Eslam Saeed Hamdy Hassan 20230060

Eslam Waleed Salah AbdElmotaleb 20230062

Youssef Farid Sayed Hassanin 20230504

# Problem 1:

## #menu 1

Prompt the user to enter a letter A or B

Input(query)

If query == ‘A’

## #menu2

Input(num)

Prompt the user to enter a letter A, B, C or D representing the base of the entered num

Input(numbers base)

In case of an invalid input return to menu 1 again

Use function is\_the\_right\_base(num, numbers\_base) to detect if the number is indeed in the base user has entered

## #menu3

Prompt the user to enter a letter A, B, C or D representing the base of the wanted number

In case of invalid input

return to menu3 again

In case of A as an input

If(numbers\_base == wanted\_base)

Print(num) since no transformation happens in this case

else

transfer the number to decimal if it is not already using function convert\_to\_decimal(num, numbers\_base)

transfer the number to the wanted base using function convert\_to\_wanted\_base(num, wanted\_base)

print(num)

else if query == ‘B’

exit

else

return to menu1

function convert\_to\_decimal(num, base)

loop \*from end to start\*

transform the character to an integer

result += num \* (base \*\* (number of the digit – 1))

return result

function is\_the\_right\_base(N, base)

loop

if it’s not the case that 0 =< character <= base

return False

return True

function convert\_to\_wanted\_base(N, base):

N = integer(NONE)

string hex = "0123456789ABCDEF"

string Result = ""

while N != 0:

integer Reminder = N % base

Result = hex[Reminder] + Result

N = integer(N/base)

Return Result

# Problem 2

function first\_complement(binary1):

result = ““

for i from 0 to size(binary1) - 1:

if binary1[i] == “0”:

result += “1”

else:

result += “0”

return result

function second\_complement(binary1):

result = ““

result2 = ““

counter = size(binary1) - 1

count\_ones = 0

while counter >= 0:

if binary1[counter] == “0”:

if count\_ones == 0:

result += “0”

else if count\_ones >= 1:

result += “1”

else if binary1[counter] == “1”:

if count\_ones == 0:

result += “1”

else if count\_ones >= 1:

result += “0”

count\_ones += 1

counter -= 1

for i from 0 to size(result) - 1:

result2 += result[size(result) - 1 - i]

return result2

function is\_binary(binary\_str):

for bit in binary\_str:

if bit not in “01”:

return false

return true

function addition(binary1, binary2):

if not is\_binary(binary2):

output(“Please insert a valid binary number.”)

return

max1 = max(size(binary1), size(binary2))

binary1 = binary1.zfill(max1)

binary2 = binary2.zfill(max1)

binary\_sum1 = ““

binary\_sum2 = ““

carry = 0

for i from 0 to max1 - 1:

if carry == 0:

if integer(binary1[max1 - 1 - i]) == 0 or integer(binary2[max1 - 1 - i]) == 0:

binary\_sum1 += string(integer(binary1[max1 - 1 - i]) + integer(binary2[max1 - 1 - i]))

if integer(binary1[max1 - 1 - i]) == 1 and integer(binary2[max1 - 1 - i]) == 1:

binary\_sum1 += string(0)

carry += 1

else if carry == 1:

if integer(binary1[max1 - 1 - i]) == 0 and integer(binary2[max1 - 1 - i]) == 0:

binary\_sum1 += string(1)

carry = 0

if integer(binary1[max1 - 1 - i]) == 1 and integer(binary2[max1 - 1 - i]) == 0:

binary\_sum1 += string(0)

if integer(binary1[max1 - 1 - i]) == 0 and integer(binary2[max1 - 1 - i]) == 1:

binary\_sum1 += string(0)

if integer(binary1[max1 - 1 - i]) == 1 and integer(binary2[max1 - 1 - i]) == 1:

binary\_sum1 += string(1)

if carry == 1:

binary\_sum1 += string(carry)

for i from 0 to size(binary\_sum1) - 1:

binary\_sum2 += binary\_sum1[size(binary\_sum1) - 1 - i]

return binary\_sum2

function subtraction(binary1, binary2):

if not is\_binary(binary2):

output(“Please insert a valid binary number.”)

return

len\_diff = size(binary1) - size(binary2)

if len\_diff > 0:

binary2 = “0” \* len\_diff + binary2

else if len\_diff < 0:

binary1 = “0” \* abs(len\_diff) + binary1

result = ““

borrow = 0

for i from size(binary1) - 1 to 0 step -1:

bit1 = integer(binary1[i])

bit2 = integer(binary2[i])

temp\_result = bit1 - bit2 - borrow

if temp\_result < 0:

temp\_result += 2

borrow = 1

else:

borrow = 0

result = string(temp\_result) + result

first\_one\_index = find\_first\_one(result)

result = result[first\_one\_index:] if first\_one\_index is not None else “0”

if result == “”  
 return 0

else

return result

Main function:

while true:

output(“\n\*\*binary calculator\*\*”)

output(“A) Insert new numbers”)

output(“B) Exit”)

choice1 = input(“Enter your choice (A/B): “).toUpperCase()

if choice1 == “B”:

break

else if choice1 == “A”:

binary1 = input(“Please insert the first number: “)

if not is\_binary(binary1):

output(“Please insert a valid binary number.”)

continue

output(“\n\*\* please select the operation \*\*”)

output(“A) Compute one”s complement”)

output(“B) Compute two”s complement”)

output(“C) Addition”)

output(“D) Subtraction”)

choice2 = input(“Please enter your choice (A/B/C/D): “).toUpperCase()

if choice2 == “A”:

output(first\_complement(binary1))

if choice2 == “B”:

output(second\_complement(binary1))

if choice2 == “C”:

binary2 = input(“Please insert the second number: “)

output(addition(binary1, binary2))

if choice2 == “D”:

binary2 = input(“Please insert the second number: “)

output(subtraction(binary1, binary2))

if choice2 != “A” and choice2 != “B” and choice2 != “C” and choice2 != “D”:

output(“Please select a valid operation.”)

continue

else if choice1 != “A” and choice1 != “B”:

output(“Please select a valid choice.”)