Programming project 1

Due: see Canvas
Group work: You may work in groups of 1-3. Include all group members when submitting to Gradescope.

Write a Python class, *WolfGoatCabbage*, that describes the Wolf, goat and cabbage problem (same problem from HW #2) and can then be used to solve it by calling a search algorithm.

- The class must extend class Problem in the search.py code.
- Represent the state by a *set* of characters, representing the objects on the left bank. Use the characters: 'F', 'G', 'W', 'C'. Note that it is sufficient to represent the objects on one bank since the remaining will be on the other bank. E.g., {'F', 'G'} represents Farmer and Goat on the left bank and Wolf and Cabbage on the right.
- An action in this puzzle is 1-2 objects crossing in the boat. Represent an action as a ***set*** of characters representing the objects crossing. E.g., {'F', 'G'} represents the farmer and goat crossing. Note that it is not necessary to represent the direction of the boat as this will be clear from the state (e.g., if the farmer is on the left, then the boat will have to cross to the right).

The class should be usable in the main function below to print the sequence of actions to reach the goal state.

```
from search import *
# YOUR CODE GOES HERE

if __name__ == '__main__':
    wgc = WolfGoatCabbage()
    solution = depth_first_graph_search(wgc).solution()
    print(solution)
    solution = breadth_first_graph_search(wgc).solution()
    print(solution)
```

Your code should print something like:

```
[{'G', 'F'}, {'F'}, {'W', 'F'}, {'G', 'F'}, {'C', 'F'}, {'F'}, {'G', 'F'}]
[{'G', 'F'}, {'F'}, {'C', 'F'}, {'G', 'F'}, {'W', 'F'}, {'F'}, {'G', 'F'}]
```

Hints:

- The class will be similar to class EightPuzzle in search.py. However, this is a different type of puzzle, so the code logic will be very different, just the structure remains the same. Specifically, your class should have a

1. a constructor that sets the initial and goal states
2. a method `goal_test(state)` that returns True if the given state is a goal state
3. a method `result(state, action)` that returns the new state reached from the given state and the given action[1]. Assume that the action is valid.
4. a method `actions(state)` that returns a list of valid actions in the given state

- It is recommended to implement the methods in the order above and test each method individually
- Do not change anything in search.py

Submit via Gradescope:

Submission will be via Gradescope. Upload a single file with your class. The name of the file must be `wolfgoatcabbage.py`. The name of the class should be `WolfGoatCabbage`. Gradescope will run a few unit tests automatically and show you the results. Include all group members when submitting to Gradescope

[1] In result() you will have to return the next state as a frozenset since the search algorithms require the state to be represented as a hashable data type.