

"""

COSC364 2022-S1 Assignment: RIP routing
Authors: MENG ZHANG (71682325), ZHENG CHAO (21671773)
File: IO_parser.py

```
def router_config(file_name):
    """
    Parameter:
    file_name: string
    file format:
    i.e.
    -----
    router-id 2
    input-ports 6020, 6021
    output-ports 6010-1-1, 6030-2-3
    period 3
    timeout 18
    -----

    Return: config_data
    a dictionary with 4 keys of router_id, input_ports, output_ports,
    timers
    i.e. {'router_id': 2, 'input_ports': [6020, 6021],
    'output_ports_metric_id': {6010: {'metric': 1, 'router_id': 1},
    6020: {...}}, 'period': 3, 'timeout': 18}
    """
    raw_config = read_config(file_name)
    config_data = parse_config(raw_config)
    return config_data


def read_config(file_name):
    """
    Parameter:
    file_name: string.
    file format:
    i.e.
    -----
    router-id 2
    input-ports 6020, 6021
    output-ports 6010-1-1, 6030-2-3
    period 3
    timeout 18
    -----

    Return: a list of strings with 4 elements.
    i.e. ['router-id 2', 'input-ports 6020, 6021', 'output-ports 6010-1-1,
    6030-2-3', 'period 3', 'timeout 18']
    """
    try:
        with open(file_name) as config_file:
            raw_config = config_file.read().splitlines()
            return raw_config
    except FileNotFoundError:
        print("Error: the config file name is invalid")


def parse_config(raw_config):
    """
    Parameter:
    raw_config: a list of strings with 4 elements.
    i.e. ['router-id 2', 'input-ports 6020, 6021', 'output-ports 6010-1-1,
    6030-2-3', 'period 3', 'timeout 18']

    Return: config_data
```

```

64 a dictionary with 4 keys of router_id, input_ports, output_ports,
65 timers
66 i.e. {'router_id': 2, 'input_ports': [6020, 6021],
67      'output_ports_metric_id': {6010: {'metric': 1, 'router_id': 1},
68      6020: {...}}, 'period': 3, 'timeout': 18}
69 """
70 try:
71     # get router id
72     router_id = parse_id(raw_config[0])
73     # get input ports
74     input_ports = parse_input_ports(raw_config[1])
75     # check if input ports contains duplicate ports
76     if contains_duplicates(input_ports):
77         raise ValueError("The input ports contains duplicate ports")
78     # get output ports
79     output_ports, output_ports_metric_id = parse_output_ports(raw_config[2])
80     # check if input ports and output ports contain duplicate ports
81     if duplicate_lists(input_ports, output_ports):
82         raise ValueError("The input ports and output ports contain duplicate ports")
83     # get period
84     period = parse_period(raw_config[3])
85     # get timeout
86     timeout = parse_timeout(raw_config[4])
87     # check timeout vs period ratio
88     if not is_valid_timer_ratio(period, timeout):
89         raise ValueError("The ratio timeout vs period should be 6")
90     # create coinfig_data dictionary
91     config_data = {"router_id": router_id, "input_ports": input_ports,
92                  "output_ports_metric_id": output_ports_metric_id,
93                  "period": period, "timeout": timeout}
94     return config_data
95 except IndexError as ie:
96     print(ie)
97     print("Some value of the config file is not available")
98 except ValueError as ve:
99     print(ve)
100    print("Some value of the config file is invalid")
101
102
103
104 def parse_id(raw_id):
105     """
106     Parameter:
107     raw_id: a string
108     i.e. 'router-id 2'
109
110     Return: router_id
111     an interger between 1 and 64000 i.e. 1
112     """
113     try:
114         router_id = int(raw_id.split()[1])
115         if (router_id < 1 or router_id > 64000):
116             raise ValueError("Router ID value is out of bounds")
117         return router_id
118     except IndexError as e:
119         print(e)
120         print("The config router ID value is not available")
121     except ValueError as e:
122         print(e)
123         print("The config router ID value must be an integer between 1 and 64000")
124
125
126 def parse_input_ports(raw_input_ports):
127     """
128     Parameter:
129     raw_input_ports: a string
130     i.e 'input-ports 6020, 6021'
131

```

```

132     Return: input_ports
133     a list of integers which are between 1024 and 64000
134     i.e. [6020, 6021]
135     """
136
137     try:
138         input_ports_temp = raw_input_ports.split()[1:]
139         input_ports = []
140         for port_str in input_ports_temp:
141             port_int = int(port_str.strip(','))
142             if (port_int < 1024 or port_int > 64000):
143                 raise ValueError("Input port value is out of bounds")
144             input_ports.append(port_int)
145         return input_ports
146     except IndexError as e:
147         print(e)
148         print("The config input port value is not available")
149     except ValueError as e:
150         print(e)
151         print("The config input port value must be an integer between 1024 and 64000")
152
153
154 def parse_output_ports(raw_output_ports):
155     """
156     Parameter:
157     raw_input_ports: a string
158     i.e 'output-ports 6010-1-1, 6030-2-3'
159
160     Return: output_ports, output_ports_metric_id
161     output_ports: a list of integers which are between 1024 and 64000
162     i.e. [6010, 6030]
163     output_ports_metric_id: a dict of dicts in which key is port number
164     and each sub dict contains key(port)'s metric and id.
165     Metric > 0, 1 <= ID <= 64000
166     i.e. {6010: {'metric': 1, 'router_id': 1}, 6020: {...}}
167     """
168
169     try:
170         output_ports_combo_temp = raw_output_ports.split()[1:]
171         output_ports = []
172         output_ports_metric_id = {}
173         for port_combo_str in output_ports_combo_temp:
174             port_combo_temp = port_combo_str.strip(',').split('-')
175             port_int = int(port_combo_temp[0])
176             metric_int = int(port_combo_temp[1])
177             id_int = int(port_combo_temp[2])
178             if (port_int < 1024 or port_int > 64000):
179                 raise ValueError("Output port value is out of bounds")
180             if metric_int < 1:
181                 raise ValueError("Output port metric is out of bounds")
182             if id_int < 1 or id_int > 64000:
183                 raise ValueError("Output id is out of bounds")
184             output_ports.append(port_int)
185             # output_ports_metric_id.append([port_int, metric_int, id_int])
186             output_ports_metric_id[port_int] = {'metric': metric_int,
187                                                 'router_id': id_int}
188         return output_ports, output_ports_metric_id
189     except IndexError as e:
190         print(e)
191         print("The config output port value is not available")
192     except ValueError as e:
193         print(e)
194         print("The config output ports must be formatted as port-metric-id")
195         print("The config output port value must be an integer between 1024 and 64000")
196         print("The config output port metric must be an integer greater than 0")
197         print("The config output port id must be an integer between 1 and 64000")
198
199

```

```

200
201 def parse_period(raw_period):
202     """
203     Parameter:
204     raw_period: a string
205     i.e. 'period 3'
206
207     Return: period
208     period: a positive integer
209     i.e. 3
210     """
211     try:
212         period = int(raw_period.split()[1])
213         if period < 1:
214             raise ValueError("Router period value is out of bounds")
215         return period
216     except IndexError as e:
217         print(e)
218         print("The config router period value is not available")
219     except ValueError as e:
220         print(e)
221         print("The config router timeout value must be a positive integer")
222
223
224 def parse_timeout(raw_timeout):
225     """
226     Parameter:
227     raw_timeout: a string
228     i.e. 'timeout 18'
229
230     Return: timeout
231     timeout: a positive integer
232     i.e. 18
233     """
234     try:
235         timeout = int(raw_timeout.split()[1])
236         if timeout < 1:
237             raise ValueError("Router timeout value is out of bounds")
238         return timeout
239     except IndexError as e:
240         print(e)
241         print("The config router timeout value is not available")
242     except ValueError as e:
243         print(e)
244         print("The config router timeout value must be a positive integer")
245
246
247 def contains_duplicates(lst):
248     """
249     Parameter:
250     lst: a list
251
252     Return: boolean
253     if the lst contains duplicates, return true, otherwise false
254     """
255     return len(set(lst)) != len(lst)
256
257
258 def duplicate_lists(lst1, lst2):
259     """
260     Parameters:
261     lst1: a list
262     lst2: a list
263
264     Return: boolean
265     if the two lists contains duplicate items, return true, otherwise false
266     """
267     return len(set(lst1).union(set(lst2))) != len(lst1) + len(lst2)

```

```
268
269 def is_valid_timer_ratio(period, timeout):
270     """
271     Parameters:
272     period: a positive integer
273     period: a positive integer
274
275     Return: boolean
276     if timeout / period = 6, return true, otherwise false
277     """
278     return timeout / period == 6
```