Problem #1

Subject: Hack Disassembler

Filename: hw02_01.py
Allowed modules: NONE

Write a program that opens asks the user for the name of a Hack machine code file (WITHOUT the
extension). Open the file and read the contents. Output a disassembled code listing, to the console, in a
format similar to the above (the original machine encoding, some space, and the disassembled
instruction).
Do NOT make this harder than it is! You do not need to know what any of these instructions do, you are
merely translating patterns of ones and zeroes into corresponding strings of text.
You may read the file a line at a time or all at once, it is up to you.
Use a separate dictionary for the three fields of the C-type instruction.

WORK

First, a dictionary will need to be made for comp when a=0, comp when a=1, dest, and jump. Then the user will enter the name of a file (without the extension) containing lines each with 16 digits (0s and 1s). This file will then need to be read and converted to assembly code using the dictionaries mentioned above.

OUTPUT

DATA ENTRY

Enter file name (Without extension): hackman

DISASSEMBLED OUTPUT

```
0000000000000000       @0

1110101010001000      M=0

1110111111000000        1

1110111010000000       -1

1110001100100001    A=D;JGT

1110110000010000      D=A

1111110000000000        M

1110001101000000       !D
```

```
1110110001000010     !A;JEQ
0111111111111111     @32767
1111110001000000       !M
1110001111000000       -D
1110110011110000     AD=-A
1111110011000000       -M
1110011111000000       D+1
1110110111000000       A+1
0000000011111111      @255
1111110111100011   A=M+1;JGE
1110001110000000       D-1
1110110010000000       A-1
1111110010101000     AM=M-1
1110000010000000       D+A
1111000010000100    D+M;JLT
1110010011111000    AMD=D-A
1111010011000000       D-M
1110000111000101    A-D;JNE
1111000111101110  AM=M-D;JLE
1110000000000000       D&A
1111000000000000       D&M
1110010101000000       D|A
1111010101000111    D|M;JMP
0100000000000001     @16385
```

CODE

```
'''

PROGRAMMER:..Christopher Colbert

USERNAME:....ccolbert

PROGRAM:.....hw02_01.py


DESCRIPTION: Takes in the input of a file name (Without extension). Open and read the given file, outputs the original file

input with some space and then the dissasembled machine instruction

'''

#setup comp instruction dictionaries

comp_0 = {"101010": "0","111111": "1","111010": "-1","001100": "D","110000": "A", "001101": "!D", "110001": "!A", "001111": "-D","110011": "-A",

        "011111": "D+1", "110111":"A+1","001110": "D-1","110010": "A-1", "000010": "D+A","010011": "D-A", "000111": "A-D","000000": "D&A","010101": "D|A"}

comp_1 = {"101010": "","111111": "","111010": "","001100": "","110000": "M", "001101": "", "110001": "!M", "001111": "","110011": "-M",

        "011111": "", "110111":"M+1","001110": "","110010": "M-1", "000010": "D+M","010011": "D-M", "000111": "M-D","000000": "D&M","010101": "D|M"}

dest = {"000": "", "001": "M","010": "D","011": "MD","100": "A","101": "AM","110": "AD","111": "AMD"}

jump = {"000": "","001": "JGT","010": "JEQ","011": "JGE","100": "JLT","101": "JNE","110": "JLE","111": "JMP"}


#obtain file name from user

print("DATA ENTRY")

file_name = (input("Enter file name (Without extension): "))

file_name = file_name + ".hack"
```

```
print("DISASSEMBLED OUTPUT")

with open(file_name, "r") as file:
  #for each line in file
  for line in file:
    #remove newline character from each line
    line = line.strip()
    #if first digit is 0, convert to decimal
    if line[0] == "0":
      output = "@" + str(int(line[1:], 2))

    #if first 3 digits are "111"
    elif line[0:3] == "111":
      a = line[3]
      comp_bits = line[4:10]
      dest_bits = line[10:13]
      jump_bits = line[13:]

      comp_dict = comp_0 if line[3] == "0" else comp_1
      comp_mnemonic = comp_dict[comp_bits]
      dest_mnemonic = dest[dest_bits]
      jump_mnemonic = jump[jump_bits]
      #normal output: comp=dest;jump
      if dest_mnemonic and jump_mnemonic:
          output = dest_mnemonic + "=" + comp_mnemonic + ";" + jump_mnemonic
      #if jump is empty output: comp=dest
      elif dest_mnemonic:
          output = dest_mnemonic + "=" + comp_mnemonic
      #if dest is missing output: comp;jump
      elif jump_mnemonic:
          output = comp_mnemonic + ";" + jump_mnemonic
      #if dest and jump are missing output: comp
      else:
          output = comp_mnemonic

    print("%16s        %10s" % (line, output))
```

Problem #2

Subject: Proper mortgage payment

Filename: hw02_02.py
Allowed modules: none

Write a program that asks the user for a loan amount, an APR, and a term (in years) and produces a
report (to the console) stating the loan terms, including the monthly payment and the amount of the
final payment. It should also provide the total amount that the borrow will pay over the life of the loan
and the cost of credit (i.e., the total amount of interest paid).
Run your program for a 30-year loan of $388,000 at an APR of 2.5% and also at an APR of 7%.

WORK

I will need to first convert my program from HW01_01 into a function that returns the residual balance. I will then need to create a function that finds the monthly payment using the loan amount, loan APR, and loan term length. Using the equation:

$$M = P * \frac{r(1+r)^n}{r(1+r)^n - 1}$$

Where M = the monthly payment, P = principle payment, r = monthly interest rate, and n = number of payments.

I will then need to output the all the given and derived information.

<u>OUTPUT</u>

```
DATA ENTRY
Enter loan amount ($): .. 388000
Enter loan APR (%): ..... 7
Enter loan term (yrs): .. 30
MORTGAGE TERMS
Loan Amount: ......... $ 388000.00
Loan Rate: ...........      7.000%
Loan Term: ...........       30 years
Monthly Payment: ..... $   2581.37
Residual Balance: .... $      0.10
Final Amount: ........ $   2581.27
Total Payment: ....... $ 929294.53
Cost of Credit: ...... $ 541294.53
```

<u>CODE</u>

```
'''
PROGRAMMER:..Christopher Colbert
USERNAME:....ccolbert
PROGRAM:.....hw02_02.py

DESCRIPTION: Takes user input of loan amount, loan APR, and loan term length and
calculates monthly payment, final month payment, total amount paid, and cost of credit
'''
```

```
def mortgage_residual(loan_amount, loan_apr, loan_term, monthly_payment):
   #Convert APR to monthly interest and monthly payment amount
   monthly_interest_rate= (loan_apr/100)/12
   remaining_balance=loan_amount

   #for each month
   for month in range(1, loan_term*12 + 1):
      #subtracting interest from payment
      monthly_interest_amount= round(remaining_balance * monthly_interest_rate, 2)
      principal_payment = monthly_payment - monthly_interest_amount

      #take payment - interest out of balance
      remaining_balance -= principal_payment
   return remaining_balance
#end mortgage_residual function

def mortgage_payment(oan_amount, loan_apr, loan_term):
   monthly_interest_rate= (loan_apr/100)/12

   monthly_payment = loan_amount * (monthly_interest_rate * (1 + monthly_interest_rate)
** (loan_term * 12)) / ((1 + monthly_interest_rate) ** (loan_term * 12) - 1)

   final_payment = monthly_payment * (1 + monthly_interest_rate) ** (loan_term * 12) -
loan_amount

   return monthly_payment, final_payment
#end mortgage_payment function
```

```
#Main program

print("DATA ENTRY")

loan_amount = float(input("Enter loan amount ($): .. "))

loan_apr = float(input("Enter loan APR (%): ..... "))

loan_term = int(input("Enter loan term (yrs): .. "))


monthly_payment, final_payment = mortgage_payment(loan_amount, loan_apr,
loan_term)

residual_balance = mortgage_residual(loan_amount, loan_apr, loan_term,
monthly_payment)


#Print results

print("MORTGAGE TERMS")

print("Loan Amount: ......... $%10.2f" % loan_amount)

print("Loan Rate: ........... %11.3f%%" % loan_apr)

print("Loan Term: ........... %11d years" % loan_term)

print("Monthly Payment: ..... $%10.2f" % monthly_payment)

print("Final Amount: ........ $%10.2f" % (monthly_payment + residual_balance))

print("Total Payment: ....... $%10.2f" % (monthly_payment * loan_term * 12))

print("Cost of Credit: ...... $%10.2f" % (monthly_payment * loan_term * 12 - loan_amount))
```

Problem #3

Subject: Extract assembly code from disassembler output

Filename: hw02_03.py
Allowed modules: re

Capture the output from your Problem #2 code (or have your script for that problem write it to a file,
your choice) named 'hackman.dis'.
Write a program that asks the user for a disassembler output file (w/o extension), reads the file (.dis
extension), and produces an assembler file (.asm extension) with just the assembly code. The code
should also be echoed to the console.
Run your program on your hackman.dis file.

WORK

Using the code given in the problem, I just need to find a regular expression to remove the 16 digits, and the remaining whitespace, and capture the remaining characters (the assembly instructions)

\d+ matches one or more digits

\s+ matches one or more whitespace characters

.* matches the remaining characters on the line excluding the newline character

() around the '.*' should capture this part of the regular expression

OUTPUT

```
DATA ENTRY

Enter file name (Without extension): hackman

@0

M=0

1

-1

A=D;JGT

D=A

M
```

```
!D
!A;JEQ
@32767
!M
-D
AD=-A
-M
D+1
A+1
@255
A=M+1;JGE
D-1
A-1
AM=M-1
D+A
D+M;JLT
AMD=D-A
D-M
A-D;JNE
AM=M-D;JLE
D&A
D&M
D|A
D|M;JMP
@16385
```

<u>CODE</u>

```
'''
PROGRAMMER:..Christopher Colbert
USERNAME:....ccolbert
PROGRAM:.....hw02_03.py

DESCRIPTION: Obtains file name (without extension) from user containing lines of:
machine_code                    assembly_code
Then creates a file just the assembly code on each line. File output is echoed to the console
'''
import re

print("DATA ENTRY")
file_name = (input("Enter file name (Without extension): "))
fp = open(file_name + ".dis", "rt")
data = fp.read()
fp.close()

#takes expressions of any number of digits, any number of whitespace, and then captures any remaining
#characters on the line (excluding '\n')
regex = re.compile(r'\d+\s+(.*)')

results = regex.findall(data)
```

```
fp = open(file_name + ".asm", "wt")
for item in results:
   print(item)
   fp.write(item + "\n")
fp.close()
```