Sensor Nodes

Create a struct Alert, this will contain the basic information of an alert to send:
- alert time: a struct tm type, which contains day, date, time
- number of matching nodes: an integer

Create a struct NodeInfo, this will contain all the information of a node to send:
- rank: an integer
- coordinates: an array with size 2
- temperature: an integer
- MAC address: a character array (string)
- IP address: a character array (string)

*__Note__: we cannot create a single struct that stores the alert information + nodes information, because the size of the struct will be dynamic (some ranks will only have 2 neighbors) hence will not be able to create a fixed size struct to send. So to make this dynamic, we pack Alert + NodeInfo into a buffer. Suppose we have 2 neighboring nodes, we send the 'report' in the following order:

1. Pack alert into buffer (struct Alert)
2. Pack the number of neighboring nodes into buffer (integer)
3. For each neighboring node, pack the NodeInfo struct into buffer (struct NodeInfo)

nodeFunction algorithm:
At every iteration:

Initialize local variable temperature to store random value
Generate random temperature for the first time
Execute 3 sections simultaneously:

Section 1	Section 2	Section 3

generate random temperature
if value $>$ 80:
send NodeInfo request to all neighbors
receive NodeInfo request from neighbors
wait until all NodeInfo requests satisfied
 	
if $\geqslant$ 2 temperatures difference by $\leq$ 5:
get alert time
get the number of nodes within the boundary
add the information into an Alert struct
pack struct into buffer
for all received processes:
pack received NodeInfo into buffer
send buffer to base station
	receive NodeInfo request
	get rank
	get coordinates
	get temperature (obtain generated above)
	get MAC address
	get IP address
	pack these information into a NodeInfo struct
	send the NodeInfo struct to requesting rank
		receive termination request
		terminate by returning function

*Note: all 3 sections will execute simulatenously

nodeFunction pseudocode:

```
while (not terminated yet):
    generate random temperature
    if (value > 80):
        Isend(..., tag = requestNode, request = nodeRequests[all neighbors]) to all neighbors
        Irecv(..., datatype = NodeInfoType, request = nodeRequests[all neighbors]) from all neighbors
        set wait to true
    sleep(5)
    if (Iprobe(..., tag = requestNode, ...) is true):
        Recv(..., source = rank, tag = requestNode, ...)
        get current rank, coordinates, temperature, MAC address, IP address
        assign these information into NodeInfo struct
        Isend(..., datatype = NodeInfoType, destination = rank, request = nodeRequests[all neighbors])

    if (Iprobe(..., tag = termination, ...) is true):
        Recv(.., tag = termination, ...)
        for each MPI_Request in nodeRequests:
            test if the request is satisfied
            if (request is not satisfied yet):
                cancel the request
        set terminated to true
        break loop

    while (wait is true):
        if (Iprobe(..., tag = requestNode, ...) is true):
            Recv(..., source = rank, tag = requestNode, ...)
            get current rank, coordinates, temperature, MAC address, IP address
            assign these information into NodeInfo struct
            Isend(..., datatype = NodeInfoType, destination = rank, ...)

        if (Iprobe(..., tag = termination) is true):
            Recv(.., tag = termination, ...)
            for each MPI_Request in nodeRequests:
                test if the request is satisfied
                if (request is not satisfied yet):
                    cancel the request
            set wait to false
            set terminated to true
            continue loop

        test if all nodeRequests are satisfied
        if (all requests satisfied):
            absoluteCount = getAbsoluteCount(dataReceived)
            if (absoluteCount >= 2):
                get alert time, absoluteCount
                assign these information into an Alert struct
                pack the Alert struct into buffer
                pack the current process's NodeInfo into buffer (itself and not neighbors)
                pack the number of neighbours integer into buffer
                for all received processes (neighbors):
                    pack received NodeInfo into buffer
                send buffer to base station
            set wait to false

    if (node is terminated):
        continue
    sleep for 1 second before generating next temperature
```

Helper functions:

```
int getAbsoluteCount(array of NodeInfo, myTemperature):
        count = 0
        for each NodeInfo:
                if (|node.temperature – myTemperature|  <= 5):
                        count++
        return count
```

Base Station

Create a struct InfraredData, this will contain the data simulated by the satellite for all processes:
- timestamp: a struct tm time, which contains day, date and time that simulates the temperatures
- values: an array of size of the cartesian grid that stores the temperature generated for each process at this time

baseFunction algorithm
create an array of InfraredData structs to store the simulated temperature values
create one thread to populate the random temperature simulation
the algorithm will run in 2 sections simultaneously:

Section 1 (baseFunction)

```
set the global array of InfraredData to a size of 10 (10 time units)
create a global array of InfraredData struct of a size of number of time units
for every time unit in the global array:
    set timestamp to NULL
    intialize the values array in the struct to a size of the cartesian grid
    set each of those values to 0 (default temperature)
create a thread and pass the cartesian size to the thread
set count to 0
while (count < 100):
    receive reportTag from any nodes
        get the current timestamp
        unpack the Alert struct
        get the communication time (based on the Alert's timestamp and current timestamp)
        unpack the NodeInfo of the reporting node
        unpack the number of neighbors
        create an array of reportNeighbors to store to the NodeInfo of the reporting node's neighbors
        for every neighbor:
            unpack the NodeInfo struct
            add this NodeInfo struct into the reportNeighbors array
        get timestamp of the reporting node
        set alert to be false, incremenet the falseAlertCount
        for every time unit simulation in the global array:
            if the | simulation's timestamp - reporting node's timestamp | ⩽ 1 second:
                if both temperatures have differences of ⩽ 5:
                    set alert to be true, decrement the falseAlertCount, increment trueAlertCount
                    get the simulation's timestamp
                    get the simulation's temperature
                    break
        get the current log timestamp
        report the metrics above into a log file
    increase count by 1
    sleep for 500 milliseconds // 100 iterations, each iteration = 0.5 second, total runtime = 50 seconds
send termination signal to all processes
report final log file
```

Section 2 (thread)

```
get the size of the cartesian
for every time unit simulation in the global array:
    for every rank in the cartesian:
        get a random temperature
        assign the random temperature in the values array
        sleep for 100 milliseconds before next random generation
```

logFileData at each iteration
Iteration: X
Logged Time: X
Alert Type: X
Number of Adjacent Matching Nodes: X

Reporting Node	Coordinate	Temperature	MAC Address	IP Address
X	X	X	X	X

Adjacent Nodes	Coordinate	Temperature	MAC Address	IP Address
X	X	X	X	X

Infrared Satellite Reporting Time: X
Infrared Satellite Reporting Temperature: X
Communication Time: X

finalLogFileData when base station terminates
Number of True Alerts: X
Number of False Alerts: X
Average Number of Adjacent Matching Nodes: X
Average Communication Time between Base Station and Node: X

*log file content requires further discussion

Iteration : 15
Logged Time : Wed 2020-09-16 19:23:12
Alert Reported Time : Wed 2020-09-16 19:23:11
Alert Type: True

Reporting Node	Coord	Temp	MAC	IP
12	(2,2)	98	54:bf:64:91:f8:8a	118.139.135.57

Adjacent Nodes Temp	Coord	Temp	MAC	IP
7	(1,2)	96	54:bf:64:91:f8:8a	118.139.135.57
11	(2,1)	95	54:bf:64:91:f8:8a	118.139.135.57
13	(2,3)	97	54:bf:64:91:f8:8b	118.139.135.58
17	(3,2)	80	54:bf:64:91:f8:8b	118.139.135.58

Infrared Satellite Reporting Time (Celsius) : Wed 2020-09-16 19:23:08
Infrared Satellite Reporting (Celsius) : 98
Infrared Satellite Reporting Coord : (2,2)

Communication Time (seconds) : 0.075
Total Messages send between reporting node and base station: 1
Number of adjacent matches to reporting node: 3

Unpack from Alert struct
Unpack from NodeInfo struct
Calculated in baseFunction
May not needed in log