Question 1.

Continue working on Lab #7: implement a circular queue in C.

Question 2.

For this question, you will want to work in Python.

Consider the problem we introduced in lecture: we want to make up an amount of money using coins from a given set of denominations, using the least possible number of coins. For example, if our denominations are `[1c, 4c, 5c]`, the most efficient way to make up `8c` is with two coins, `4c + 4c`.

Part (a)

Suppose you have computed `OPT(0), OPT(1), OPT(2), ..., OPT(n-1)`, the numbers of coins that are needed to make up `0c, 1c, 2c, ..., (n-1)c`.

Write down an expression for `OPT(n)`, in terms of the denominations and `OPT(0), ...., OPT(n)`. Hint: if you want to use the 5c coin, the total number of coins you'll need is `OPT(n-5)+1`

Part (b)

Write code to compute `OPT(0)`, then `OPT(1)`, then `OPT(2)`, ..., then `OPT(n)`. You should write a function that takes in the denominations list and the target amount, and returns the `OPT` list.

Part (c)

Write code that returns a list of the coins you need to make up a target amount, when the least possible number of coins is used. (For example, the function would return `[4, 4]`, if the denominations are `[1c, 4c, 5c]` and the target is `8`).

Follow the example from `houses.py`. The first coin used, `denom`, should be such that `1 + OPT(n-denom)` is as small as possible. To determine the next coin, `denom2`, make `1 + 1 + OPT(n-denom-denom2)` as small as possible.

(Note: it is possible to keep track of the coins used when computing `OPT`, but here we are asking for you to do something analogous to `houses.py`).

Question 3. 2024 Exam Question

Given a non-empty string s and a dictionary `wordDict` containing a list of non-empty words, determine if s can be segmented into a space-separated sequence of one or more dictionary words.

Write a function `canBeSegmented(s, wordDict)` that returns `True` if s can be segmented and `False` otherwise.

If you are given:

$$s = \text{"applepenpple"},$$
$$\texttt{wordDict} = [\text{"apple"}, \text{"pen"}],$$

The output should be `True` because `"applepenpple"` can be segmented as `"apple pen apple"`.

If you are given:

$$s = \text{"catsandog"},$$
$$\texttt{wordDict} = [\text{"cats"}, \text{"dog"}, \text{"sand"}, \text{"and"}, \text{"cat"}],$$

The output should be `False` because `"catsandog"` cannot be segmented into a space-separated sequence of one or more dictionary words.

Hint: the string `s` can be segmented if there is a word `word` in `wordDict` such that `word` is a prefix of `s` and `s[len(word):]` can be segmented.