|  |  |  |
| --- | --- | --- |
| A picture containing text, clipart  Description automatically generated | **INTRODUCTION TO COMPUTING**  **Assignment 2** | Instructor: Drakhshan Bokhat  Total Marks: 10  Marks Obtained: \_\_\_\_\_\_\_\_ |
| Roll No: \_\_2021-SE-39\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Mapping CLOs: CLO1** |

**Activity 02a The Unicode System**

A. The Unicode Table (unicode-table.com) is a universal way to encode every character in every language into a single system of Hexadecimal Numbers.

Aside from the Latin alphabet, there are many other alphabets represented in the Unicode Table. Name 3 different alphabets, and their starting point (as a Unicode value, in base 16)

|  |  |  |
| --- | --- | --- |
| **Letter** |  | **Unicode** |
|  |
| ب |  | **U+0628** |
| پ |  | U+067E |
| ت |  | U+062A |

B. What patterns do you see between Unicode values of the Upper Case and Lower-Case letters of the English alphabet? Explain this pattern in the context of base 16.

The Ascii Code of “A” in Decimal is 65 and in base 16 is 41

The Ascii Code of “a” in Decimal is 97 and in base 16 is 61

The Difference between Capital and Small alphabets in base 16 is of 20 Decimal numbers.

C. Unicode characters are often written in Hexadecimal. If you look at Unicode table, you can see how the underlying nature of base 16 helps organize the table. Explain how base 16 is used to organize Unicode characters.

The Unicode standard defines values for over 128,000 characters and can be seen at the Unicode Consortium. ... UTF-16: Uses two bytes (16 bits) to encode **the most commonly used characters**. If needed, the additional characters can be represented by a pair of 16-bit numbers.

**Activity 02b Introduction to Circuits in Logic.ly**

1. Logic Gates use electricity to do much of the work of a computer. Go to the web site <http://logic.ly/demo>

|  |  |  |  |
| --- | --- | --- | --- |
| Name | AND | OR | NOT |
| picture | Untitled Circuit* - Logic.ly Online Demo - Personal - Microsoft​ Edge | **Untitled Circuit* - Logic.ly Online Demo - Personal - Microsoft​ Edge** | **Untitled Circuit* - Logic.ly Online Demo - Personal - Microsoft​ Edge** |
| how it works | If all the input values are 1 then output is 1 else 0 | If all the input values are 0 then output is 0 else 1 | It inverts the input values |

B. Make this in Logic.ly. (A or B) and not C.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Draw its picture in the space below  Untitled Circuit* - Logic.ly Online Demo - Personal - Microsoft​ Edge | Make a table of all possible values of A, B, and C.  Determine if the light is on or off for each possibility.   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | Light | | on | on | on | On | | on | on | off | On | | on | off | on | On | | on | off | off | On | | off | on | on | On | | off | on | off | On | | off | off | on | On | | off | off | off | Off | |

**Activity 01b Introduction to Circuits in Logic.ly page 2**

C. George Boole was a British mathematician who invented a kind of mathematics called Boolean Algebra. It uses the values of True and False to represent “on” and “off”. A table of all possible outputs of the variables is called a Truth Table.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Draw a logic gate picture: not (A or B) and C  Untitled Circuit* - Logic.ly Online Demo - Personal - Microsoft​ Edge | Make a table of all possible values of A, B, and C. Write True or False   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | Output | | T | T | T | F | | T | T | F | F | | T | F | T | T | | T | F | F | F | | F | T | T | T | | F | T | F | F | | F | F | T | T | | F | F | F | F | |

D. Below is a circuit. The picture shows F, F, F

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Screen Shot 2016-09-22 at 7.50.52 AM.png  write the equivalent Boolean expression: | Make a table of all possible values of A, B, and C. Write True or False (T or F)   |  |  |  |  | | --- | --- | --- | --- | | A | B | C | Output | | T | T | T | T | | T | T | F | T | | T | F | T | F | | T | F | F | F | | F | T | T | F | | F | T | F | F | | F | F | T | F | | F | F | F | F | |

**Activity 02c Code.org Lossless Compression**

A. Files can be huge, so we need to find ways to compress the data. Do a web search to find the Code.org Compression Widget. Watch the video. Then, complete the widget on Unit 2, Lesson 2, puzzle 2. Try to get the best compression rate you can.

|  |  |
| --- | --- |
| **Name of Poem/Song** | **Record Your Best Compression Rate** |
| So wake me up | 33.33% |
| I need a dollar | 29.27% |
| The man | 55% |
| Pitter Patter | 30% |
| A tutor | 21.27% |
| She sells | 38% |
| I know an old lady | 20.98% |

B. Take another look at your work in compression. What formula is being used to compute the compression rate?

Compute the compression rate for the following examples:

|  |  |  |
| --- | --- | --- |
|  | File 1 | File 2 |
| compressed text size | 45 | 876 |
| dictionary size | 6 | 43 |
| total | 51 | 919 |
| original text size | 73 | 1345 |
| compression rate | 13.77% | 14.63% |

**Activity 02d Hamming (7,4) code Source: https://en.wikipedia.org/wiki/Hamming(7,4)**

A Hamming (7,4) code is an example of a linear error-correcting code.

Steps to Encode Example 1 Example 2

|  |  |  |
| --- | --- | --- |
| 1. Start with some number from 0 - 15 that you want to encode. Imagine your number can stand for a character, a sound, a color, or really anything. | 13 (base 10) = 1 1 0 1(base2) | 5 (base 10) = 0 1 0 1 (base2) |
| 2. Take the four digits, and place them in a Venn Diagram according to the picture below:  d1 = digit 1, d2 = digit 2, ….  Diagram, venn diagram  Description automatically generated | Diagram  Description automatically generated |  |
| 3. Parity is kind of like remainder. Each circle should have an even parity.  Fill in the parity bits:  p1 = p2 = p3 = | Diagram  Description automatically generated | Diagram, venn diagram  Description automatically generated |
| 4. Rewrite the digits in the order shown below  Diagram, venn diagram  Description automatically generated  This is your encoded number!! | 1101 | 0101 |

**Steps to Decode:**

Example 1 Example 2

|  |  |  |
| --- | --- | --- |
| 1. Start with some Hamming (7,4) number that may or may not have an error in it. | **0 1 1 0 1 1 0** | **1 1 1 0 0 0 1** |
| 2. Take the seven bits, and place them in a Venn Diagram according to the picture below:  Diagram, venn diagram  Description automatically generated | Diagram, venn diagram  Description automatically generated[[1]](#footnote-1) | Diagram, venn diagram  Description automatically generated |
| 3. In the picture above, check each of the three parity spots. Mark any that have an error: | In P2 there is a parity error because it is odd parity while other are even | In P3 there is a parity error because it not odd nor even parity while other are even. |
| 4. Change just one bit in the diagram so that there are no errors. If you can’t, then you probably have 2 errors, and this number cannot be corrected | Change any bit to 0 or 1 to make it correct. | In p3 change any to bits to 1 to correct it. |
| 5. Rewrite your 4 data bits using the picture below:  Diagram, venn diagram  Description automatically generated  Convert the four bits to decimal!! | d1= 1  d2= 1  d3= 1  d4= 0 | d1= 1  d2= 0  d3= 0  d4= 0 |

This system works if we assume that it is unlikely to get 2 or more bits out of 7 incorrect

1. [↑](#footnote-ref-1)