ITCS323 Computer Data Communication

Programming Assignment: Error code

Submitted to

Asst. Prof. Boonsit Yimwadsana

Asst. Prof. Thitinan Tantidham

Presented by

Tharit Chantanalertvilai 6188068

Sarit Opaspakornkij 6188081

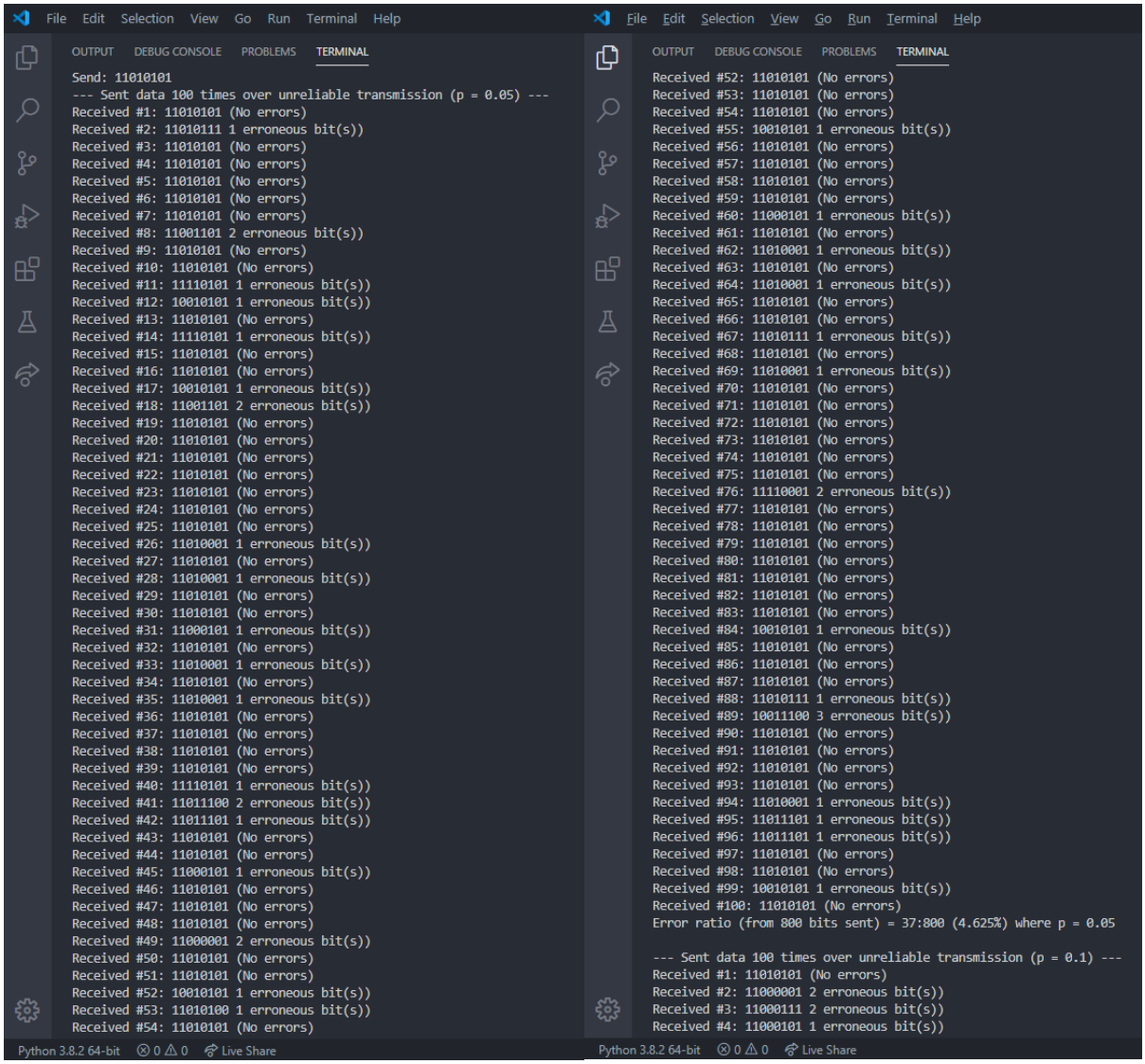
Mahidol University Salaya

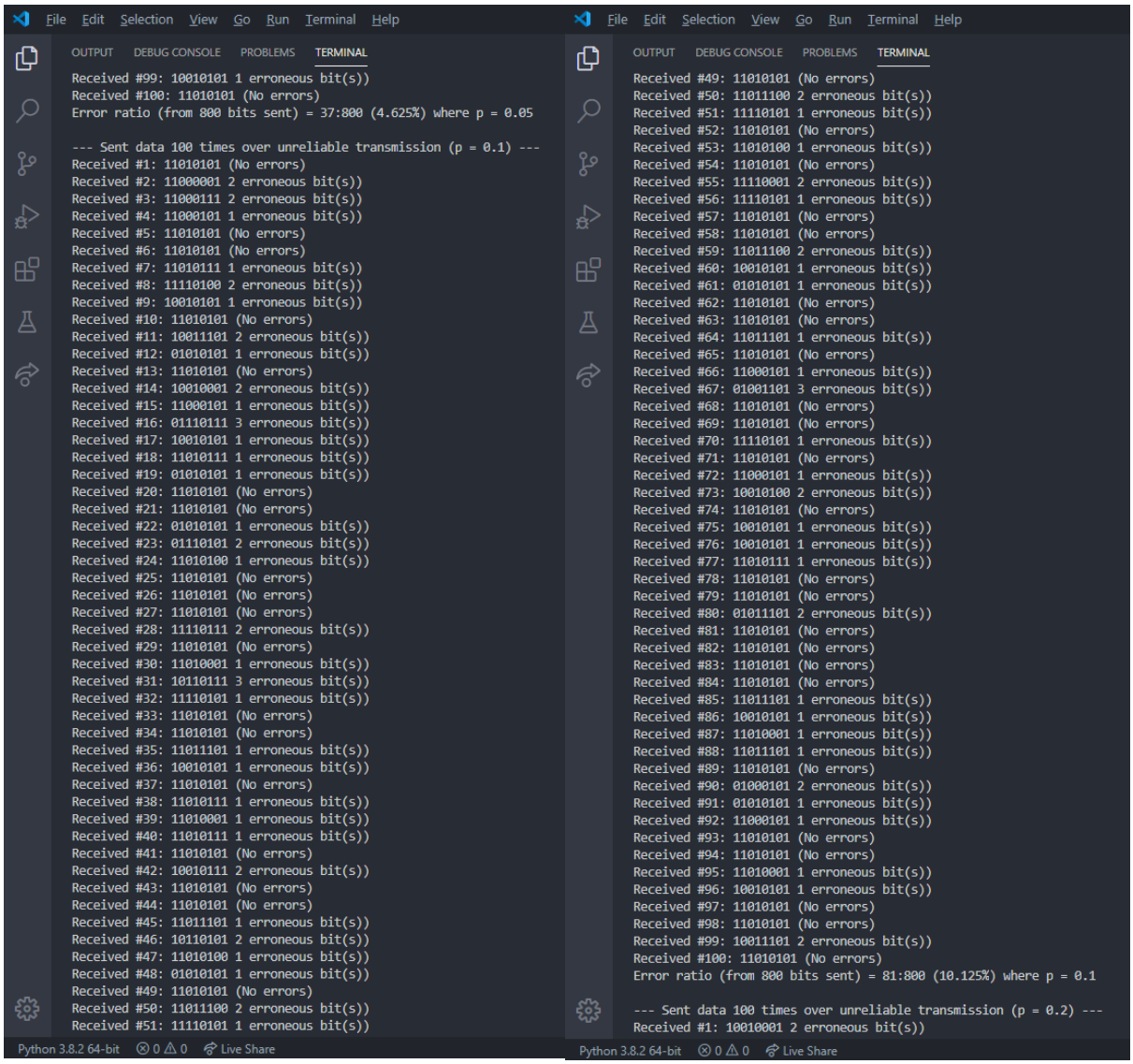
Faculty of Information and Communication Technology

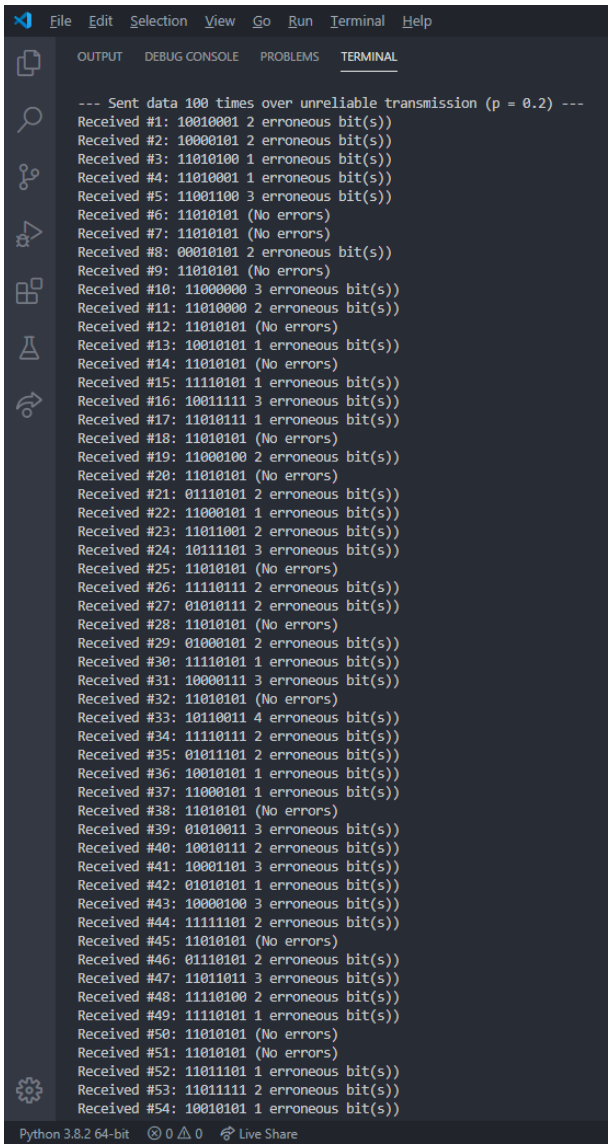
Unreliable Transmission

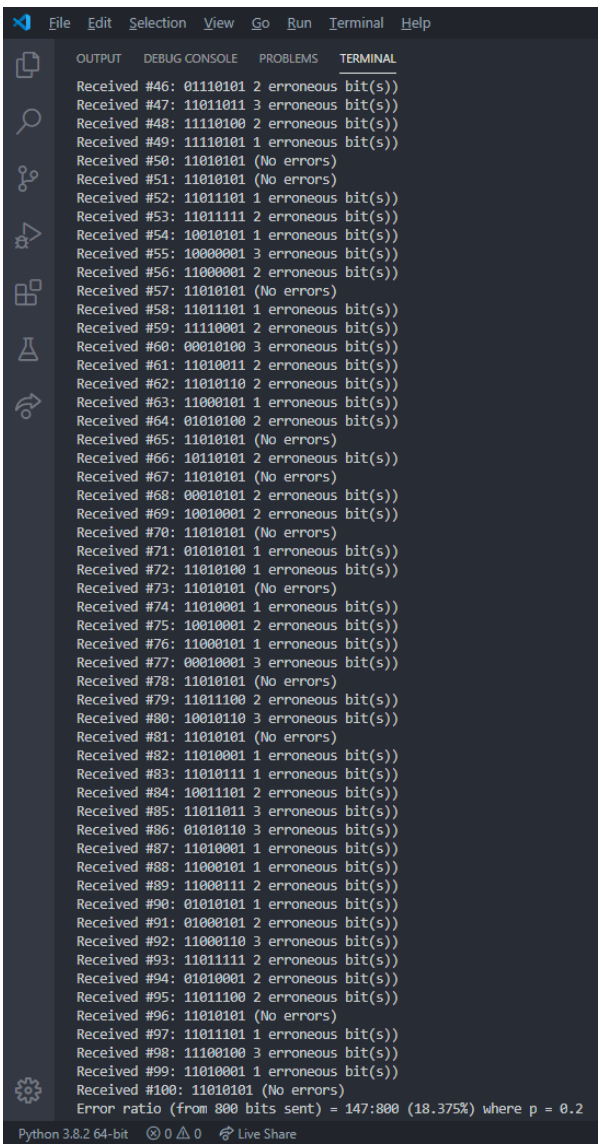
How to run the program (UnreliableTransmission.py): In `__main__`, you can change the input codeword by changing the value of `codeword` (line 24). Then, you can run via VS Code or the command line.

Testing Results:




File Edit Selection View Go Run Terminal Help
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL
Received #99: 10010101 1 erroneous bit(s))
Received #100: 11010101 (No errors)
Error ratio (from 800 bits sent) = 37:800 (4.625%) where p = 0.05
--- Sent data 100 times over unreliable transmission (p = 0.1) ---
Received #1: 11010101 (No errors)
Received #2: 11000001 2 erroneous bit(s))
Received #3: 11000111 2 erroneous bit(s))
Received #4: 11000101 1 erroneous bit(s))
Received #5: 11010101 (No errors)
Received #6: 11010101 (No errors)
Received #7: 11010111 1 erroneous bit(s))
Received #8: 11110100 2 erroneous bit(s))
Received #9: 10010101 1 erroneous bit(s))
Received #10: 11010101 (No errors)
Received #11: 10011101 2 erroneous bit(s))
Received #12: 01010101 1 erroneous bit(s))
Received #13: 11010101 (No errors)
Received #14: 10010001 2 erroneous bit(s))
Received #15: 11000101 1 erroneous bit(s))
Received #16: 01110111 3 erroneous bit(s))
Received #17: 10010101 1 erroneous bit(s))
Received #18: 11010111 1 erroneous bit(s))
Received #19: 01010101 1 erroneous bit(s))
Received #20: 11010101 (No errors)
Received #21: 11010101 (No errors)
Received #22: 01010101 1 erroneous bit(s))
Received #23: 01110101 2 erroneous bit(s))
Received #24: 11010100 1 erroneous bit(s))
Received #25: 11010101 (No errors)
Received #26: 11010101 (No errors)
Received #27: 11010101 (No errors)
Received #28: 11110111 2 erroneous bit(s))
Received #29: 11010101 (No errors)
Received #30: 11010001 1 erroneous bit(s))
Received #31: 10110111 3 erroneous bit(s))
Received #32: 11110101 1 erroneous bit(s))
Received #33: 11010101 (No errors)
Received #34: 11010101 (No errors)
Received #35: 11011101 1 erroneous bit(s))
Received #36: 10010101 1 erroneous bit(s))
Received #37: 11010101 (No errors)
Received #38: 11010111 1 erroneous bit(s))
Received #39: 11010001 1 erroneous bit(s))
Received #40: 11010111 1 erroneous bit(s))
Received #41: 11010101 (No errors)
Received #42: 10010111 2 erroneous bit(s))
Received #43: 11010101 (No errors)
Received #44: 11010101 (No errors)
Received #45: 11011101 1 erroneous bit(s))
Received #46: 10110101 2 erroneous bit(s))
Received #47: 11010100 1 erroneous bit(s))
Received #48: 01010101 1 erroneous bit(s))
Received #49: 11010101 (No errors)
Received #50: 11011100 2 erroneous bit(s))
Received #51: 11110101 1 erroneous bit(s))
Python 3.8.2 64-bit ⊗ 0 ⚠ 0 Live Share
File Edit Selection View Go Run Terminal Help
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL
Received #49: 11010101 (No errors)
Received #50: 11011100 2 erroneous bit(s))
Received #51: 11110101 1 erroneous bit(s))
Received #52: 11010101 (No errors)
Received #53: 11010100 1 erroneous bit(s))
Received #54: 11010101 (No errors)
Received #55: 11110001 2 erroneous bit(s))
Received #56: 11110101 1 erroneous bit(s))
Received #57: 11010101 (No errors)
Received #58: 11010101 (No errors)
Received #59: 11011100 2 erroneous bit(s))
Received #60: 10010101 1 erroneous bit(s))
Received #61: 01010101 1 erroneous bit(s))
Received #62: 11010101 (No errors)
Received #63: 11010101 (No errors)
Received #64: 11011101 1 erroneous bit(s))
Received #65: 11010101 (No errors)
Received #66: 11000101 1 erroneous bit(s))
Received #67: 01001101 3 erroneous bit(s))
Received #68: 11010101 (No errors)
Received #69: 11010101 (No errors)
Received #70: 11110101 1 erroneous bit(s))
Received #71: 11010101 (No errors)
Received #72: 11000101 1 erroneous bit(s))
Received #73: 10010100 2 erroneous bit(s))
Received #74: 11010101 (No errors)
Received #75: 10010101 1 erroneous bit(s))
Received #76: 10010101 1 erroneous bit(s))
Received #77: 11010111 1 erroneous bit(s))
Received #78: 11010101 (No errors)
Received #79: 11010101 (No errors)
Received #80: 01011101 2 erroneous bit(s))
Received #81: 11010101 (No errors)
Received #82: 11010101 (No errors)
Received #83: 11010101 (No errors)
Received #84: 11010101 (No errors)
Received #85: 11011101 1 erroneous bit(s))
Received #86: 10010101 1 erroneous bit(s))
Received #87: 11010001 1 erroneous bit(s))
Received #88: 11011101 1 erroneous bit(s))
Received #89: 11010101 (No errors)
Received #90: 01000101 2 erroneous bit(s))
Received #91: 01010101 1 erroneous bit(s))
Received #92: 11000101 1 erroneous bit(s))
Received #93: 11010101 (No errors)
Received #94: 11010101 (No errors)
Received #95: 11010001 1 erroneous bit(s))
Received #96: 10010101 1 erroneous bit(s))
Received #97: 11010101 (No errors)
Received #98: 11010101 (No errors)
Received #99: 10011101 2 erroneous bit(s))
Received #100: 11010101 (No errors)
Error ratio (from 800 bits sent) = 81:800 (10.125%) where p = 0.1
--- Sent data 100 times over unreliable transmission (p = 0.2) ---
Received #1: 10010001 2 erroneous bit(s))
Python 3.8.2 64-bit ⊗ 0 ⚠ 0 Live Share


File Edit Selection View Go Run Terminal Help
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL
--- Sent data 100 times over unreliable transmission (p = 0.2) ---
Received #1: 10010001 2 erroneous bit(s))
Received #2: 10000101 2 erroneous bit(s))
Received #3: 11010100 1 erroneous bit(s))
Received #4: 11010001 1 erroneous bit(s))
Received #5: 11001100 3 erroneous bit(s))
Received #6: 11010101 (No errors)
Received #7: 11010101 (No errors)
Received #8: 00010101 2 erroneous bit(s))
Received #9: 11010101 (No errors)
Received #10: 11000000 3 erroneous bit(s))
Received #11: 11010000 2 erroneous bit(s))
Received #12: 11010101 (No errors)
Received #13: 10010101 1 erroneous bit(s))
Received #14: 11010101 (No errors)
Received #15: 11110101 1 erroneous bit(s))
Received #16: 10011111 3 erroneous bit(s))
Received #17: 11010111 1 erroneous bit(s))
Received #18: 11010101 (No errors)
Received #19: 11000100 2 erroneous bit(s))
Received #20: 11010101 (No errors)
Received #21: 01110101 2 erroneous bit(s))
Received #22: 11000101 1 erroneous bit(s))
Received #23: 11011001 2 erroneous bit(s))
Received #24: 10111101 3 erroneous bit(s))
Received #25: 11010101 (No errors)
Received #26: 11110111 2 erroneous bit(s))
Received #27: 01010111 2 erroneous bit(s))
Received #28: 11010101 (No errors)
Received #29: 01000101 2 erroneous bit(s))
Received #30: 11110101 1 erroneous bit(s))
Received #31: 10000111 3 erroneous bit(s))
Received #32: 11010101 (No errors)
Received #33: 10110011 4 erroneous bit(s))
Received #34: 11110111 2 erroneous bit(s))
Received #35: 01011101 2 erroneous bit(s))
Received #36: 10010101 1 erroneous bit(s))
Received #37: 11000101 1 erroneous bit(s))
Received #38: 11010101 (No errors)
Received #39: 01010011 3 erroneous bit(s))
Received #40: 10010111 2 erroneous bit(s))
Received #41: 10001101 3 erroneous bit(s))
Received #42: 01010101 1 erroneous bit(s))
Received #43: 10000100 3 erroneous bit(s))
Received #44: 11111101 2 erroneous bit(s))
Received #45: 11010101 (No errors)
Received #46: 01110101 2 erroneous bit(s))
Received #47: 11011011 3 erroneous bit(s))
Received #48: 11110100 2 erroneous bit(s))
Received #49: 11110101 1 erroneous bit(s))
Received #50: 11010101 (No errors)
Received #51: 11010101 (No errors)
Received #52: 11011101 1 erroneous bit(s))
Received #53: 11011111 2 erroneous bit(s))
Received #54: 10010101 1 erroneous bit(s))
Python 3.8.2 64-bit
0 0
Live Share


File Edit Selection View Go Run Terminal Help
OUTPUT DEBUG CONSOLE PROBLEMS TERMINAL
Received #46: 01110101 2 erroneous bit(s))
Received #47: 11011011 3 erroneous bit(s))
Received #48: 11110100 2 erroneous bit(s))
Received #49: 11110101 1 erroneous bit(s))
Received #50: 11010101 (No errors)
Received #51: 11010101 (No errors)
Received #52: 11011101 1 erroneous bit(s))
Received #53: 11011111 2 erroneous bit(s))
Received #54: 10010101 1 erroneous bit(s))
Received #55: 10000001 3 erroneous bit(s))
Received #56: 11000001 2 erroneous bit(s))
Received #57: 11010101 (No errors)
Received #58: 11011101 1 erroneous bit(s))
Received #59: 11110001 2 erroneous bit(s))
Received #60: 00010100 3 erroneous bit(s))
Received #61: 11010011 2 erroneous bit(s))
Received #62: 11010110 2 erroneous bit(s))
Received #63: 11000101 1 erroneous bit(s))
Received #64: 01010100 2 erroneous bit(s))
Received #65: 11010101 (No errors)
Received #66: 10110101 2 erroneous bit(s))
Received #67: 11010101 (No errors)
Received #68: 00010101 2 erroneous bit(s))
Received #69: 10010001 2 erroneous bit(s))
Received #70: 11010101 (No errors)
Received #71: 01010101 1 erroneous bit(s))
Received #72: 11010100 1 erroneous bit(s))
Received #73: 11010101 (No errors)
Received #74: 11010001 1 erroneous bit(s))
Received #75: 10010001 2 erroneous bit(s))
Received #76: 11000101 1 erroneous bit(s))
Received #77: 00010001 3 erroneous bit(s))
Received #78: 11010101 (No errors)
Received #79: 11011100 2 erroneous bit(s))
Received #80: 10010110 3 erroneous bit(s))
Received #81: 11010101 (No errors)
Received #82: 11010001 1 erroneous bit(s))
Received #83: 11010111 1 erroneous bit(s))
Received #84: 10011101 2 erroneous bit(s))
Received #85: 11011011 3 erroneous bit(s))
Received #86: 01010110 3 erroneous bit(s))
Received #87: 11010001 1 erroneous bit(s))
Received #88: 11000101 1 erroneous bit(s))
Received #89: 11000111 2 erroneous bit(s))
Received #90: 01010101 1 erroneous bit(s))
Received #91: 01000101 2 erroneous bit(s))
Received #92: 11000110 3 erroneous bit(s))
Received #93: 11011111 2 erroneous bit(s))
Received #94: 01010001 2 erroneous bit(s))
Received #95: 11011100 2 erroneous bit(s))
Received #96: 11010101 (No errors)
Received #97: 11011101 1 erroneous bit(s))
Received #98: 11100100 3 erroneous bit(s))
Received #99: 11010001 1 erroneous bit(s))
Received #100: 11010101 (No errors)
Error ratio (from 800 bits sent) = 147:800 (18.375%) where p = 0.2
Python 3.8.2 64-bit
0 0
Live Share

Parity Bit

How to run the program (Parity.py): In `__main__`, you can change the input dataword for 1D parity by changing the value of `dataword_input` (line 189). For 2D party, you can change the input dataword and the size of block by changing the value of `dataword_input` (line 220) and `size` (line 222). Then, you can run via VS Code or the command line.

Testing Results:

```
--- Even ---
Dataword: 11011100
Send: 110111001
Received (5% Error):  110111001 (Valid)
Received (10% Error): 010111001 (Invalid)
Received (20% Error): 110011011 (Valid)
--- Odd ---
Dataword: 11011100
Send: 110111000
Received (5% Error):  110111000 (Valid)
Received (10% Error): 100110000 (Valid)
Received (20% Error): 110011001 (Valid)
--- Two Dimensional Even ---
Dataword: 1100111 1011101 0111001
Send: ['11001111', '10111011', '01110010', '00000110']
Received (5% Error):  ['11011111', '10111011', '01111010', '00000110'] (Invalid)
Received (10% Error): ['11100011', '10101011', '01110010', '00100110'] (Invalid)
Received (20% Error): ['11001111', '10011011', '00110110', '00000110'] (Invalid)
['11001110', '10111010', '01110011', '11111000']
['01000110', '10111010', '01110011', '11111000']
--- Two Dimensional Odd ---
Dataword: 1100111 1011101 0111001
Send: ['11001110', '10111010', '01110011', '11111000']
Received (5% Error):  ['11001110', '10111010', '01110011', '11111000'] (Valid)
Received (10% Error): ['01000110', '10111010', '01110011', '11111000'] (Invalid)
Received (20% Error): ['11011111', '11111010', '00110011', '11111000'] (Invalid)
```

```
--- Even ---
Dataword: 11011100 11011100
Send: 11011100110111000
Received (5% Error):  11011100110111000 (Valid)
Received (10% Error): 11010100010110000 (Invalid)
Received (20% Error): 11011100100110000 (Valid)
--- Odd ---
Dataword: 11011100 11011100
Send: 11011100110111001
Received (5% Error):  11001100110111001 (Invalid)
Received (10% Error): 11010101110111001 (Valid)
Received (20% Error): 10011100101111001 (Invalid)
--- Two Dimensional Even ---
Dataword: 1100111 1011101 0111001 0101001
Send: ['11001111', '10111011', '01110010', '01010011', '01010101']
Received (5% Error):  ['11001111', '10111011', '01110010', '01010011', '01010101'] (Valid)
Received (10% Error): ['11001111', '10111000', '01010010', '01010011', '01010101'] (Invalid)
Received (20% Error): ['11001111', '10101011', '00010000', '01000011', '01110001'] (Invalid)
--- Two Dimensional Odd ---
Dataword: 1100111 1011101 0111001 0101001
Send: ['11001110', '10111010', '01110011', '01010010', '10101010']
Received (5% Error):  ['11001110', '10111010', '01110011', '00010010', '10101010'] (Invalid)
Received (10% Error): ['11001100', '10111011', '01110001', '01010010', '10101000'] (Invalid)
Received (20% Error): ['11000110', '00111000', '11110011', '11010000', '10001011'] (Invalid)
```

```
--- Even ---
Dataword: 1100010101001
Send: 11000101010010
Received (5% Error):  11000101010000 (Invalid)
Received (10% Error): 11010101010000 (Valid)
Received (20% Error): 11001101011010 (Valid)
--- Odd ---
Dataword: 1100010101001
Send: 11000101010011
Received (5% Error):  11000101010001 (Invalid)
Received (10% Error): 11000111010001 (Valid)
Received (20% Error): 11011111000011 (Valid)
--- Two Dimensional Even ---
Dataword: 11001110 10110101 01101001 10010110
Send: ['110011101', '101101011', '011010010', '100101100', '100001000']
Received (5% Error):  ['110011101', '101101011', '011010010', '100101100', '100001000'] (Valid)
Received (10% Error): ['000011101', '101111011', '011110010', '100101100', '100000000'] (Invalid)
Received (20% Error): ['111111101', '101111010', '001011010', '110101100', '000100000'] (Invalid)
--- Two Dimensional Odd ---
Dataword: 11001110 10110101 01101001 10010110
Send: ['110011100', '101101010', '011010011', '100101101', '011110111']
Received (5% Error):  ['110011100', '101101010', '011010111', '100101101', '011110111'] (Invalid)
Received (10% Error): ['110011100', '101101010', '010010011', '100101101', '011110111'] (Invalid)
Received (20% Error): ['110011000', '101100010', '011110011', '100110101', '011110101'] (Invalid)
```

CRC

How to run the program (CRC.py): In `__main__`, you can change the input dataword for by changing the value of `dataword ` (line 103). Then, you can run via VS Code or the command line.

Testing Results:

```
--- CRC-4 ---
Dataword: 1110110
Send: 11101100101
Received (5% Error):  11101100101 (Valid)
Received (10% Error): 11101100101 (Valid)
Received (20% Error): 01001101100 (Valid)

--- CRC-8 ---
Dataword: 1110110
Send: 111011011101010
Received (5% Error):  111011001101010 (Invalid)
Received (10% Error): 111011011101010 (Valid)
Received (20% Error): 111011011101010 (Valid)

--- ReCRC-16 ---
Dataword: 1110110
Send: 1110110010000010110111
Received (5% Error):  1110110010000010110111 (Valid)
Received (10% Error): 1110110010000110110111 (Invalid)
Received (20% Error): 1110111011000000100011 (Invalid)

--- CRC-16 ---
Dataword: 1110110
Send: 1110110100000100110111
Received (5% Error):  1110101100000100110111 (Invalid)
Received (10% Error): 1110010100000100110111 (Invalid)
Received (20% Error): 0110110100000011110111 (Invalid)

--- CRC-24 ---
Dataword: 1110110
Send: 111011010010011001010110101011011
Received (5% Error):  111001010010011001010100101011011 (Invalid)
Received (10% Error): 111111010010011010011001011011 (Invalid)
Received (20% Error): 011111111010011001000110011011 (Invalid)

--- CRC-32 ---
Dataword: 1110110
Send: 11101101111101000110010010100000101010101
Received (5% Error):  11101101111101000110010010100000101010101 (Valid)
Received (10% Error): 11101101111011000110010010100000011001 (Invalid)
Received (20% Error): 11101011111101000110010011101000101010100 (Invalid)
```

```
--- CRC-4 ---
Dataword: 11001010
Send: 110010100110
Received (5% Error):  110010100110 (Valid)
Received (10% Error): 110010100100 (Invalid)
Received (20% Error): 110010100100 (Invalid)

--- CRC-8 ---
Dataword: 11001010
Send: 1100101000100100
Received (5% Error):  1100001000100100 (Invalid)
Received (10% Error): 1100100000100100 (Invalid)
Received (20% Error): 1100101001000101 (Invalid)

--- ReCRC-16 ---
Dataword: 11001010
Send: 1100101011000001000011001
Received (5% Error):  1000101011000001100011001 (Invalid)
Received (10% Error): 1100101011100101000111001 (Invalid)
Received (20% Error): 1101011000010000000101001 (Invalid)

--- CRC-16 ---
Dataword: 11001010
Send: 1100101000000001010111100
Received (5% Error):  1010101000000001010110100 (Invalid)
Received (10% Error): 0110101100100000010111100 (Invalid)
Received (20% Error): 1100001101001010101111100 (Invalid)

--- CRC-24 ---
Dataword: 11001010
Send: 110010100010101101001100010001100
Received (5% Error):  110110100010101101001100000101100 (Invalid)
Received (10% Error): 100010100010001101001101000001100 (Invalid)
Received (20% Error): 110000100011101101000000011010111 (Invalid)

--- CRC-32 ---
Dataword: 11001010
Send: 11001010011100100000000001000110010011111
Received (5% Error):  11001010011100100010000001000110010011111 (Invalid)
Received (10% Error): 11000110010100100101000001000110010011111 (Invalid)
Received (20% Error): 10001010111100110010101001011011111111 (Invalid)
```

Checksum

How to run the program (Checksum.py): In `__main__`, you can change the input dataword, word size, and number of blocks by changing the value of `dataword_input` (line 92), `word_size` (line 94), and `num_blocks` (line 95). Then, you can run via VS Code or the command line.

Test Results:

```
--- Checksum ---
Dataword: 10011001 11100010 00100100 10000100
Send: ['10011001', '11100010', '00100100', '10000100', '11011010']
Received (5% Error)  ['10011001', '01100010', '10100100', '10000100', '11011010'] ( Valid )
Received (10% Error) ['10011001', '11000011', '00101100', '10000100', '10011000'] ( Invalid )
Received (20% Error) ['10010001', '11010010', '01100010', '10000000', '11010010'] ( Invalid )
```

```
--- Checksum ---
Dataword: 00111101 00101001 11001001 11100101
Send: ['00111101', '00101001', '11001001', '11100101', '11101001']
Received (5% Error)  ['00110111', '00101001', '11001001', '11100101', '11101100'] ( Invalid )
Received (10% Error) ['00111101', '00101001', '11001001', '11100101', '11101001'] ( Valid )
Received (20% Error) ['01111101', '00100001', '01000101', '01100110', '11111101'] ( Invalid )
```

```
--- Checksum ---
Dataword: 10101001 00111001
Send: ['10101001', '00111001', '00011101']
Received (5% Error)  ['10111000', '00101001', '00011101'] ( Invalid )
Received (10% Error) ['10101001', '00111001', '00011101'] ( Valid )
Received (20% Error) ['00101011', '00111000', '00011101'] ( Invalid )
```

```
--- Checksum ---
Dataword: 00111010 11111111
Send: ['00111010', '11111111', '11000101']
Received (5% Error)  ['00111011', '11111111', '11000101'] ( Invalid )
Received (10% Error) ['00111010', '11111111', '11000101'] ( Valid )
Received (20% Error) ['01111010', '11110111', '11000101'] ( Invalid )
```

```
--- Checksum ---
Dataword: 00000000 11111111
Send: ['00000000', '11111111', '00000000']
Received (5% Error)  ['00000000', '11111111', '00000000'] ( Valid )
Received (10% Error) ['00110000', '11111111', '00000000'] ( Invalid )
Received (20% Error) ['11000000', '11111101', '01011000'] ( Invalid )
```

```
--- Checksum ---
Dataword: 00011000 11100111
Send: ['00011000', '11100111', '00000000']
Received (5% Error)  ['00011000', '11100111', '00000000'] ( Valid )
Received (10% Error) ['00111000', '11100101', '00000000'] ( Invalid )
Received (20% Error) ['00011100', '01110101', '00001100'] ( Invalid )
```

```
--- Checksum ---
Dataword: 01010110 11010101
Send: ['01010110', '11010101', '11010011']
Received (5% Error)  ['01010110', '11010101', '10010011'] ( Invalid )
Received (10% Error) ['01010110', '11010101', '11010011'] ( Valid )
Received (20% Error) ['01010110', '11000001', '10010100'] ( Invalid )
```

```
--- Checksum ---
Dataword: 01010110 11010101 11010101
Send: ['01010110', '11010101', '11010101', '11111101']
Received (5% Error)  ['01010110', '11010101', '11010101', '11111101'] ( Valid )
Received (10% Error) ['01010110', '11010101', '11010100', '11111101'] ( Invalid )
Received (20% Error) ['01000010', '11010100', '11110111', '11101101'] ( Invalid )
```

```
--- Checksum ---
Dataword: 10101011 01110101 00111101
Send: ['10101011', '01110101', '00111101', '10100001']
Received (5% Error)  ['10101111', '01110101', '00111100', '10100001'] ( Invalid )
Received (10% Error) ['10101011', '01110101', '00111101', '10100001'] ( Valid )
Received (20% Error) ['00101111', '00010101', '00111101', '10101101'] ( Invalid )
```

```
--- Checksum ---
Dataword: 11111111 00000000 11111111
Send: ['11111111', '00000000', '11111111', '00000000']
Received (5% Error)  ['11111011', '00000000', '11111111', '00000000'] ( Invalid )
Received (10% Error) ['11111111', '00000000', '11111111', '00000000'] ( Valid )
Received (20% Error) ['11110110', '00110001', '11101111', '00010001'] ( Invalid )
```

Hamming Code

How to run the program (HammingCode.py): In `__main__`, you can change the input dataword by changing the value of `dataword ` (line 102). Then, you can run via VS Code or the command line.

```
--- Hamming Code ---
Dataword: 1001101
Send: 10011100101
Received (5% Error): 10011100101 (Error Position -1)
Received (10% Error): 10011110101 (Error Position 5)
Received (20% Error): 11011100101 (Error Position 10)
```

```
--- Hamming Code ---
Dataword: 1001101
Send: 10011100101
Received (5% Error):  10011100101 (Error Position -1)
Received (10% Error): 10001100101 (Error Position 8)
Received (20% Error): 11011100101 (Error Position 10)
```

```
--- Hamming Code ---
Dataword: 0011100001
Send: 00111010001110
Received (5% Error):  00111010001010 (Error Position 3)
Received (10% Error): 00111010001110 (Error Position -1)
Received (20% Error): 00110010001110 (Error Position 10)
```

```
--- Hamming Code ---
Dataword: 1011010101
Send: 10110100101101
Received (5% Error):  10110100101101 (Error Position -1)
Received (10% Error): 10110101101101 (Error Position 7)
Received (20% Error): 11110100101101 (Error Position 13)
```

```
--- Hamming Code ---
Dataword: 1001010101011110
Send: 10010010101010110011
Received (5% Error):  10000010101010110011 (Error Position 18)
Received (10% Error): 10010110101010110011 (Error Position 16)
Received (20% Error): 10010010001010110011 (Error Position 13)
```

```
--- Hamming Code ---
Dataword: 11010
Send: 111010011
Received (5% Error):  111011011 (Error Position 4)
Received (10% Error): 111010011 (Error Position -1)
Received (20% Error): 111110011 (Error Position 6)
```

```
--- Hamming Code ---
Dataword: 11011101
Send: 110111101100
Received (5% Error):  110111101100 (Error Position -1)
Received (10% Error): 010111101100 (Error Position 12)
Received (20% Error): 111111101100 (Error Position 10)
```

```
--- Hamming Code ---
Dataword: 11101101
Send: 111011101111
Received (5% Error):  111001101111 (Error Position 8)
Received (10% Error): 011011101111 (Error Position 12)
Received (20% Error): 111011111111 (Error Position 5)
```

```
--- Hamming Code ---
Dataword: 110110
Send: 1100110000
Received (5% Error):  0100110000 (Error Position 10)
Received (10% Error): 1100110000 (Error Position -1)
Received (20% Error): 1100100000 (Error Position 5)
```

```
--- Hamming Code ---
Dataword: 1111
Send: 1111111
Received (5% Error):  1111111 (Error Position -1)
Received (10% Error): 1111011 (Error Position 3)
Received (20% Error): 1110111 (Error Position 4)
```