QUESTION 1. [12 MARKS]

I have left out the documentation for function `s(n)`. Work out what it produces, starting at the smallest n and working up.

```
def s(n: int) -> tuple:
    """For you to figure out..."""
    if n == 1:
        return (1, 1)
    else:
        return min([(2 * s(n - i)[0] + 2**i - 1, i) for i in range(1,n)])
```

PART (A) [2 MARKS]

`s(1)`: (1,1)

PART (B) [2 MARKS]

`s(2)`: (3,1)

PART (C) [2 MARKS]

`s(3)`: (5,2)

PART (D) [2 MARKS]

`s(4)`: (9,2)

PART (E) [2 MARKS]

`s(5)`: (13,2)

PART (F) [2 MARKS]

`s(6)`: (17,3)

QUESTION 2. [15 MARKS]

Python's `str.join` concatenates a list of strings, using the given separator string, and returns the new string, as follows:

```
>>> str.join('+',['one', 'two', 'three'])
'one+two+three'
```

Complete the definition of `nested_join(...)`, which concatenates the strings in a nested list of strings, using the given separator string, and returns the result.

```
def nested_join(s: str, L: list) -> str:
    """Return join of nested list of strings L with separator string s

    >>> nested_join(' ', [])
    ''
```

```
    >>> nested_join(' ', ['one'])
    'one'
    >>> nested_join(' ', ['one', 'two'])
    'one two'
    >>> nested_join(' ', ['one', ['two', 'three'], 'four'])
    'one two three four'
    """
```

SOLUTION:

```
    return str.join(s, [nested_join(s, x) if isinstance(x, list) else x for x in L])
```

QUESTION 3. [15 MARKS]

You are to implement `FunctionalList`, a subclass of built-in class `list`. `FunctionalList` should have two new methods:

```
functional_append(self: 'FunctionalList', o: object) -> 'FunctionalList':
    """Return a copy of this FunctionalList with o appended"""

functional_sort(self: 'FunctionalList') -> 'FunctionalList':
    """Return a sorted copy of this FunctionalList"""
```

Note that `functional_append` and `functional_sort` are NOT allowed to change the original list they are called on.

SOLUTION:

```
class FunctionalList(list):
    """list with some functional methods."""

    def functional_append(self: 'FunctionalList', o: object) -> 'FunctionalList':
        """Return a copy of this list with o appended"""
        return FunctionalList([x for x in self] + [o])

    def functional_sort(self: 'FunctionalList') -> 'FunctionalList':
        """Return a sorted copy of this FunctionalList"""
        L = [x for x in self]
        L.sort()
        return FunctionalList(L)
```

PART (A) [8 MARKS]

Implement `FunctionalList` in the space below.

PART (B) [7 MARKS]

Describe three more test cases that would increase your confidence that `functional_append` and `functional_sort` work as specified. One example is given.

- Create `FL = FunctionalList([1])`. Then `FL.functional_append(2)` returns the `FunctionalList` `[1, 2]` and `FL` is left unchanged.
- Create `FL = FunctionalList([])`. Then `FL.functional_append(2)` returns the `FunctionalList` `[2]` and `FL` is left unchanged.
- Create `FL = FunctionalList([3, 1, 2])`. Then `FL.functional_sort()` returns the `FunctionalList` `[1, 2, 3]` and `FL` is left unchanged.
- Create `FL = FunctionalList([3])`. Then `FL.functional_append(2).functional_sort()` returns the `FunctionalList` `[2, 3]` and `FL` remains unchanged.

This page is left (mainly) blank for things that don't fit elsewhere.

1: _____/12

2: _____/15

3: _____/15

TOTAL: _____/42