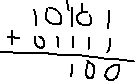


Binary

1. Write the following decimal number as a binary number
   1. 4 = 100
   2. 11 = 1011
   3. 16 = 10000
   4. 19 = 1 0011
   5. 59 = 11 1011
2. Write the following binary number as a decimal number
   1. 11 = 3
   2. 110 = 6
   3. 1100 =2^3 + 2^2 + 0 + 0 = 8 + 4 = 12
   4. 1101 = 13
   5. 101111 = 47
3. Add these binary numbers
   1. 11 + 11 = 110
   2. 1100 + 0101



* 1. 1000 +0011 = 1011
  2. 10101+01111



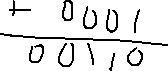
1. Two complements – use two complements to compute these binary operations
   1. 110 – 101 =



* 1. 1001 – 0111 =



* 1. 1000 – 0011 = 1000 + (1100) = 1000 + (1101) = 0101
  2. 10101 – 01111 =



1. ASCII numbers – transform these letters into ascii or the other way around
   1. computer = 99 111 109 112 117 116 101 114



* 1. fundamental = 102 117 110 100 97 109 101 110 116 97 108
  2. 80 121 116 104 111 110 = Python