CN

Assignment 5

AIDS-A

roll -10

Task: Write a program for error detection and correction for 7/8 bits ASCII codes using Hamming Codes.

Code:

*import* random

def get\_parity\_positions(*data\_length*):

parity\_positions = []

i = 0

*while* (2 \*\* i) <= *data\_length* + len(parity\_positions):

parity\_positions.append(2 \*\* i - 1)

i += 1

*return* parity\_positions

def generate\_hamming\_code(*ascii\_bin*):

data\_bits = list(*ascii\_bin*)

parity\_positions = get\_parity\_positions(len(data\_bits))

*for* pos *in* parity\_positions:

data\_bits.insert(pos, '0')

*for* pos *in* parity\_positions:

parity\_value = 0

*for* i *in* range(pos, len(data\_bits), 2 \* (pos + 1)):

parity\_value ^= sum(int(data\_bits[j]) *for* j *in* range(i, min(i + pos + 1, len(data\_bits))))

data\_bits[pos] = str(parity\_value)

*return* ''.join(data\_bits)

def introduce\_error(*hamming\_code*):

error\_position = random.randint(0, len(*hamming\_code*) - 1)

flipped\_bit = '1' *if* *hamming\_code*[error\_position] == '0' *else* '0'

corrupted\_code = *hamming\_code*[:error\_position] + flipped\_bit + *hamming\_code*[error\_position + 1:]

print(f"Error introduced at position {error\_position + 1}")

*return* corrupted\_code, error\_position + 1

def detect\_and\_correct(*hamming\_code*):

data\_bits = list(*hamming\_code*)

parity\_positions = get\_parity\_positions(len(data\_bits))

error\_position = 0

*for* pos *in* parity\_positions:

parity\_value = 0

*for* i *in* range(pos, len(data\_bits), 2 \* (pos + 1)):

parity\_value ^= sum(int(data\_bits[j]) *for* j *in* range(i, min(i + pos + 1, len(data\_bits))))

*if* parity\_value:

error\_position += pos + 1

*if* error\_position and error\_position <= len(data\_bits):

print(f"Error detected at position {error\_position}")

*# Correct the error*

data\_bits[error\_position - 1] = '1' *if* data\_bits[error\_position - 1] == '0' *else* '0'

print(f"Error corrected at position {error\_position}")

*else*:

print("No error detected.")

*return* ''.join(data\_bits)

def extract\_original\_data(*corrected\_code*):

parity\_positions = get\_parity\_positions(len(*corrected\_code*))

original\_data = ''.join([*corrected\_code*[i] *for* i *in* range(len(*corrected\_code*)) *if* i not in parity\_positions])

*return* original\_data

ascii\_char = input("Enter a single character: ")

ascii\_bin = format(ord(ascii\_char), '08b')

print(f"Original ASCII binary: {ascii\_bin}")

hamming\_code = generate\_hamming\_code(ascii\_bin)

print(f"Hamming Code: {hamming\_code}")

*# Introduce error*

corrupted\_code, error\_pos = introduce\_error(hamming\_code)

print(f"Corrupted Hamming Code: {corrupted\_code}")

*# Detect and correct*

corrected\_code = detect\_and\_correct(corrupted\_code)

print(f"Corrected Hamming Code: {corrected\_code}")

*# Extract original ASCII binary*

recovered\_ascii\_bin = extract\_original\_data(corrected\_code)

recovered\_char = chr(int(recovered\_ascii\_bin, 2))

print(f"Recovered ASCII Character: {recovered\_char}")

Output:

