San Francisco Bay University

CE305 - Computer Organization

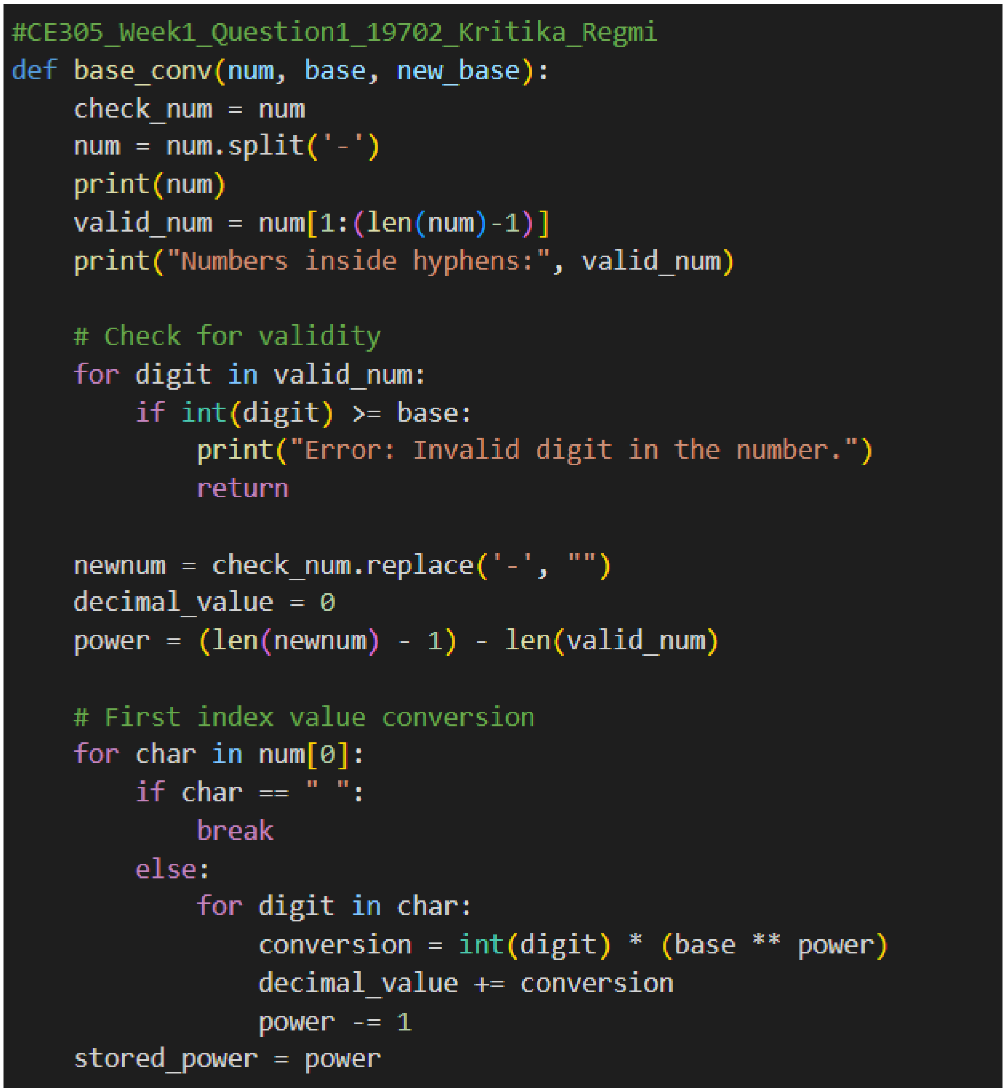
2023 Fall Homework #1

ID:19702
Kritika Regmi

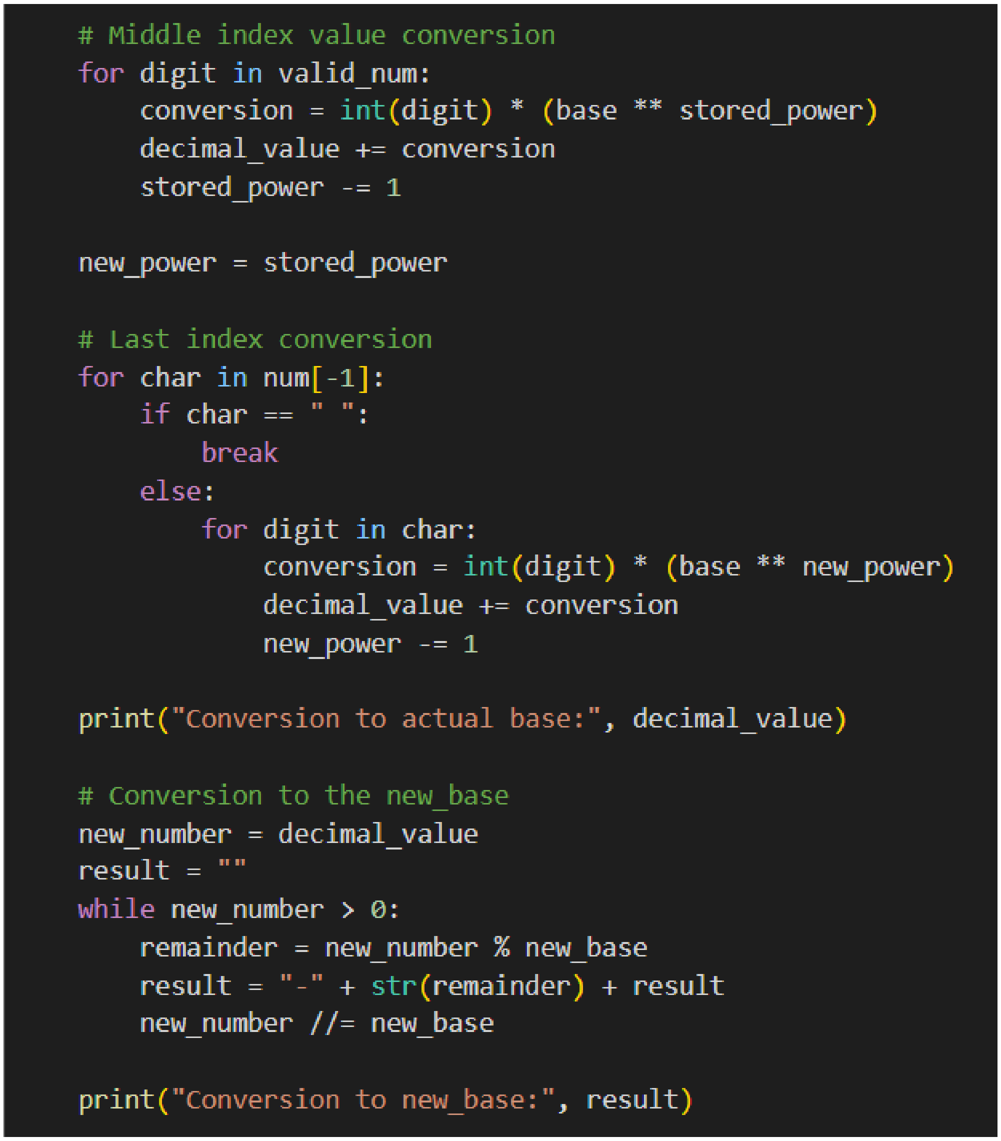
1. Write the program in any computer language to convert the given number from any base to a different base. The program needs to verify the validity of the given number first. If it is invalid, please prompt error information. Otherwise, print the correct result in the new base.

ANSWER:

CODE:

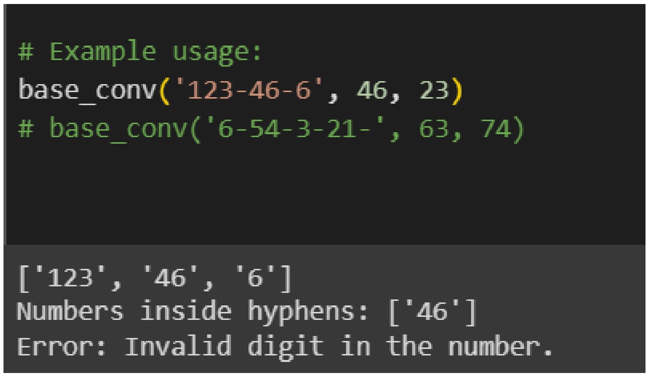


```
#CE305_Week1_Question1_19702_Kritika_Regmi
def base_conv(num, base, new_base):
    check_num = num
    num = num.split('-')
    print(num)
    valid_num = num[1:(len(num)-1)]
    print("Numbers inside hyphens:", valid_num)

    # Check for validity
    for digit in valid_num:
        if int(digit) >= base:
            print("Error: Invalid digit in the number.")
            return

    newnum = check_num.replace('-', "")
    decimal_value = 0
    power = (len(newnum) - 1) - len(valid_num)

    # First index value conversion
    for char in num[0]:
        if char == " ":
            break
        else:
            for digit in char:
                conversion = int(digit) * (base ** power)
                decimal_value += conversion
                power -= 1
    stored_power = power
```



```python
    # Middle index value conversion
    for digit in valid_num:
        conversion = int(digit) * (base ** stored_power)
        decimal_value += conversion
        stored_power -= 1

    new_power = stored_power

    # Last index conversion
    for char in num[-1]:
        if char == " ":
            break
        else:
            for digit in char:
                conversion = int(digit) * (base ** new_power)
                decimal_value += conversion
                new_power -= 1

    print("Conversion to actual base:", decimal_value)

    # Conversion to the new_base
    new_number = decimal_value
    result = ""
    while new_number > 0:
        remainder = new_number % new_base
        result = "-" + str(remainder) + result
        new_number //= new_base

    print("Conversion to new_base:", result)
```

OUTPUT:



```python
# Example usage:
base_conv('123-46-6', 46, 23)
# base_conv('6-54-3-21-', 63, 74)
```

```
['123', '46', '6']
Numbers inside hyphens: ['46']
Error: Invalid digit in the number.
```

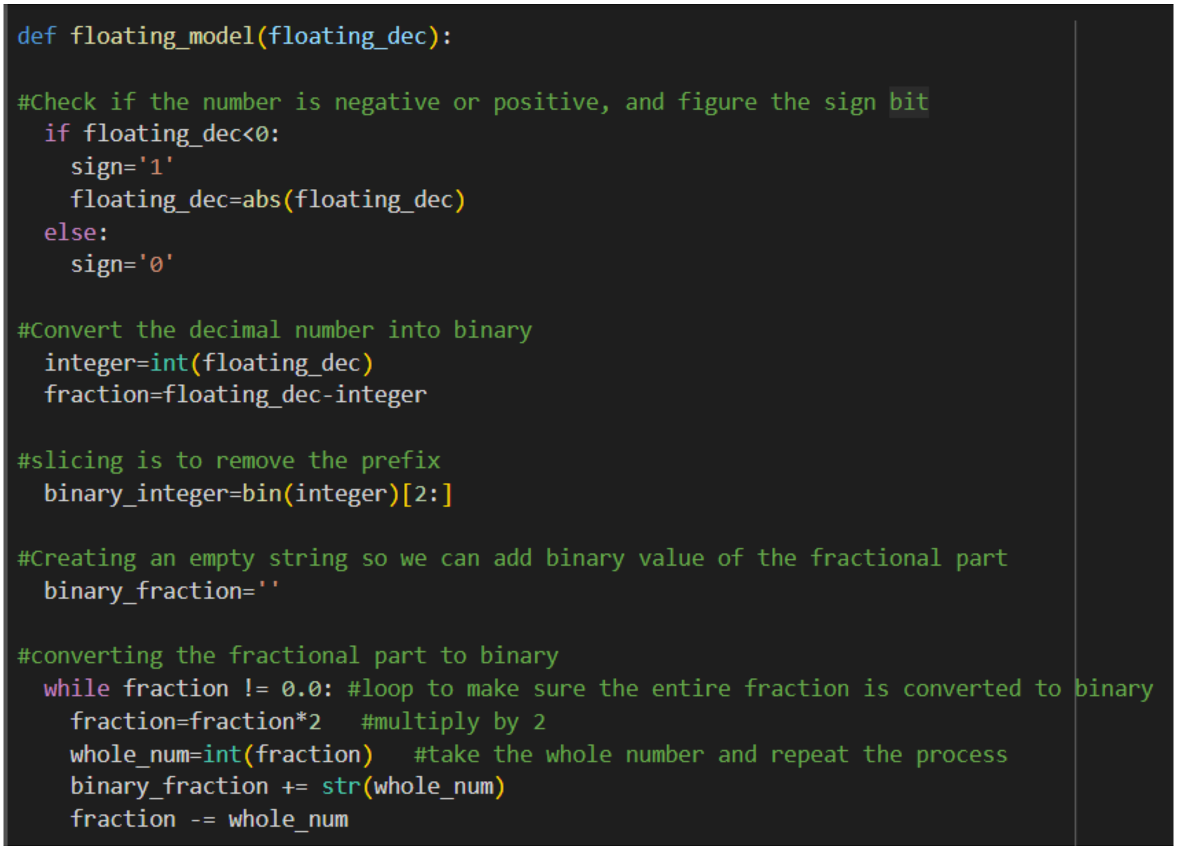
2. Write the program in any computer language to convert the floating decimal number to 14-bits binary floating-point model as the real digital values in the hardware memory. The example -26.625$_{10}$ will be saved in the 14-bits hardware memory shown as follows.

$$26.625_{10} = 11010.101_2 = 0.11010101 \times 2^5$$

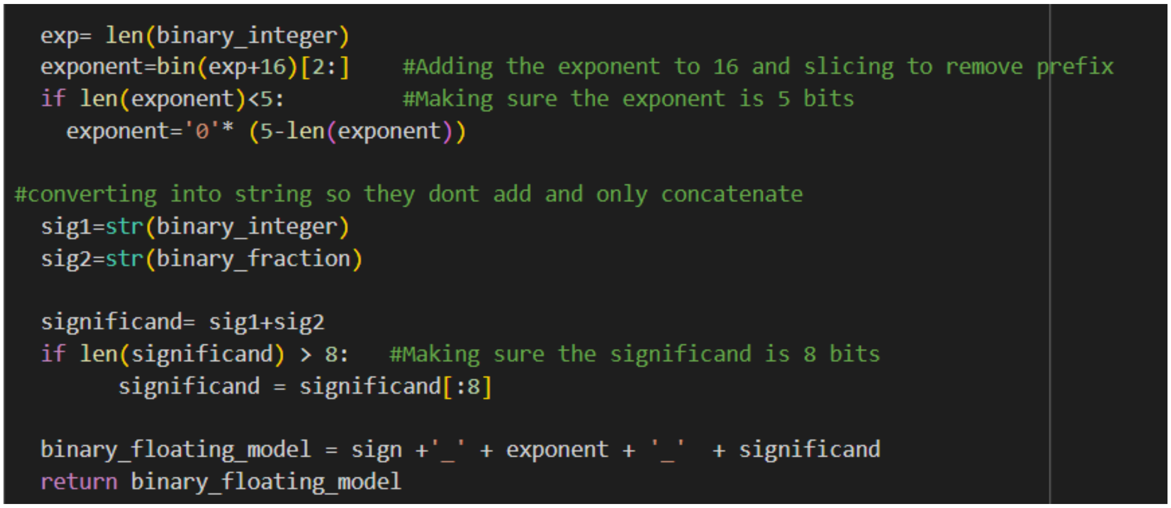
1	1	0	1	0	1	1	1	0	1	0	1	0	1
1: negative 0: positive	5_{10} (power of 2) +16_{10} = 21_{10} = 10101_2					11010101 as significand							
1 bit	5 bits					8 bits							
Total: 14 bits													

ANSWER:

- **CODE:**

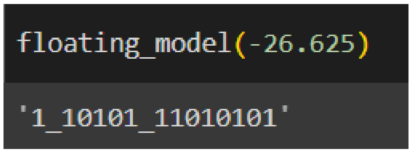


```
def floating_model(floating_dec):

#Check if the number is negative or positive, and figure the sign bit
  if floating_dec<0:
    sign='1'
    floating_dec=abs(floating_dec)
  else:
    sign='0'

#Convert the decimal number into binary
  integer=int(floating_dec)
  fraction=floating_dec-integer

#slicing is to remove the prefix
  binary_integer=bin(integer)[2:]

#Creating an empty string so we can add binary value of the fractional part
  binary_fraction=''

#converting the fractional part to binary
  while fraction != 0.0: #loop to make sure the entire fraction is converted to binary
    fraction=fraction*2   #multiply by 2
    whole_num=int(fraction)   #take the whole number and repeat the process
    binary_fraction += str(whole_num)
    fraction -= whole_num
```



```
  exp= len(binary_integer)
  exponent=bin(exp+16)[2:]    #Adding the exponent to 16 and slicing to remove prefix
  if len(exponent)<5:         #Making sure the exponent is 5 bits
    exponent='0'* (5-len(exponent))

#converting into string so they dont add and only concatenate
  sig1=str(binary_integer)
  sig2=str(binary_fraction)

  significand= sig1+sig2
  if len(significand) > 8:   #Making sure the significand is 8 bits
        significand = significand[:8]

  binary_floating_model = sign +'_' + exponent + '_'  + significand
  return binary_floating_model
```

- **OUTPUT:**



```
floating_model(-26.625)

'1_10101_11010101'
```