Programming

You are given 2 questions, and need to solve both of them.
Out of 100 points, 70 or higher points are required to pass the programming section.

Instructions to submit your solutions:

1. The files `QE_prob1.py` and `QE_prob2.py` contain your solution to problems 1 and 2, respectively.
2. Remove any debugging or logging code before you submit. It may disturb the automatic grading process, and as a result, you will likely get a lower score.
3. Compress the three files `QE_prob1.py` and `QE_prob2.py` to a single submission `file 20XX_XXXXX.zip` (`20XX_XXXXX` is your SNU student id, *e.g.*, 2021_12345.zip). The submission file should contain at most three files: `QE_prob1.py` and `QE_prob2.py`.
4. Send the submission file to `gsds_qe@aces.snu.ac.kr` from your SNU email account (if it is not an SNU email account, we will not accept your solution). The title of the submission email should be `[QE] 20XX-XXXXX` (*e.g.*, `[QE] 2020-12345`).
5. Make sure that the attached file is easily downloadable from the email message. We will not accept any submission that requires third-party tools or storages (*e.g.*, Google Drive).

Note: You may use the Internet for API search, but communication with other people in any matter is strictly prohibited. Violation to this will be considered as academic misconduct.

1. [50 pts] In this problem, given a string `s`, you will finally implement a function `max_palindromes(s)` that returns a list of substrings of `s` that are maximal palindromes. That is, the list contains palindromes that are not a substring of another palindrome. A string of characters is a palindrome if it is identical to its reversion. A substring is a contiguous sequence of characters within a string. The characters used in a string are only lower-case alphabets and a space character. **You may not use any built-in functions in Python.**
For example,

- For `s = "kabccbadzdefgfeda"`,
  `max_palindromes(s)` should return `["k", "abccba", "dzd", "defgfed"]`.
- For `s = "kabccba dzabccbaza"`,
  `max_palindromes(s)` should return `["k", " ", "d", "zabccbaz", "aza"]`.

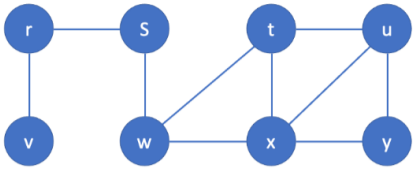
(a) [10 pts] Write a function `palindrome(s)` that checks if the string `s` is a palindrome.

(b) [10 pts] Write a function `substring(s, t)` that checks if the string `t` is a substring of the string `s`.

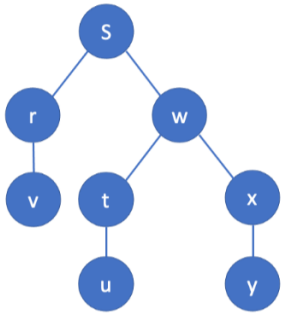
(c) [30 pts] Write the function `max_palindromes(s)` that uses `palindrome(s)` and `substring(s, t)`.

The submission file `QE_prob1.py` should only contain the implementations of the three functions `palindrome(s)`, `substring(s, t)`, and `max_palindromes(s)`. **You will likely get a lower score if there is any print or debugging code in your submission.**

2. [50 pts] In this problem, given a connected undirected graph `G` and a node `s` in `G`, you will implement a function `level_partition(G, s)` that partitions the nodes in the breadth-first tree of `G` rooted at `s` according to their level in the tree. It returns the list of partitions, and the partitions in the list are sorted in an increasing order of the levels. For example, the following graph `G`



produces the following breadth-first tree rooted at `s`:



Thus, `level_partition(G, s)` returns `"[[s], [r, w], [v, t, x], [u, y]]"`.
A node in an undirected graph is defined as follows:

```
# Node definition.
class GNode:
    def __init__(self, id, color="W", d=0, p=None):
        self.id = id.  # id is a string
        self.color = color  # color (status) of node
        self.distance = d
        self.parent = p

    def __str__(self):
        return self.id
```

You can freely add members in the `GNode` definition for your conveniences. Undirected graph `G` is implemented as an adjacency list using a dictionary as follows:

```
>> r, s, t, u, v = GNode('r'), GNode('s'), GNode('t'), GNode('u'), GNode('v')
>> w, x, y = GNode('w'), GNode('x'), GNode('y')
>> G = dict()
>> G[r], G[w], G[t], G[u], G[v] = [s, v], [w, r], [w, x, u], [t, x, y], [r]
>> G[w], G[x], G[y] = [s, t, x], [w, t, u, y], [x, u]
```

(a) [10 pts] Write a function `bfs(G, s)` that performs a breadth-first search (BFS) algorithm on a connected undirected graph `G` from the source node `s`.

(b) [40 pts] Write the function `level_partition(G, s)` that uses `bfs(G, s)`.

The submission file `QE_prob2.py` should contain only the definition of the GNode and implementations of the two functions `bfs(G, s)` and `level_partition(G, s)`. **You will likely get a lower score if there is any print or debugging code in your submission.**