San Francisco Bay University

CE305 - Computer Organization
2023 Fall Homework #1

Due day: 9/27/2023

Instruction:

1. **Homework answer sheet should contain the original questions and corresponding answers.**
2. **Answer sheet must be in PDF file format with Github links for the programming questions, but MS Word file can't be accepted. As follows is the answer sheet name format.**
   <course_id>_week<week_number>_StudentID_FirstName_LastName.pdf
3. **The program name in Github must follow the format like**
   <course_id>_week<week_number>_q<question_number>_StudentID_FirstName_LastName
4. **Show screenshot of all running results, including the system date/time.**
5. **Only accept homework submission uploaded via Canvas.**
6. **Overdue homework submission can't be accepted.**
3. **Takes academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

SWEKCHHA HAMAL,19700.

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. For instance, as follows is the *def* function "base_conv" in Python.

```
def base_conv (num, base, new_base):
  """
    conv(123-45-6, 44, 23)    # -45- represents one digit in base 44
    123-45-6 is invalid, since one single bit -45- is bigger than base 44

    conv(123-45-6, 46, 23)
    # 123-45-6 (base 46) = 1*46^4 + 2*46^3 + 3*46^2 + 45*46^1 + 6*46^0 = 4680552 (base 10)
    # 4680552 (base 10) = -16-16-15-21-6 (base 23) = 16*23^4 + 16*23^3 + 15*23^2 + 21*23^1 + 6*23^0

    -16-16-15-21-6  (base 23)     # Final answer with 5 digits for base 23

    conv(6-54-3-21-, 46, 23)    # -54- and -21- represent one digit in base 46 respectively
    6-54-3-21- is invalid, since one single bit -54- is bigger than base 46

    conv(6-54-3-21-, 63, 74)
```

6-54-3-21- (base 63) = 6*63^3+54*63^2+3*63^1+21*63^0 = 1714818 (base 10)
1714818 (base 10) = 4-17-11-16- (base 74) = 4*74^3 + 17*74^2 + 11*74^1 + 16*74^0

4-17-11-16- (base 74) # Final answer with 4 digits for base 74

"""

ANS:

```
def base_conv(num,base,new_base):
  check_num = num
  num = num.split('-')
  print(num)
  valid_num = num[1:(len(num)-1)]
  print("No inside hyphen",valid_num)
  #Check for validity
  for i in valid_num:
    if int(i) > int(base):
      print("Error information")
  else:
      newnum = check_num.replace('-',"")
      bc = 0
      power = (len(newnum)-1)-len(valid_num)


    #first index value conversion
      for i in num[0]:
        if i == " ":
          break
        else:
         for j in i:
           conv = int(j) * (base**power)
           bc = conv + bc
           power-=1
      stored_power = power
  #middle index value conversion
      for i in valid_num:
        conv = int(i)*(base**stored_power)
        bc = conv + bc
        stored_power -= 1

      new_power = stored_power
```

```
  #last index conversion
      for i in num[-1]:
        if i==" ":
          break
        else:
         for j in i:
           conv = int(j) * (base**new_power)
           bc = conv + bc
           new_power-=1
      print("Conversion to actual base:",bc)
  #Conversion to the new_base
  newnumber= bc
  result = ""
  while newnumber > 0:
    remainder = newnumber % new_base
    result = "-"+ str(remainder)  + result
    newnumber //= new_base

  print("Conversion to new_base:",result)









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

OUTPUT:

```
['123', '46', '6']
No inside hyphen ['46']
Conversion to actual base: 4680598
Conversion to new_base: -16-16-16-0-6
```

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: 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		

def floating_model(floating_dec):
"""
floating_model(-26.625)

1_10101_11010101

"""

ANS:

```
def float_to_binary_floating_point(floating_dec):
    if floating_dec == 0:
        return "1_00000_00000000"

    sign_bit = "1" if floating_dec < 0 else "0"
    abs_floating_dec = abs(floating_dec)

    int_part = int(abs_floating_dec)
    frac_part = abs_floating_dec - int_part

    int_binary = bin(int_part)[2:]
    frac_binary = ""

    while len(int_binary) < 5:
        int_binary = "0" + int_binary

    for _ in range(8):
        frac_part *= 2
        frac_bit = "1" if frac_part >= 1 else "0"
        frac_binary += frac_bit
        if frac_part >= 1:
            frac_part -= 1

    binary_representation = f"{sign_bit}_{int_binary}_{frac_binary}"
    return binary_representation[:14]

floating_dec = -26.625
binary_representation = float_to_binary_floating_point(floating_dec)
print(binary_representation)
```

OUTPUT:

```
1_11010_101000
```