COMP9021 Principles of Programming
Term 1, 2024

Coding Quiz 4

Worth 4 marks and due Week 7 Thursday @ 9pm

Description

You are provided with a **stub** in which you need to **insert your code where indicated without doing any changes to the existing code** to complete the task.

Implement a function called, **`encode(list_of_integers)`**, that, based on the encoding of a single strictly positive integer that in **base 2** reads as $b_1 \dots b_n$, as $b_1b_1 \dots b_nb_n$, encodes a sequence of strictly positive integers $N_1, \dots, N_k$ with k >= 1 as $N_1^{*}0 \dots 0\ N_k^{*}$ where for all 0 < i <= k, N_i^{*} is the encoding of N_i.

Implement a function called **`decode(integer)`**, to decode a strictly positive integer **N** into a sequence of (one or more) strictly positive integers according to the previous encoding scheme or return **None** in case **N** does not encode such a sequence.

We **assume** that the user input is **valid**. No need to check for validity, nor to take action in case it is invalid.

Let us explain the above using an example.

First, the **encoding** (resp. **decoding**) scheme used here is that each bit is **duplicated** (resp. **halved**).

For instance, **424896**, in **base 2**, is **1100111101111000000**. Taking the first 2 bits, 11, this **decodes** to 1, the next 2 bits 00 decodes to 0, and so on until we reach the single 0 which represents a separator, meaning we are moving on to a new number. We then start again with the next 2 bits 11 which decodes to 1, and so on until we reach the end of the string. We are then left with a list of numbers in base 2 as **[1011, 11000]** which we convert back to base **10** as **[11, 24]**.

The **encoding** process is the opposite of the above. We take **[11, 24]** as input, converted to base 2 **[1011, 11000]**. Taking the first number, we duplicate each bit getting **11001111** then for the separator add a **0** then duplicate each bit in the next number **1111000000**. Combined together, it forms a single number **1100111101111000000** which when converted to **base 10** is **424896**.

This means that if the base 2 number is something like **111** (7), **101** (5), or **11000** (24), it is then **impossible** to **decode**.

See test cases below for more examples.

Due Date and Submission

Quiz 4 is due **Week 7 Thursday 28 March 2024 @ 9.00pm** (Sydney time).

Note that **late** submission with **5% penalty per day** is allowed **up to 3 days** from the due date, that is, any late submission after **Week 7 Sunday 31 March 2024 @ 9pm** will be discarded.

Make sure not to change the filename `quiz_4.py` while submitting by clicking on **[Mark]** button in **Ed**. It is your responsibility to check that your submission did go through properly using **Submissions** link in Ed otherwise your mark will be **zero** for Quiz 4.

Test Cases

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 1
   In base 2, 1 reads as 1
Incorrect encoding!
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 12345
   In base 2, 12345 reads as 11000000111001
Incorrect encoding!
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 3
   In base 2, 3 reads as 11
   It encodes: [1]
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: [1]
   In base 2, [1] reads as [1]
   It is encoded by 3
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 51315663
   In base 2, 51315663 reads as 11000011110000001111001111
   It encodes: [4891]
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list  of strictly positive integers: [4891]
   In base 2, [4891] reads as [1001100011011]
   It is encoded by 51315663
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 424896
   In base 2, 424896 reads as 1100111101111000000
   It encodes: [11, 24]
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: [11, 24]
   In base 2, [11, 24] reads as [1011, 11000]
   It is encoded by 424896
```

```
$ python3 quiz_4.py
Input either a strictly positive integer
or a nonempty list of strictly positive integers: 857310204
   In base 2, 857310204 reads as 110011000110011000001111111100
   It encodes: [10, 20, 30]
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: [10, 20, 30]
   In base 2, [10, 20, 30] reads as [1010, 10100, 11110]
   It is encoded by 857310204
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: 13609683913728
   In base 2, 13609683913728 reads as 11000110000011000000011000000000110000000000
   It encodes: [2, 4, 8, 16, 32]
```

```
$ python3 quiz_4.py

Input either a strictly positive integer
or a nonempty list of strictly positive integers: [2, 4, 8, 16, 32]
   In base 2, [2, 4, 8, 16, 32] reads as [10, 100, 1000, 10000, 100000]
   It is encoded by 13609683913728
```

Some More Test Cases

Here are some more examples that may help clarify doubts about Quiz 4 requirements:

```
$ python quiz_4_sol.py
Input either a strictly positive integer
or a nonempty list of strictly positive integers: 100
  In base 2, 100 reads as 1100100
Incorrect encoding!

$ python quiz_4_sol.py
Input either a strictly positive integer
or a nonempty list of strictly positive integers: 99
  In base 2, 99 reads as 1100011
  It encodes:  [2, 1]

$ python quiz_4_sol.py
Input either a strictly positive integer
or a nonempty list of strictly positive integers: [2, 1]
  In base 2, [2, 1] reads as [10, 1]
  It is encoded by 99
```