***Problem 1 Algorithm:***

function from\_Decimal\_to\_Binary(num):

binary\_representation = ""

while num > 0:

remainder = num % 2

binary\_representation = remainder + binary\_representation

num = num // 2

return binary\_representation

function from\_Decimal\_to\_Octal(num):

octal\_representation = ""

while num > 0:

remainder = num % 8

octal\_representation = remainder + octal\_representation

num = num // 8

print octal\_representation

function from\_Decimal\_to\_Hexadecimal(num):

Hexadecimal\_representation = ""

while num > 0:

remainder = num % 16

if remainder < 10 :

Hexadecimal\_representation = remainder + Hexadecimal\_representation

else if remainder == 10 :

Hexadecimal\_representation = "A" + Hexadecimal\_representation

else if remainder == 11 :

Hexadecimal\_representation = "B" + Hexadecimal\_representation

else if remainder == 12 :

Hexadecimal\_representation = "C" + Hexadecimal\_representation

else if remainder == 13 :

Hexadecimal\_representation = "D" + Hexadecimal\_representation

else if remainder == 14 :

Hexadecimal\_representation = "E" + Hexadecimal\_representation

else if remainder == 15 :

Hexadecimal\_representation = "F" + Hexadecimal\_representation

num = num // 16

print Hexadecimal\_representation

function check\_binary\_number(binary\_num):

while binary\_num > 0:

if binary\_num % 10 != 1 and binary\_num % 10 != 0:

return False

binary\_num = binary\_num // 10

if binary\_num == 0:

return binary\_num

function binary\_to\_decimal(binary\_num):

result = 0

counter = 0

while binary\_num > 0:

num = binary\_num % 10

added\_num = num \* (2^counter)

result = result + added\_num

binary\_num = binary\_num // 10

counter = counter + 1

return result

function binary\_to\_octal(binary\_num):

result = ""

counter = 0

if length(binary\_num) % 3 == 1:

binary\_num = "00" + binary\_num

else if length(binary\_num) % 3 == 2:

binary\_num = "0" + binary\_num

while counter < length(binary\_num):

three\_digits = substring(binary\_num, counter, counter+3)

dec\_of\_three\_digits = binary\_to\_decimal(three\_digits)

result = result + dec\_of\_three\_digits

counter = counter + 3

return result

function binary\_to\_hexadecimal(binary\_num):

result = ""

counter = 0

if length(binary\_num) % 4 == 1:

binary\_num = "000" + binary\_num

else if length(binary\_num) % 4 == 2:

binary\_num = "00" + binary\_num

else if length(binary\_num) % 4 == 3:

binary\_num = "0" + binary\_num

while counter < length(binary\_num):

four\_digits = substring(binary\_num, counter, counter+4)

dec\_of\_four\_digits = binary\_to\_decimal(four\_digits)

if dec\_of\_four\_digits == 10:

dec\_of\_four\_digits = "A"

else if dec\_of\_four\_digits == 11:

dec\_of\_four\_digits = "B"

else if dec\_of\_four\_digits == 12:

dec\_of\_four\_digits = "C"

else if dec\_of\_four\_digits == 13:

dec\_of\_four\_digits = "D"

else if dec\_of\_four\_digits == 14:

dec\_of\_four\_digits = "E"

else if dec\_of\_four\_digits == 15:

dec\_of\_four\_digits = "F"

result = result + dec\_of\_four\_digits

counter = counter + 4

return result

function octal\_to\_decimal(octal\_num):

if set(octal\_num) - set('01234567'):

return 'Invalid input. Please enter a valid octal number.'

decimal\_num = 0

base = 1

while octal\_num:

digit = int(substring(octal\_num, length(octal\_num)-1, length(octal\_num)))

decimal\_num = decimal\_num + digit \* base

base = base \* 8

octal\_num = substring(octal\_num, 0, length(octal\_num)-1)

return decimal\_num

function octal\_to\_binary(octal\_num):

if set(octal\_num) - set('01234567'):

return 'Invalid input. Please enter a valid octal number.'

decimal = 0

power = 3

for digit in octal\_num:

decimal = decimal + int(digit) \* (8 ^ power)

power = power - 1

binary = ""

while decimal > 0:

binary = str(decimal % 2) + binary

decimal = decimal // 2

return binary

function octal\_to\_hexadecimal(octal\_num):

octal\_to\_decimal = { '0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7 }

if set(octal\_num) - set('01234567'):

return 'Invalid input. Please enter a valid octal number.'

decimal\_num = 0

for digit in octal\_num:

decimal\_num = decimal\_num \* 8 + octal\_to\_decimal[digit]

decimal\_to\_hexadecimal = {‘0': '0', '1': '1', '2': '2', '3': '3', '4': '4', '5': '5', '6': '6', '7': '7', '8': '8', '9': '9',

'10': 'A', '11': 'B', '12': 'C', '13': 'D', '14': 'E', '15': 'F'}

hexadecimal\_num = ''

while decimal\_num > 0:

decimal\_num, remainder = divide(decimal\_num, 16)

hexadecimal\_num = decimal\_to\_hexadecimal[str(remainder)] + hexadecimal\_num

return hexadecimal\_num

function hex\_to\_decimal(hex\_num):

decimal\_num = 0

power = 0

hex\_to\_decimal\_map = {

'0': 0, '1': 1, '2': 2, '3': 3, '4': 4,

'5': 5, '6': 6, '7': 7, '8': 8, '9': 9,

'A': 10, 'B': 11, 'C': 12, 'D': 13, 'E': 14, 'F': 15,

'a': 10, 'b': 11, 'c': 12, 'd': 13, 'e': 14, 'f': 15

}

for digit in reverse(hex\_num):

decimal\_num = decimal\_num + hex\_to\_decimal\_map[digit] \* (16 ^ power)

power = power + 1

return decimal\_num

function hex\_to\_bin(hex\_num):

binary\_num = ""

for digit in hex\_num:

if digit == '0':

binary\_num = binary\_num + '0000'

else if digit == '1':

binary\_num = binary\_num + '0001'

else if digit == '2':

binary\_num = binary\_num + '0010'

else if digit == '3':

binary\_num = binary\_num + '0011'

else if digit == '4':

binary\_num = binary\_num + '0100'

else if digit == '5':

binary\_num = binary\_num + '0101'

else if digit == '6':

binary\_num = binary\_num + '0110'

else if digit == '7':

binary\_num = binary\_num + '0111'

else if digit == '8':

binary\_num = binary\_num + '1000'

else if digit == '9':

binary\_num = binary\_num + '1001'

else if digit == 'A':

binary\_num = binary\_num + '1010'

else if digit == 'a':

binary\_num = binary\_num + '1010'

else if digit == 'B':

binary\_num = binary\_num + '1011'

else if digit == 'b':

binary\_num = binary\_num + '1011'

else if digit == 'C':

binary\_num = binary\_num + '1100'

else if digit == 'c':

binary\_num = binary\_num + '1100'

else if digit == 'D':

binary\_num = binary\_num + '1101'

else if digit == 'd':

binary\_num = binary\_num + '1101'

else if digit == 'E':

binary\_num = binary\_num + '1110'

else if digit == 'e':

binary\_num = binary\_num + '1110'

else if digit == 'F':

binary\_num = binary\_num + '1111'

else if digit == 'f':

binary\_num = binary\_num + '1111

return binary\_num

function hex\_to\_oct(hex\_num):

hex\_values = {

'0': 0, '1': 1, '2': 2, '3': 3, '4': 4,

'5': 5, '6': 6, '7': 7, '8': 8, '9': 9,

'A': 10, 'B': 11, 'C': 12, 'D': 13, 'E': 14, 'F': 15,

'a': 10, 'b': 11, 'c': 12, 'd': 13, 'e': 14, 'f': 15

}

decimal\_num = 0

for digit in hex\_num:

decimal\_num = decimal\_num \* 16 + hex\_values[digit]

octal\_num = ""

while decimal\_num > 0:

octal\_num = str(decimal\_num % 8) + octal\_num

decimal\_num = decimal\_num // 8

return octal\_num

function main():

print("\*\* numbering system converter \*\*")

print("A) Insert a new number")

print("B) Exit program")

choice\_menu1 = input("Please select an option (A/B): ").toUpperCase()

while choice\_menu1 == "A" :

num = input("Please enter the number: ")

print("\*\* Please select the base you want to convert a number from \*\*")

print("A) Decimal")

print("B) Binary")

print("C) Octal")

print("D) Hexadecimal")

from\_base = input("Please select an option (A/B/C/D): ").toUpperCase()

print("\*\* Please select the base you want to convert a number to \*\*")

print("A) Decimal")

print("B) Binary")

print("C) Octal")

print("D) Hexadecimal")

to\_base = input("Please select an option (A/B/C/D): ").toUpperCase()

if from\_base == "A" and to\_base == "B":

print(from\_Decimal\_to\_Binary(num))

else if from\_base == "A" and to\_base == "C":

from\_Decimal\_to\_Octal(num)

else if from\_base == "A" and to\_base == "D":

from\_Decimal\_to\_Hexadecimal(num)

else if from\_base == "B" and to\_base == "A" :

if check\_binary\_number(num) == False:

print('\nInvalid binary number\n')

else:

print(binary\_to\_decimal(num))

else if from\_base == "B" and to\_base == "C" :

if check\_binary\_number(num) == False:

print('\nInvalid binary number\n')

else:

print(binary\_to\_octal(num))

else if from\_base == "B" and to\_base == "D" :

if check\_binary\_number(num) == False:

print('\nInvalid binary number\n')

else:

print(binary\_to\_hexadecimal(num))

else if from\_base == "C" and to\_base == "A" :

print(octal\_to\_decimal(num))

else if from\_base == "C" and to\_base == "B" :

print(octal\_to\_binary(num))

else if from\_base == "C" and to\_base == "D" :

print(octal\_to\_hexadecimal(num))

else if from\_base == "D" and to\_base == "B" :

print(hex\_to\_bin(num))

else if from\_base == "D" and to\_base == "A" :

print(hex\_to\_decimal(num))

else if from\_base == "D" and to\_base == "C" :

print(hex\_to\_oct(num))

print("\*\* numbering system converter \*\*")

print("A) Insert a new number")

print("B) Exit program")

choice\_menu1 = input("Please select an option (A/B): ").toUpperCase()

main()

***Problem 2 Algorithm:***

function decimal\_to\_binary(binary\_num):

set result to ""

set count to 0

set binary\_inloop to binary\_num

while (binary\_inloop != 0):

set result to result + str(binary\_inloop % 2)

set binary\_inloop to binary\_inloop // 2

if (binary\_num is 0):

set result to "0"

return result[::-1]

function binary\_to\_decimal(binary\_num):

set result to 0

set counter to 0

set binary\_num to int(binary\_num)

while (binary\_num > 0):

set num to binary\_num % 10

set added\_num to num \* (2 \*\* counter)

set result to result + added\_num

set binary\_num to binary\_num // 10

set counter to counter + 1

return result

function binary\_addition(binary\_num1, binary\_num2):

align\_binary\_numbers(binary\_num1, binary\_num2)

set result to ""

set carry to 0

for (x in range(length(binary\_num1))):

set result\_bit, carry to add\_bits(binary\_num1[x], binary\_num2[x], carry)

set result to result + result\_bit

if (carry is 1):

set result to result + "1"

return result[::-1]

function binary\_subtraction(binary\_num1, binary\_num2):

align\_binary\_numbers(binary\_num1, binary\_num2)

set result to ""

set carry to 0

for (x in range(length(binary\_num1))):

set result\_bit, carry to subtract\_bits(binary\_num1[x], binary\_num2[x], carry)

set result to result + result\_bit

if (carry is 1):

set result to result + "1"

return result[::-1]

function menu\_1():

print("\*\* binary calculator \*\*")

print("A) Insert new numbers")

print("B) Exit")

set menu1\_choice to get\_user\_input(">>> ").upper()

if (menu1\_choice is "B"):

exit\_program()

elif (menu1\_choice is "A"):

insert\_first\_number()

else:

print("Please insert a valid selection")

menu\_1()

function menu\_2(binary\_num1):

print("\*\* Please select the operation \*\*")

print("A) Compute one”s complement")

print("B) Compute two”s complement")

print("C) Addition")

print("D) Subtraction")

set menu\_2\_choice to get\_user\_input(">>> ").upper()

if (menu\_2\_choice is "C"):

set binary\_num2 to insert\_second\_number()

print(“The result of addition is {binary\_addition(binary\_num1, binary\_num2)}”)

menu\_1()

elif (menu\_2\_choice is "A"):

print(“The one complement is {one\_complement(binary\_num1)}”)

menu\_1()

elif (menu\_2\_choice is "B"):

print(“The two complement is {two\_complement(binary\_num1)}”)

menu\_1()

elif (menu\_2\_choice is "D"):

set binary\_num2 to insert\_second\_number()

print(“The result of subtraction is {binary\_subtraction(binary\_num1, binary\_num2)}”)

menu\_1()

else:

print("Please insert a valid selection")

menu\_2(binary\_num1)

function insert\_first\_number():

set binary\_num1 to get\_user\_input("Please insert a binary number: ")

set checking\_binary to check\_binary\_number(binary\_num1)

while (checking\_binary is False):

print("Please insert a valid binary number: ")

set binary\_num1 to get\_user\_input("Please insert a binary number: ")

set checking\_binary to check\_binary\_number(binary\_num1)

menu\_2(binary\_num1)

function insert\_second\_number():

set binary\_num2 to get\_user\_input("Please insert a second binary number: ")

set checking\_binary to check\_binary\_number(binary\_num2)

while (checking\_binary is False):

print("Please insert a valid binary number: ")

set binary\_num2 to get\_user\_input("Please insert a second binary number: ")

set checking\_binary to check\_binary\_number(binary\_num2)

return binary\_num2

function check\_binary\_number(binary\_num):

while (binary\_num > 0):

if (binary\_num % 10 != 1 and binary\_num % 10 != 0):

return False

set binary\_num to binary\_num // 10

if (binary\_num is 0):

return True

function one\_complement(binary\_num):

set binary\_num to list(binary\_num)

if (binary\_num[0] is "0"):

return ““.join(binary\_num)

else:

for (i in range(length(binary\_num))):

if (binary\_num[i] is “0”):

set binary\_num[i] to “1”

elif (binary\_num[i] is “1”):

set binary\_num[i] to “0”

set one\_complement to ““.join(binary\_num)

return one\_complement

function two\_complement(num):

set num to list(num)

if (num[0] is "0"):

return ““.join(num)

else:

for (i in range(length(num))):

if (num[i] is “0”):

set num[i] to “1”

elif (num[i] is “1”):

set num[i] to “0”

set ones\_comp to ““.join(num)

set result to ““

set carry to 1

for (bit in reversed(ones\_comp)):

if (carry is 1):

set sum\_bit to (int(bit) + carry) % 2

set carry to (int(bit) + carry) // 2

set result to str(sum\_bit) + result

else:

set result to bit + result

return result

menu\_1()