```
def is_parent(self, mother):
    """
    mother: Member
    Returns: Boolean, whether or not `mother` is the
    parent of this Member
    """
    return self.parent == mother
```

```
def get_parent(self):
    """
    Returns the parent
    Member node of this Member
    """
    return self.parent
```

```
def add_child(self, child):
    """
    child: Member
    Adds another child Member node to this Member
    """
    self.children.append(child)
```

call function

```
def add_parent(self, mother):
    """
    mother: Member
    Sets the parent of this node to the
    """
    self.parent = mother
```

```
def is_child(self, child):
    """
    child: Member
    Returns: Boolean, whether or not `child` is a
    child of this Member
    """
    return child in self.children
```

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

```
class Member(object):
    def __init__(self, founder):
        """
        founder: string
        Initializes a member.
        Name is the string of name of this node,
        parent is None, and no children
        """
        self.name = founder
        self.parent = None
        self.children = []
```

```
def is_parent(self, mother, kid):
    """
    Returns True or False whether mother is parent of

    Keyword arguments:
    mother -- string of mother's name
    kid -- string of kid's name
    """
    mom_node = self.names_to_nodes[mother]
    child_node = self.names_to_nodes[kid]
    return child_node.is_parent(mom_node)
```

call function

```
def set_children(self, mother, list_of_children):
    """
    Set all children of the mother.

    Keyword arguments:
    mother -- mother's name as a string
    list_of_children -- children names as strings
    """
    # convert name to Member node (should check for validity)
    mom_node = self.names_to_nodes[mother]
    # add each child
    for c in list_of_children:
        # create Member node for a child
        c_member = Member(c)
        # remember its name to node mapping
        self.names_to_nodes[c] = c_member
        # set child's parent
        c_member.add_parent(mom_node)
        # set the parent's child
        mom_node.add_child(c_member)
```

call function

```
def add_parent(self, mother):
    """
    mother: Member
    Sets the parent of this node to the
    """
    self.parent = mother
```

Menber(mother)

Menber(mother)

```
c_member.add_parent(mom_node)
---> self.parent---->c_member.parent = mom_node
```

```
class Family(object):
    def __init__(self, founder):
        """
        Initialize with string of name of oldest ancestor

        Keyword arguments:
        founder -- string of name of oldest ancestor
        """
        self.names_to_nodes = {}
        self.root = Member(founder)
        self.names_to_nodes[founder] = self.root
```

```
def is_child(self, kid, mother):
    """
    Returns True or False whether kid is child of mother.

    Keyword arguments:
    kid -- string of kid's name
    mother -- string of mother's name
    """
    mom_node = self.names_to_nodes[mother]
    child_node = self.names_to_nodes[kid]
    return mom_node.is_child(child_node)
```