



You can view this report online at : <https://www.hackerrank.com/x/tests/1731513/candidates/57286188/report>

Full Name:	Waqar Saleem
Email:	waqar.saleem@sse.habib.edu.pk
Test Name:	CS101 - HW1 - Fall23
Taken On:	21 Oct 2023 06:14:22 PKT
Time Taken:	77 min 1 sec/ 20160 min
Invited by:	Aisha
Skills Score:	
Tags Score:	<div>CS101 40/40</div> <div>Input 20/20</div> <div>Lists 20/20</div> <div>Medium 20/20</div> <div>Output 20/20</div> <div>Python 3 40/40</div> <div>Sequence 20/20</div>

**100%****120/120**

scored in **CS101 - HW1 - Fall23**  
in 77 min 1 sec on 21 Oct 2023  
06:14:22 PKT

**Candidate Feedback:**

A more complete Emacs mode. Make all text readable in dark mode.

**Recruiter/Team Comments:**

No Comments.

Question Description	Time Taken	Score	Status
Q1 <b>Greedy ATM</b> > Coding	14 min 9 sec	20/ 20	✓
Q2 <b>Multiplication Table</b> > Coding	8 min 44 sec	20/ 20	✓
Q3 <b>Collatz Conjecture - Recursive</b> > Coding	2 min 22 sec	20/ 20	✓
Q4 <b>Generatus Emailius</b> > Coding	18 min 5 sec	20/ 20	✓
Q5 <b>The Babington Plot</b> > Coding	21 min 58 sec	20/ 20	✓
Q6 <b>Difficulty Meter</b> > Multiple Choice	15 sec	5/ 5	✓
Q7 <b>Feedback (Graded)</b> > Subjective	2 min 37 sec	15/ 15	✓

**QUESTION 1**

Correct Answer

**Greedy ATM** > Coding 

CS101

Python 3

**QUESTION DESCRIPTION**

Write a function `make_change()` that takes an `amount` as a parameter and uses a *greedy* approach to find the minimum number of notes of different denominations that sum up to `amount`. The greedy approach is to choose as many notes as possible of a higher denomination before choosing any note of a lower denomination.

The available denominations are 5000, 1000, 500, 100, 50, 20, 10.

### Example

The greedy algorithm proceeds as follows for an amount of 7,860.

1. Select as many bills as possible of the highest denomination (1 x 5,000). Calculate the remaining amount (7,860 - 5,000 = 2,860).
2. Select as many bills as possible of the next denomination (2 x 1,000). Calculate the remaining amount (2,860 - 2,000 = 860).
3. Select as many bills as possible of the next denomination (1 x 500). Calculate the remaining amount (860 - 500 = 360).
4. Select as many bills as possible of the next denomination (3 x 100). Calculate the remaining amount (360 - 300 = 60).
5. Select as many bills as possible of the next denomination (1 x 50). Calculate the remaining amount (60 - 50 = 10).
6. Select as many bills as possible of the next denomination (0 x 20). Calculate the remaining amount (10 - 0 = 10).
7. Select as many bills as possible of the last denomination (1 x 10).

### Constraints

- `isinstance(amount, int)`
- `amount >= 0`
- `amount % 10 == 0`

#### ▼ Input Format For Custom Testing

The first (and only) line contains an integer that specifies the rupee `amount`.

You must take input.

#### ▼ Sample Case 0

##### Sample Input For Custom Testing

```
7860
```

##### Sample Output

```
5000: 1
1000: 2
500: 1
100: 3
50: 1
20: 0
10: 1
```

### Explanation

Each line specifies a denomination, from highest (5000) to lowest (10) in order, followed by a colon (:) character, followed by the number of bills.

$$7860 = (1 \times 5000) + (2 \times 1000) + (1 \times 500) + (3 \times 100) + (1 \times 50) + (0 \times 20) + (1 \times 10)$$

#### ▼ Sample Case 1

##### Sample Input For Custom Testing

```
3730
```

### Sample Output

```
5000: 0
1000: 3
500: 1
100: 2
50: 0
20: 1
10: 1
```

### Explanation

Each line specifies a denomination, from highest (5000) to lowest (10) in order, followed by a colon (:) character, followed by the number of bills.

$$1730 = (0 \times 5000) + (3 \times 1000) + (1 \times 500) + (2 \times 100) + (0 \times 50) + (1 \times 20) + (1 \times 10)$$

### INTERVIEWER GUIDELINES

```
def make_change(amount):
    print('5000:', amount // 5000)
    amount %= 5000
    print('1000:', amount // 1000)
    amount %= 1000
    print('500:', amount // 500)
    amount %= 500
    print('100:', amount // 100)
    amount %= 100
    print('50:', amount // 50)
    amount %= 50
    print('20:', amount // 20)
    amount %= 20
    print('10:', amount // 10)
```

### CANDIDATE ANSWER

Language used: **Python 3**

```
1 # Write your code here.
2 def make_change(amount):
3     denomination = 5000
4     notes = amount // denomination
5     print(denomination, ': ', notes, sep='')
6     amount %= denomination
7     denomination = 1000
8     notes = amount // denomination
9     print(denomination, ': ', notes, sep='')
10    amount %= denomination
11    denomination = 500
12    notes = amount // denomination
13    print(denomination, ': ', notes, sep='')
14    amount %= denomination
15    denomination = 100
16    notes = amount // denomination
17    print(denomination, ': ', notes, sep='')
18    amount %= denomination
19    denomination = 50
20    notes = amount // denomination
21    print(denomination, ': ', notes, sep='')
22    amount %= denomination
23    denomination = 20
```

```

24     notes = amount // denomination
25     print(denomination, ': ', notes, sep='')
26     amount %= denomination
27     denomination = 10
28     notes = amount // denomination
29     print(denomination, ': ', notes, sep='')
30     amount %= denomination
31
32 amount = int(input())
33

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2	0.0401 sec	9.88 KB
Testcase 1	Easy	Sample case	✔ Success	2	0.0492 sec	9.76 KB
Testcase 2	Easy	Sample case	✔ Success	2	0.0617 sec	9.68 KB
Testcase 3	Easy	Sample case	✔ Success	2	0.0747 sec	9.43 KB
Testcase 4	Easy	Sample case	✔ Success	2	0.0592 sec	9.74 KB
Testcase 5	Easy	Hidden case	✔ Success	2	0.0472 sec	9.55 KB
Testcase 6	Easy	Hidden case	✔ Success	2	0.055 sec	9.73 KB
Testcase 7	Easy	Hidden case	✔ Success	2	0.039 sec	9.63 KB
Testcase 8	Easy	Hidden case	✔ Success	2	0.0452 sec	9.49 KB
Testcase 9	Easy	Hidden case	✔ Success	2	0.0939 sec	9.5 KB

No Comments

## QUESTION 2



Correct Answer

Score 20

## Multiplication Table > Coding

### QUESTION DESCRIPTION

### Problem

Your younger cousin is having trouble learning her multiplication tables so you have decided to write her a function to display the tables. Write a function named `multiplication_table` that takes integer parameters named `n` and `k` and prints the multiplication table of `n` till `n * k` as shown below. To show off your programming skills, you have decided to write the function *recursively*.

### Sample

```

>>> multiplication_table(5,10)
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50

```

### Constraints

- `isinstance(n, int)`
- `isinstance(k, int)`

- `n > 0` and `k > 0`

## Interaction

The input comprises a single line containing two integers denoting the values of `n` and `k` respectively. You need not take input.

## Note

To be considered for grading, the function must be recursive.

### INTERVIEWER GUIDELINES

#### Solution

```
def multiplication_table(n, k):  
    # recursively  
    if k > 0:  
        multiplication_table(n, k-1)  
        print(n, '*', k, '=', n*k)
```

### CANDIDATE ANSWER

Language used: **Python 3**

```
1 # Enter your code here.  
2 def multiplication_table(n, k):  
3     if k > 0:  
4         multiplication_table(n, k-1)  
5         print(n, '*', k, '=', n * k)  
6
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Medium	Sample case	✔ Success	5	0.0573 sec	11.7 KB
Testcase 1	Medium	Hidden case	✔ Success	5	0.1022 sec	11.4 KB
Testcase 2	Medium	Hidden case	✔ Success	5	0.1267 sec	11.8 KB
Testcase 3	Medium	Hidden case	✔ Success	5	0.1242 sec	11.4 KB

No Comments

#### QUESTION 3



Correct Answer

Score 20

## Collatz Conjecture - Recursive > Coding

Medium

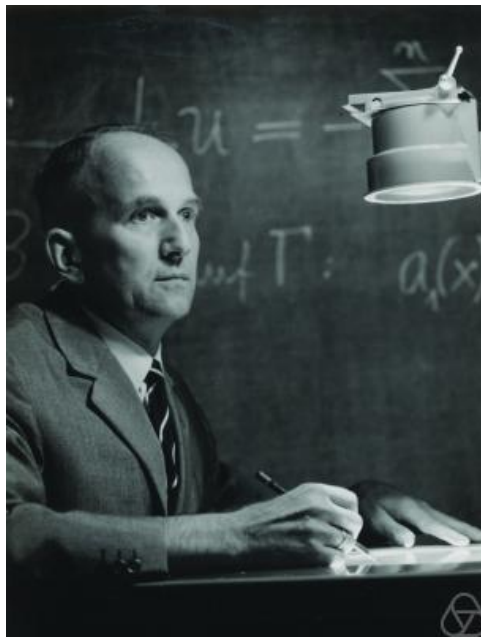
Lists

Sequence

Output

Input

### QUESTION DESCRIPTION



[Wikipedia](#)

In 1937, Lothar Collatz, a German mathematician, put forth the following conjecture: if you start with any positive integer, and you half the integer if it is even, or triple it and add one to it if it is odd, and you keep doing that, you will eventually arrive at one. [The conjecture](#) has, to date, not been proven to be true or false. However, it has been verified up to a very large integer.

We want to verify this conjecture by applying the above algorithm to various numbers and verifying that the algorithm ultimately generates 1.

Given  $n = 5$ , the generated numbers are as follows.  $5 * 3 + 1 = 16$ ,  $16 / 2 = 8$ ,  $8 / 2 = 4$ ,  $4 / 2 = 2$ ,  $2 / 2 = 1$ .

#### Function Description

Write the recursive function

`verify_collatz()` that takes an integer  $n$  as parameter and prints the sequence corresponding to  $n$ . The sequence ends with 1.

#### Constraints

- `(isinstance(n, int) and  $n > 0$ )`

#### ▼ Input Format For Custom Testing

The first line contains  $n$ . Your function must not read input.

The output must contain the sequence of numbers starting from  $n$  and ending with 1. Each number is printed on a new line.

#### ▼ Sample Case 0

##### Sample Input For Custom Testing

1

##### Sample Output

1

##### Explanation

$n$  is 1 so the algorithm does not generate any further numbers.

#### ▼ Sample Case 1

##### Sample Input For Custom Testing

52

##### Sample Output

52  
26  
13  
40

20  
10  
5  
16  
8  
4  
2  
1

### Explanation

52 is even so it is halved to 26. 26 is halved to 13. 13 is odd so it leads to 40. 40 leads to 20 which is halved to 10 which is halved to 5. 5 is odd so leads to 16. 16 is even and generates only even numbers whose successive halving eventually leads to 1.

### INTERVIEWER GUIDELINES

### Solution

```
# Define function.
def verify_collatz(n):
    print(n)
    if n > 1:
        if n % 2 == 0:
            verify_collatz(n//2)
        else:
            verify_collatz(3*n + 1)
```

### CANDIDATE ANSWER

Language used: **Python 3**

```
1
2 # Define function.
3 def verify_collatz(n):
4     print(n)
5     if n > 1:
6         if n % 2 == 0:
7             verify_collatz(n // 2)
8         else:
9             verify_collatz(3*n + 1)
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Sample case	✔ Success	2	0.0999 sec	11.4 KB
Testcase 1	Easy	Sample case	✔ Success	2	0.0966 sec	11.7 KB
Testcase 2	Easy	Sample case	✔ Success	2	0.1551 sec	11.6 KB
Testcase 3	Easy	Sample case	✔ Success	2	0.0668 sec	11.6 KB
Testcase 4	Easy	Sample case	✔ Success	2	0.1165 sec	11.4 KB
Testcase 5	Easy	Hidden case	✔ Success	2	0.0789 sec	11.4 KB
Testcase 6	Easy	Hidden case	✔ Success	2	0.1352 sec	11.6 KB
Testcase 7	Easy	Hidden case	✔ Success	2	0.1095 sec	11.6 KB
Testcase 8	Easy	Hidden case	✔ Success	2	0.1222 sec	11.6 KB
Testcase 9	Easy	Hidden case	✔ Success	2	0.0805 sec	11.4 KB

## QUESTION 4



Correct Answer

Score 20

## Generatus Emailius &gt; Coding

## QUESTION DESCRIPTION

## Problem



Credit:  
[Wikipedia](#)

As an intern at the Incantation Technology (IT) department at Hogwarts University (HU), you are given the exciting task of generating email ID's for the incoming batch of students. You have not quite gotten the hang of your chants yet but luckily your programming-fu is strong. It takes you no time to write a function or two to generate the ID's. As far as results go, that is as good a magic as any!

An ID is generated from a name as follows. There is a running serial number starting from 0. The ID begins with 2 lower case letters representing the first and last initials. The 2 letters are followed by five digits representing the serial number. The ID ends with "@st.hu.edu.pk". Once an ID is generated, the serial number is incremented for the next ID.

Write a function named `generate_ids` that takes a parameter `n`. The function inputs `n` names and prints the ID for each.

## Hint

- You can drag the divider in this window to expand this text area.
- You can find useful string methods at [Python String Methods](https://www.w3schools.com/python/python_ref_string.asp).

[https://www.w3schools.com/python/python\\_ref\\_string.asp](https://www.w3schools.com/python/python_ref_string.asp){Python String Methods}.

## Sample

```
>>> generate_ids(3)
Harry Potter          # input
hp00000@st.hu.edu.pk  # output
Hermione Granger      # input
hg00001@st.hu.edu.pk  # output
Tom Marvolo Riddle    # input
tr00002@st.hu.edu.pk  # output
>>> generate_ids(4)
Waqar Saleem Dumbledore # input
wd00000@st.hu.edu.pk    # output
Umair Azfar Khan        # input
uk00001@st.hu.edu.pk    # output
Shahid Snape Hussain    # input
sh00002@st.hu.edu.pk    # output
Shah Jamal Alam         # input
sa00003@st.hu.edu.pk    # output
```

## Input Format

The input contains `n` followed by `n` names on separate lines. Each name is a `str` containing at minimum a non-empty first name and a non-empty last name separated by space characters. There may also be space characters before or after the name.

## Output

Your code should produce `n` lines of output with line `i` containing the ID of the `i`-th name.

## Constraints

- `isinstance(n, int)` and `n > 0`



```
# Generate email ID.
def generate_ids(n):
    suffix = '@st.hu.edu.pk'
    for i in range(n):
        name = input().strip()
        initials = get_initials(name)
        num = str(i)
        num = '0'*(5-len(num)) + num
        print(initials + num + suffix)

def get_initials(name):
    idx = name.rfind(' ') # index of last space.
    second_initial = name[idx+1]
    return (name[0] + second_initial).lower()

n = int(input())
generate_ids(n)
```

CANDIDATE ANSWER

Language used: **Python 3**

```
1 # Enter your code here.
2 def generate_ids(n):
3     for i in range(n):
4         name = input().strip()
5         initial = name[0]
6         index = name.rfind(' ')
7         initial += name[index+1]
8         num = str(i)
9         num = '0' * (5 - len(num)) + num
10        print(initial.lower() + num + '@st.hu.edu.pk')
11
12 n = int(input())
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 0	Easy	Hidden case	✔ Success	3.4	0.0948 sec	9.42 KB
Testcase 1	Easy	Sample case	✔ Success	3	0.0545 sec	9.25 KB
Testcase 2	Easy	Sample case	✔ Success	3.4	0.0379 sec	9.27 KB
Testcase 3	Easy	Hidden case	✔ Success	3.4	0.0472 sec	9.41 KB
Testcase 4	Easy	Hidden case	✔ Success	3.4	0.0538 sec	9.63 KB
Testcase 5	Easy	Hidden case	✔ Success	3.4	0.1452 sec	9.55 KB

No Comments

QUESTION 5



Correct Answer

The Babington Plot > Coding

CS101

Python 3

QUESTION DESCRIPTION

Background

In the 16th century, Queen Mary Stuart of Scotland conspired to have her cousin, Queen Elizabeth I of England, assassinated in order to ascend the English throne. In her messages leading up to the planned assassination, Queen Mary used an encoding in order to hide the content in case of interception. In the encoding, each letter in the original message was substituted with another letter.

One such encoding scheme and its application is shown below:

Original	Encoding
A	K
B	L
C	M
D	N
E	O
F	P
G	Q
H	R
I	S
J	T
K	U
L	V
M	W
N	X
O	Y
P	Z
Q	A
R	B
S	C
T	D
U	E
V	F
W	G
X	H
Y	I
Z	J

Message	MEETING UNDER THE BRIDGE AT MIDNIGHT
---------	--------------------------------------

Encoded	WOODSXQ EXNOB DRO LBSNQO KD WSNXSQRD
---------	--------------------------------------

Code	LOSXQ PYVVYGVON MKXMOV WOODSXQ
Decoded	BEING FOLLOWED CANCEL MEETING

The scheme above uses a key with a value of 10. That is, the encoding of a letter is the letter that is 10 positions ahead of it in circular alphabetical order. And the decoding is the letter that is 10 positions behind.

### Task

Given text and a key, write a program that either encodes or decodes text using the given key.

### Function Description

To implement the given task, write the following functions:

1. **encode** that takes two argument: **message** of type *str*, **key** of type *int*. The function should **return** a string, which is the encoding obtained from the message using the given key. Letters may be mixed case, but the encoded message will have all capital letters. Any characters other than letters of the alphabet are unchanged.
2. **decode** that takes two argument: **code** of type *str*, **key** of type *int*. The function should **return** a string, which is the original message obtained from the code using the given key. Letters may be mixed case, but the original message must have all capital letters. Any characters other than letters of the alphabet are unchanged.

### Hint

You may find it helpful to use the built-in functions, `ord()` and `chr()`.

### Constraints

- `isinstance(message, str)` and `isinstance(code, str)` and `isinstance(key, int)`
- `key > 0` and `len(message) > 0` and `len(code) > 0`

You may not use any modules or helper functions.

### ▼ Input Format For Custom Testing

The first line contains *selector*, an int, whose value is either 1 or 2.

The second line contains *text*, a str. It contains either the message to encode, or code to decode.

The third line contains *key*, an int. It is the number of positions to move ahead for encoding, or behind for decoding.

You need not take input.

The output must be in upper case and leave any special characters unchanged.

### ▼ Sample Case 0

#### Sample Input For Custom Testing

```
1
Meeting under the bridge at midnight.
10
```

#### Sample Output

```
WOODSXQ EXNOB DRO LBSNQO KD WSNXSQRD.
```

#### Explanation

First line contains "1", so the text is message, and must be encoded. Second line contains the message. Third line contains the key, which is 10.

Each letter must be shifted ten positions **forward** in circular alphabetical order. For example, 10 letters

after M is W, 10 after before E is O, ..., 10 letters after T is D (T to Z is six letters, then circling around, we move four more letters forward from A to arrive at D).

### ▼ Sample Case 1

#### Sample Input For Custom Testing

```
2
LOSXQ PYVVYGONR MKXMOV WOODSXQ
10
```

#### Sample Output

```
BEING FOLLOWEDH CANCEL MEETING
```

#### Explanation

First line contains "2", so the text is code, and must be decoded. Second line contains the code. Third line contains the key, which is 10.

Each letter must be shifted ten positions back in circular alphabetical order. For example, 10 letters before L is B, 10 letters before Q is E, ..., 10 letters before G is W (G to A is six letters, then circling back, we move four more letters back from Z to arrive at W).

#### INTERVIEWER GUIDELINES

```
def encode(message, key):
    message = message.upper()
    code = ''
    alphabet = ord('Z') - ord('A') + 1
    for letter in message:
        if letter.isalpha():
            code += chr((ord(letter) - ord('A') + key) % alphabet +
ord('A'))
        else:
            code += letter
    return code

def decode(code, key):
    return encode(code, - key)
```

#### CANDIDATE ANSWER

Language used: **Python 3**


```
1 def encode(message, key):
2     cipher = ''
3     for c in message.upper():
4         num = ord(c)
5         if 65 <= num <= 90:
6             num -= 65
7             num = (num + key) % 26
8             num += 65
9             c = chr(num)
10        cipher += c
11    return cipher
12
13 def decode(code, key):
14    message = ''
15    for c in code.upper():
16        num = ord(c)
17        if 65 <= num <= 90:
```

```
18         num -= 65
19         num = (num - key) % 26
20         num += 65
21         c = chr(num)
22         message += c
23     return message
24
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
TestCase 0	Easy	Sample case	✔ Success	2	0.1584 sec	9.72 KB
TestCase 1	Easy	Sample case	✔ Success	2	0.0779 sec	9.47 KB
TestCase 2	Easy	Sample case	✔ Success	2	0.0486 sec	9.56 KB
TestCase 3	Easy	Sample case	✔ Success	2	0.0647 sec	9.47 KB
TestCase 4	Easy	Sample case	✔ Success	2	0.0766 sec	9.57 KB
TestCase 5	Easy	Hidden case	✔ Success	2	0.0561 sec	9.61 KB
TestCase 6	Easy	Hidden case	✔ Success	2	0.0719 sec	9.57 KB
TestCase 7	Easy	Hidden case	✔ Success	2	0.0709 sec	9.39 KB
TestCase 8	Easy	Hidden case	✔ Success	2	0.0659 sec	9.68 KB
TestCase 9	Easy	Hidden case	✔ Success	2	0.0428 sec	9.43 KB

No Comments

QUESTION 6



Correct Answer

Score 5


Difficulty Meter > Multiple Choice

QUESTION DESCRIPTION

On a scale of 1 to 5, with 1 being very easy and 5 being extremely challenging, how would you rate this worksheet?


CANDIDATE ANSWER

Options: (Expected answer indicated with a tick)




☐

1




☒

2




☐

3



☐

4




☐

5

No Comments

QUESTION 7



Correct Answer

Score 15

Feedback (Graded) > Subjective

QUESTION DESCRIPTION

Please share with us your thoughts on the following :

- any particular challenges that you faced in attempting this homework
- how you overcame the above challenges

- your problem-solving and programming experience before coming to this course
- your experience learning problem-solving and programming in this course so far

#### CANDIDATE ANSWER

I thought some problems could use hints, so I added them.

I peeked at some of the student solutions that are already submitted and saw that they are using global variables so I added a check to forbid them.

My recursion is rusty so I had to struggle a little.

Recursion questions are at the very beginning. So I changed the ordering to put them at the end. I am not sure if this will reflect at the student end for those who have already begun.

No Comments

PDF generated at: 22 Oct 2023 13:02:29 UTC