```
import configparser
from itertools import combinations


class Config:
    def __init__(self, router_id, input_ports, outputs):
        self.router_id = router_id
        self.input_ports = input_ports
        self.outputs = outputs

    def __str__(self):
        lines = f"""CONFIG:
    router id: {self.router_id}
    input ports: {self.input_ports}
    outputs:"""
        for routerid, [port, metric] in self.outputs.items():
            lines += f"""
        router-id: {routerid} port: {port} metric: {metric}"""
        return lines + '\n'


def read_config_file(filename):
    config = configparser.ConfigParser()
    config.read(filename)
    try:
        return get_config(config)
    except ValueError as e:
        raise ValueError(f'CONFIG {filename} ERROR: {e}')


def get_config(config):
    """
    >>> config = configparser.ConfigParser()
    >>> config['SETTINGS'] =
    {'router-id':'2','input-ports':'2000','outputs':'3000-1-3'}
    >>> c1 = get_config(config)
    >>> print(c1)
    CONFIG:
        router id: 2
        input ports: [2000]
        outputs:
            router-id: 3 port: 3000 metric: 1
    <BLANKLINE>

    """
    router_id, input_ports, outputs = validate_config(config)
    return Config(router_id, input_ports, outputs)


def validate_configs_by_filename(filenames):
    configs = [read_config_file(filename) for filename in filenames]
```

```
51      validate_configs(configs)
52
53  def validate_configs(configs):
54      """For all the provided configs:
55      ensures that all router-ids are unique,
56      sending/receiving router-ids match between neighbours,
57      and that metrics between neighbours are the same.
58
59      >>> config1 = configparser.ConfigParser()
60      >>> config1['SETTINGS'] =
        {'router-id':'2','input-ports':'2000','outputs':'3000-1-3'}
61      >>> config2 = configparser.ConfigParser()
62      >>> config3 = configparser.ConfigParser()
63
64      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3000','outputs':'2000-1-2'}
65      >>> validate_configs([get_config(config1), get_config(config2)])
66
67      >>> config2['SETTINGS'] =
        {'router-id':'2','input-ports':'3000','outputs':'2000-1-2'}
68      >>> validate_configs([get_config(config1), get_config(config2)])
69      Traceback (most recent call last):
70      AssertionError: same router-id: 2
71
72      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3333','outputs':'3000-1-2'}
73      >>> validate_configs([get_config(config1), get_config(config2)])
74      Traceback (most recent call last):
75      AssertionError: port 3000 is already an output to router 3
76
77      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3000','outputs':'2000-1-3'}
78      >>> validate_configs([get_config(config1), get_config(config2)])
79      Traceback (most recent call last):
80      AssertionError: router-id mismatch between routers 2 and 3 on port 2000
81
82      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3000','outputs':'2222-1-2'}
83      >>> validate_configs([get_config(config1), get_config(config2)])
84      Traceback (most recent call last):
85      AssertionError: router 2 listening on port 2000 but no sender
86
87      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3333','outputs':'2000-1-2'}
88      >>> validate_configs([get_config(config1), get_config(config2)])
89      Traceback (most recent call last):
90      AssertionError: sending to router 3 on port 3000 but no receiver
91
92      >>> config2['SETTINGS'] =
        {'router-id':'3','input-ports':'3000','outputs':'2000-2-2'}
93      >>> validate_configs([get_config(config1), get_config(config2)])
```

```
94        Traceback (most recent call last):
95        AssertionError: metric mismatch between routers 2 and 3
96
97        >>> config1['SETTINGS'] = {'router-id':'2','input-ports':'2000,2001','outputs':'3000-1-3,4000-2-4'}
98        >>> config2['SETTINGS'] = {'router-id':'3','input-ports':'3000,3001','outputs':'2000-1-2,4001-3-4'}
99        >>> config3['SETTINGS'] = {'router-id':'4','input-ports':'4000,4001','outputs':'2001-2-2,3001-3-3'}
100       >>> validate_configs([get_config(config1), get_config(config2), get_config(config3)])
101       """
102       for c1, c2 in combinations(configs, 2):
103           assert c1.router_id != c2.router_id, f'same router-id: {c1.router_id}'
104
105       port_ids = {} # {port: [input_id, output_id]}
106       metrics = {}  # {(router1_id, router2_id), metric]} # where router1_id < router2_id
107       for config in configs:
108           for port in config.input_ports:
109               current_ids = port_ids.get(port, [None, None])
110               assert current_ids[0] is None, f'port {port} already an input for router {current_ids[0]}'
111               current_ids[0] = config.router_id
112               port_ids[port] = current_ids
113
114           for router_id, [port, metric] in config.outputs.items():
115               current_ids = port_ids.get(port, [None, None])
116               assert current_ids[1] is None, f'port {port} is already an output to router {current_ids[1]}'
117               current_ids[1] = router_id
118               port_ids[port] = current_ids
119
120               lower_id, upper_id = sorted([config.router_id, router_id])
121               current_metric = metrics.get((lower_id, upper_id), None)
122               if current_metric is not None:
123                   assert current_metric == metric, f'metric mismatch between routers {lower_id} and {upper_id}'
124               metrics[(lower_id, upper_id)] = metric
125
126       for port, [in_id, out_id] in port_ids.items():
127           assert in_id != None, f'sending to router {out_id} on port {port} but no receiver'
128           assert out_id != None, f'router {in_id} listening on port {port} but no sender'
129           assert in_id == out_id, f'router-id mismatch between routers {in_id} and {out_id} on port {port}'
130
131
132   def routerid_is_valid(routerid):
133       return  1 <= routerid <= 64000
```

```
134
135 def validate_router_id(routerid):
136     """
137     >>> validate_router_id('1')
138     1
139     >>> validate_router_id('64000')
140     64000
141     >>> validate_router_id('0')
142     Traceback (most recent call last):
143     ValueError: router-id must be a number between 1 and 64000. Got "0"
144     >>> validate_router_id('64001')
145     Traceback (most recent call last):
146     ValueError: router-id must be a number between 1 and 64000. Got "64001"
147     """
148     routerid = routerid.strip()
149     if routerid.isdigit() and routerid_is_valid(int(routerid)):
150         return int(routerid)
151     else:
152         raise ValueError(f'router-id must be a number between 1 and 64000. Got "{
        routerid}"')
153
154
155 def port_is_valid(port):
156     return 1024 <= port <= 64000
157
158 def validate_port(port):
159     """
160     >>> validate_port('1024')
161     1024
162     >>> validate_port('64000')
163     64000
164     >>> validate_port('1023')
165     Traceback (most recent call last):
166     ValueError: port must be a number between 1024 and 64000. Got "1023"
167     >>> validate_port('64001')
168     Traceback (most recent call last):
169     ValueError: port must be a number between 1024 and 64000. Got "64001"
170     """
171     port = port.strip()
172     if port.isdigit() and port_is_valid(int(port)):
173         return int(port)
174     else:
175         raise ValueError(f'port must be a number between 1024 and 64000. Got "{port
        }"')
176
177
178 def metric_is_valid(metric):
179     return 1 <= metric <= 16
180
181 def validate_metric(metric):
182     """
```

```
183         >>> validate_metric('1')
184         1
185         >>> validate_metric('16')
186         16
187         >>> validate_metric('0')
188         Traceback (most recent call last):
189         ValueError: metric must be a number between 1 and 16. Got "0"
190         >>> validate_metric('17')
191         Traceback (most recent call last):
192         ValueError: metric must be a number between 1 and 16. Got "17"
193         """
194         metric = metric.strip()
195         if metric.isdigit() and metric_is_valid(int(metric)):
196             return int(metric)
197         else:
198             raise ValueError(f'metric must be a number between 1 and 16. Got "{metric
                }"')
199
200
201     def validate_config(config):
202         """
203         >>> config = configparser.ConfigParser()
204         >>> validate_config(config)
205         Traceback (most recent call last):
206         ValueError: SETTINGS header not found
207
208         >>> config['SETTINGS'] = {'input-ports':'1024','outputs':'64000-0-1'}
209         >>> validate_config(config)
210         Traceback (most recent call last):
211         ValueError: "router-id" parameter not found
212
213         >>> config['SETTINGS'] = {'router-id':'1','outputs':'64000-0-1'}
214         >>> validate_config(config)
215         Traceback (most recent call last):
216         ValueError: "input-ports" parameter not found
217
218         >>> config['SETTINGS'] = {'router-id':'1','input-ports':'1024'}
219         >>> validate_config(config)
220         Traceback (most recent call last):
221         ValueError: "outputs" parameter not found
222
223         >>> config['SETTINGS'] =
            {'router-id':'1','input-ports':'2000,2000','outputs':'5000-15-1'}
224         >>> validate_config(config)
225         Traceback (most recent call last):
226         ValueError: "2000" is a duplicate port number
227
228         >>> config['SETTINGS'] =
            {'router-id':'1','input-ports':'2000','outputs':'2000-15-1'}
229         >>> validate_config(config)
230         Traceback (most recent call last):
```

```
231       ValueError: "2000" is already defined as an input port
232
233       >>> config['SETTINGS'] =                                                    ↵
          {'router-id':'1','input-ports':'1024','outputs':'64000-1-1'}
234       >>> validate_config(config)
235       (1, [1024], {1: [64000, 1]})
236
237       >>> config['SETTINGS'] = {'router-id':' 01 ','input-ports':' 01024 ,        ↵
          01025','outputs':' 064000 - 011 - 01 , 05000 - 012 - 02'}
238       >>> validate_config(config)
239       (1, [1024, 1025], {1: [64000, 11], 2: [5000, 12]})
240
241       >>> config['SETTINGS'] =                                                    ↵
          {'router-id':'1','input-ports':'2000,2001,2002','outputs':'5000-14-2,5001-15-64 ↵
          000'}
242       >>> validate_config(config)
243       (1, [2000, 2001, 2002], {2: [5000, 14], 64000: [5001, 15]})
244
245       """
246       if not 'SETTINGS' in config:
247           raise ValueError('SETTINGS header not found')
248       for param in ['router-id', 'input-ports', 'outputs']:
249           if not param in config['SETTINGS']:
250               raise ValueError(f'"{param}" parameter not found')
251
252       router_id = config['SETTINGS']['router-id']
253       router_id = validate_router_id(router_id)
254
255       input_ports_str = config['SETTINGS']['input-ports'].split(',')
256       input_ports = []
257       for port in input_ports_str:
258           port = validate_port(port)
259           if port in input_ports:
260               raise ValueError(f'"{port}" is a duplicate port number')
261           else:
262               input_ports.append(port)
263
264       outputs_str = config['SETTINGS']['outputs'].split(',')
265       outputs = {}
266       for output in outputs_str:
267           port, metric, out_routerid = output.strip().split('-')
268
269           port = validate_port(port)
270           if port in input_ports:
271               raise ValueError(f'"{port}" is already defined as an input port')
272           metric = validate_metric(metric)
273           out_routerid = validate_router_id(out_routerid)
274
275           outputs[out_routerid] = [port, metric]
276
277       if input_ports == []:
```

```
278             raise ValueError(f'There must be at least one input port')
279         if outputs == []:
280             raise ValueError(f'There must be at least one output')
281 
282         return router_id, input_ports, outputs
283 
284 
285     if __name__ == '__main__':
286         import doctest
287         results = doctest.testmod()
288         print(results)
289 
```