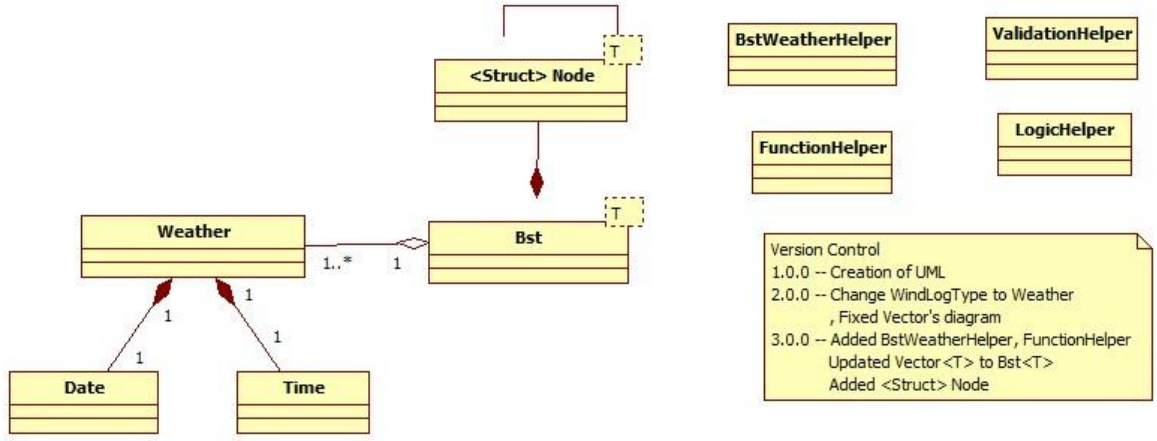
ICT 283
Assignment 2

Lim Wen Chao CT0360379/34368872

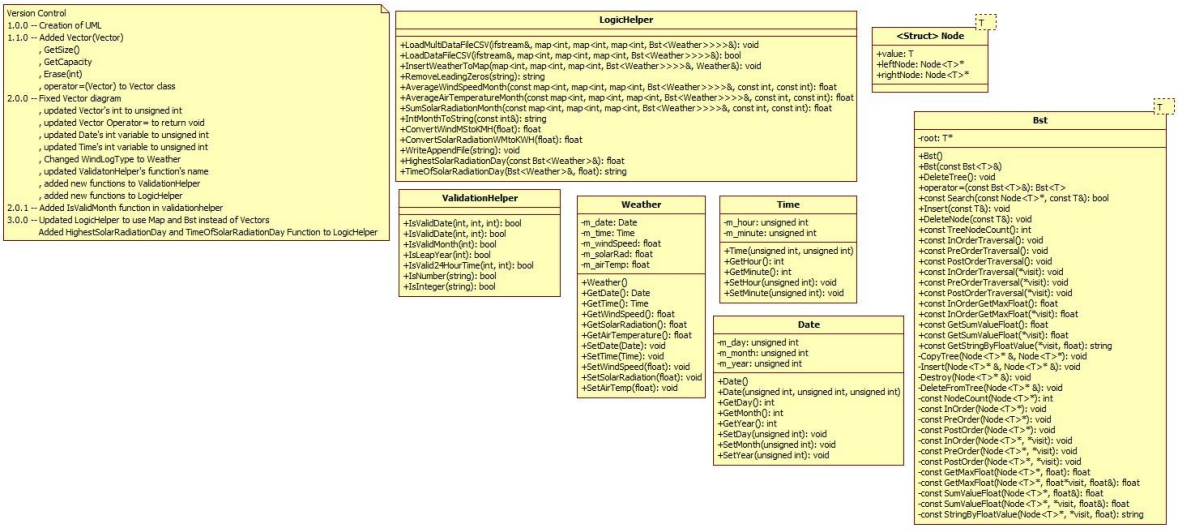
Table of Contents

UML

HIGH-LEVEL UML



LOW-LEVEL UML



Data Dictionary

Refer to the excel document “DataDictionary.xlsx”

Algorithm

High Level

1. Get .txt file with list of data file names to read (met_index.txt)
2. Read Index File line by line
    1) Get .csv file with weather data
    2) Read .csv file line by line
        1) Read column by column
        2) Get DateTime (WAST)
            a) Split DateTime into data and time

        b) Store Date in Date class
        c) Store Time in Time Class
    3) Get WindSpeed (S)
    4) Get Solar Radiation (SR)
    5) Get Ambient Air Temperature (T)
  3) Repeat for each line
3. Repeat for each file name in Index File
4. Store each line of extracted data into Weather object
5. For each Weather of a year
    1) Store in Map<Year, Map<Month, Map<Day, Bst<Weather>* >* >* > (Bst sort by time)
6. Display menu
7. If option 1 is chosen
    1) Ask user for month and year input
    2) Go to corresponding year and month of Maps and get all Weather object
    3) Sum all the WindSpeed and Air Temperature and count the number of Weather objects
    4) Divide WindSpeed and Air Temperature by count to get average
    5) Convert windspeed from m/s to km/h
    6) Print result
8. If Option 2 is chosen
    1) Ask user for year input
    2) Go to corresponding year of Maps and get all weather object
    3) Sum all wind speed and air temperature and count the number of Weather that matches
    4) Divide wind speed and air temperature by count to get average
    5) Convert wind speed from m/s to km/h
    6) Print result
9. If option 3 is chosen
    1) Ask user for year input
    2) Go to corresponding year of maps and get all weather object
    3) Sum all solar radiation
    4) Convert W/m^2 to kWh/m^2
    5) Print result
10. If option 4 is chosen
    1) Ask user for year input
    2) Go to corresponding year of maps and get all weather object
    3) Get average wind speed, average air temperature and total solar radiation
    4) Print result
11. If option 5 is chosen
    1) Ask user for date input (d/m/yyyy)
    2) Go to corresponding map for that date and get all weather objects
    3) Find the weather objects with the highest Solar Radiation
    4) Get the time for those weather objects
    5) Print result
12. If option 6 is chosen

1) Exit program

Low Level

Refer to the txt document “Pseudocode.txt”

Rationale

Weather Log data are stored in map(year) of map(month) of map(day) of BST of Weather data that is inserted based on time. The rationale for the above structure is to emulate a calendar when storing data. A particular date will only be inserted into the maps only if a log data with that date exist. Since all the current functions are searching by date, we can improve search time of the functions by storing it separated by date at the cost of increasing data loading time.
Another method would be to store Weather Log data in a map with datetime as key while storing the datetime in BSTs with one BST per year. This will improve loading time at the cost of longer search time instead as the program would have to search through the BST to look for if data with that date exist before finding the actual Weather data in map by searching through the map for the date again.

Test plan

Date Unit Test

Test	Description	Expected Output	Passed
1	Check constructor correctly initializes the data and all getters are able to retrieve the data	Default Constructor Day: -1 Month: -1 Year: -1 Constructor Day: 1 Month: 1 Year: 1	Pass
2	Check that day setter are working correctly	Test Day setter Day: 3	Pass
3	Check that month setter are working correctly	Test Month setter Month: 3	Pass
4	Check that year setter are working correctly	Test Year setter Year: 3	Pass

Time Unit Test

Test	Description	Expected Output	Passed

1	Check constructor correctly initializes the data, and all getters can retrieve the data	Default Constructor Hour: -1 Minute: -1 Constructor Hour: 1 Minute: 1	Pass
2	Check that hour setter are working correctly	Test Hour setter Hour: 3	Pass
3	Check that minute setter are working correctly	Test Minute setter Minute: 3	Pass
4	Check that < operator overload function can correctly compare 2 Time objects	Test < operator Test Time 1: 05:05 Test Time 2: 03:03 Compare testTime1 < testTime2: 0	Pass
5	Check that > operator overload function can correctly compare 2 Time objects	Test > operator Test Time 1: 05:05 Test Time 2: 03:03 Compare testTime1 > testTime2: 1	Pass

Weather Unit Test

Test	Description	Expected Output	Passed
1	Check that constructor correctly initialize the data and all getters can retrieve the data	Check constructor Date: -1/-1/-1 Time: 0-1:0-1 Wind Speed: -1 Solar Radiation-1 Air Temperature: -1	Pass
2	Check that Date setter are working correctly	Date: 16/3/2016	Pass
3	Check that Time setter are working correctly	Time: 09:15	Pass
4	Check that Wind Speed setter are working correctly	Wind Speed: 23.1	Pass
5	Check that Solar Radiation setter are working correctly	Solar Radiation: 53.15	Pass
6	Check that Air Temperature setter are working correctly	Air Temperature: 53.25	Pass
7	Check that < operator can correctly compare the time of weather object	Test < operator Time of weather 1: 0-1:0-1 Time of weather 2: 09:15 Compare weather1 < weather2: 1	Pass
8	Check that > operator can correctly compare the time of weather object	Test > operator Time of weather 1: 0-1:0-1 Time of weather 2: 09:15	Pass

		Compare weather1 > weather2: 0	

BstWeatherHelper Unit Test

Test	Description	Expected Output	Passed
1	Check that GetWeatherWindSpeed function can correctly get the wind speed data from given weather object	Test GetWeatherWindSpeed 23.1	Pass
2	Check that GetWeatherAirTemp function can correctly get the air temperature data from given weather object	Test GetWeatherAirTemp 53.25	Pass
3	Check that GetWeatherSolarRad function can correctly get the solar radiation data	Test GetWeatherSolarRad 53.15	Pass
4	Check that GetWeatherTimeBySolarRad function can correctly get the time data from given weather object based on given solar radiation value	Test GetWeatherTimeBySolarRad Found match: 09:15 Match not found:	Pass

BST Unit Test

Test	Description	Expected Output	Passed
1	Check that Constructor can correctly initialize an empty BST and TreeNodeCount function would return 0	Test Constructor 0	Pass
2	Check that the copy constructor can correctly deep copy another Vector	Test Copy constructor Tree1 values: 1 Tree2 values after copying Tree1 and deleting Tree1: 1	Pass
3	Check that the deconstructor can successfully delete the Vector	Test deconstructor Tree1 values: Tree2 values: 1 Tree2 values after decontructed:	Pass
4	Check that BST is correctly deleted using deleteTree function	Test DeleteTree function Tree1 values: Tree1 values after deleting tree:	Pass
5	Check that Operator= can correctly deep copy another Vector	Test = operator Tree1 values: 1	Pass

		Tree2 values after copying Tree1 and deleting Tree1: 1	
6	Check that Search function can correctly determine if a value exist in BST	Test Search function Value found: 1 Value not found: 0	Pass
7	Check that Insert function can correctly insert into BST	Test Insert function Tree1 values: After insert Tree1 values: 1 2 3 4 5	Pass
8	Check that DeleteNode function can correctly delete a node in BST by value given	Test DeleteNode function Tree1 values: 1 2 3 4 5 Deletes node Tree1 values after delete: 2 3 4 5	Pass
9	Check that TreeNodeCount function can return the correct number of node in BST	Test TreeNodeCount function Count: 4	Pass
10	Check that InOrderTraversal function can traverse the BST in an in order way	Test InOrderTraversal function 1 2 3 4 5	Pass
11	Check that PreOrderTraversal function can traverse the BST in a pre order way	Test PreOrderTraversal function 2 1 5 4 3	Pass
12	Check that PostOrderTraversal function can traverse the BST in a post order way	Test PostOrderTraversal function 2 1 5 4 3	Pass
13	Check that InOrderGetMaxFloat function can correctly get the largest value in BST	Test InOrderGetMaxFloat function 5 Test InOrderGetMaxFloat with function as parameter 33.1	Pass
14	Check that GetSumValueFloat function can correctly get the total value of all nodes in BST	Test GetSumValueFloat function 15 Test GetSumValueFloat with function as parameter 69.3	Pass
15	Check that GetStringByFloatValue function can return correct string based on given float value	Test GetStringByFloatValue with function as parameter 09:15 09:15 09:15	Pass

LogicHelper Unit Test

Test	Description	Expected Output	Passed
1	Check that LoadMultiDataFileCSV can	Test LoadMultiDataFileCSV function Loading file: dataFile1.csv	Pass

	correctly load multiple data file into map using a index file	Loaded data file: dataFile1.csv Loading file: dataFile2.csv Loaded data file: dataFile2.csv 1/1/2012 08:20 1/1/2012 08:30 1/1/2012 08:40 1/1/2012 08:50 1/1/2012 09:00	
2	Check that LoadDataFileCSV can correctly load a data file into map using a index file	Test LoadDataFileCSV function 17/8/2010 19:10 17/8/2010 19:20 17/8/2010 19:30 17/8/2010 19:40 17/8/2010 19:50	Pass
3	Check that InsertWeatherToMap can correctly insert a Weather Object into map	Test InsertWeatherToMap function 16/3/2016 09:15	Pass
4	Check that RemoveLeadingZeros function can correctly remove the leading zeros when given a string	Test RemoveLeadingZeros(string) Test RemoveLeadingZeros(08): 8 Test RemoveLeadingZeros(08.9): 8.9	Pass
5	Check that IntMonthToString function can correctly return the month in string when provided with a month in integer	Test IntMonthToString(unsigned int) Test IntMonthToString(8): August Test IntMonthToString(13): Error converting int month to string ERROR	Pass
6	Check that ConvertWindMStoKMH function can correctly convert windspeed provided in m/s to km/h	Test ConvertWindMStoKMH(float) Test ConvertWindMStoKMH(10): 36 Test ConvertWindMStoKMH(10.5): 37.8	Pass
7	Check that ConvertSolarRadiationWMtoKWH function can correctly convert solar radiation provided in W/m^2 to kWh/m^2	Test ConvertSolarRadiationWMtoKWH(float) Test ConvertSolarRadiationWMtoKWH(120): 0.02 Test ConvertSolarRadiationWMtoKWH(120.42): 0.02007	Pass
8	Check that AverageWindSpeedMonth function can correctly find and calculate the average windspeed for a month	Test AverageWindSpeedMonth(vector,int,int) Test AverageWindSpeedMonth(TestLog,3,2016): 5.4	Pass

		Test AverageWindSpeedMonth(TestLog,1,2020): -1	
9	Check that AverageAirTemperatureMonth function can correctly find and calculate the average ambient air temperature for a month	Test AverageAirTemperatureMonth(vector,int,int) Test AverageAirTemperatureMonth(TestLog,3,2016): 21.33 Test AverageAirTemperatureMonth(TestLog,1,2020): -1	Pass
10	Check that SumSolarRadiationMonth function can correctly find and calculate the total solar radiation for a month	Test SumSolarRadiationMonth(vector,int,int) Test SumSolarRadiationMonth(TestLog,3,2016): 2891 Test SumSolarRadiationMonth(TestLog,1,2020): -1	Pass
11	Check that WriteAppendFile able to correctly write to "WindTempSolar.csv"	Test WriteAppendFile(string) *Look for the test.csv file teststring teststring2	Pass
12	Check that HighestSolarRadiationDay function can find the largest solar radiation value in a map of weather	Test HighestSolarRadiationDay function 906	Pass
13	Check that TimeOfSolarRadiationDay function can find the list of time of weather object that has the same solar radiation value as the given value	Test TimeOfSolarRadiationDay function 09:20 09:30	Pass

ValidationHelper Unit Test

Test	Description	Expected Output	Passed
1	Check that IsValidDate function can correctly determine if a date is valid when given the day, month and year	Checking IsValidDate(int day, int month, int year) Positive case (29/2/2020): 1 Negative case, wrong day (31/2/2020): 0 Negative case, wrong month (29/13/2020): 0	Pass

		Negative case, wrong year (29/2/99): 0 Negative case, not leap year (29/2/2022): 0	
2	Check that IsValidDate function can correctly determine if a date is valid when given the month and year only	Checking IsValidDate(int month, int year) Positive case (2/2020): 1 Negative case, wrong month (13/2020): 0 Negative case, wrong year (2/99): 0	Pass
3	Check that IsValidMonth function can correctly determine if a int is valid month	Checking IsValidMonth(int month) Positive case (2): 1 Negative case, wrong month (13): 0	Pass
4	Check that IsLeapYear function can correctly determine if a year is a leap year	Checking IsLeapYear(int year) Positive case (2020): 1 Negative case (2022): 0	Pass
5	Check that IsValid24HourTime function can correctly determine if a given hour and minute is valid 24 hour time	Checking IsValid24HourTime(int hour, int minute) Positive case (8,50): 1 Negative case, wrong hour (25,50): 0 Negative case, wrong minute (24,60): 0	Pass
6	Check that IsNumber function can correctly determine if a provided string is positive int/float	Checking IsNumber(string input) Positive case (415): 1 Positive case (63.41): 1 Negative case, negative int (-1): 0 Negative case, negative float (-1.2): 0 Negative case, too many decimal points (123456.123456789012345): 1 Negative case, octal numbers (08): 0	Pass
7	Check that IsInteger function can correctly determine if a provided string is a positive integer	Checking IsInteger(string input) Positive case (415): 1 Negative case (45.21): 0 Negative case (-1): 0 Negative case (08): 0	Pass

FunctionHelper Unit Test

Test	Description	Expected Output	Passed
1	Check that DisplayMenu function can successfully display the menu	Test DisplayMenu function 1. The average wind speed and average ambient air temperature for a specified month and year. 2. Average wind speed and average ambient air temperature for each month of a specified year. 3. Total solar radiation in kWh/m2 for each month of a specified year. 4. Write average wind speed (km/h), average ambient air temperature and total solar radiation in kWh/m2 for each month of a specified year to CSV. 5. Show the times for the highest solar radiation for a date (d/m/yyy/) 6. Exit the program.	Pass
2	Check that FindAverageWindTempMonth can correctly carry out the function of option 1 of menu given map, month and year input	Test FindAverageWindTempMonth function for option 1 January 2010: 22.500000 km/h, 21.709999 degrees C	Pass
3	Check that FindAverageWindTempMonth can correctly carry out the function of option 2 of menu given map and year input	Test FindAverageWindTempMonth function for option 2 January 2010: 22.500000 km/h, 21.709999 degrees C February 2010: No Data March 2010: 21.600000 km/h, 18.940001 degrees C April 2010: No Data May 2010: No Data June 2010: No Data July 2010: No Data August 2010: 7.200000 km/h, 11.366000 degrees C September 2010: No Data October 2010: No Data November 2010: No Data December 2010: No Data	Pass

4	Check that FindSumSolarRadMonth can correctly carry out the function of option 3 of menu given map and year input	Test FindSumSolarRadMonth function for option 3 January 2010: 0.014473 kWh/m² February 2010: No Data March 2010: 0.015783 kWh/m² April 2010: No Data May 2010: No Data June 2010: No Data July 2010: No Data August 2010: 0.009472 kWh/m² September 2010: No Data October 2010: No Data November 2010: No Data December 2010: No Data	Pass
5	Check that OutputFileAverageWindTempSolarMonth can correctly carry out the function of option 4 of menu given map and year input	2010 January,22.500000,21.709999,0.014473 March,21.600000,18.940001,0.015783 August,7.200000,11.366000,0.009472 *in WindTempSolar.csv	Pass
6	Check that FindHighestSolarTimeByDate can correctly carry out the function of option 5 of menu given map, day, month and year input	Test FindHighestSolarTimeByDate function for option 5 Date: 1/1/2010 High solar radiation for the day: 0.151000W/m2 Time: 09:20 09:30	Pass

Main SIT (Using Test.csv)

Test	Description	Expected Output	Passed
1	Check that if user entered invalid file name in the index file the program will print error and continue with available data	Error while opening file: notfound.csv	Pass
2	Check that invalid month or year input while using function 1, it will result in error	Error: invalid month. Expects integer.	Pass
3	Check that invalid month and year combination	Error : invalid date.	Pass

	will result in error and stopping of program		
4	Check that function 1 can print correctly average wind speed and average air temperature when data is found for the specific month and year in km/h and degrees C respectively	January 2010: 22.500000 km/h, 21.709999 degrees C	Pass
5	Check that function 1 will show no data if no data is found for the specific month and year	February 2010: No Data	Pass
6	Check that if user entered invalid year while using function 2, it will result in error	Error: invalid year. Expects integer.	Pass
7	Check that function 2 can correctly print the average wind speed and average air temperature of all 12 months in the specific year in km/h and degrees C respectively and no data if no data is found for the month	January 2010: 22.500000 km/h, 21.709999 degrees C February 2010: No Data March 2010: 21.600000 km/h, 18.940001 degrees C April 2010: No Data May 2010: No Data June 2010: No Data July 2010: No Data August 2010: 7.200000 km/h, 11.366000 degrees C September 2010: No Data October 2010: No Data November 2010: No Data December 2010: No Data	Pass
8	Check that if user entered invalid year while using function 3, it will result in error	Error: invalid year. Expects integer.	Pass
9	Check that function 3 can correctly print the total solar radiation for each month for a specific year in kWh/m^2 and no data if no data is found for the month	2016 January 2016: No Data February 2016: No Data March 2016: 0.481833 kWh/m^2 April 2016: No Data May 2016: No Data June 2016: No Data July 2016: No Data August 2016: No Data September 2016: No Data October 2016: No Data November 2016: No Data December 2016: No Data	Pass

10	Check that if user entered invalid year while using function 4, it will result in error	Error: invalid year. Expects integer.	Pass
11	Check that function 4 can correctly output the average wind speed, average temperature and total solar radiation for each month of a specific year in km/h, degrees C and kWh/m2 and no line for months with no data	*Check WindTempSolar.csv 2010 January,22.500000,21.709999,0.014473 March,21.600000,18.940001,0.015783 August,7.200000,11.366000,0.009472	Pass
12	Check that if user entered an invalid date(dd/mm/yyyy) while using function 5, it will result in error	Error: Invalid date Error Data: 31/31/31 Accepted date format: dd/mm/yyyy	Pass
13	Check that function 5 can correctly output the highest solar radiation level of a given date and return a list of time with that solara radiation level	Date: 1/1/2010 High solar radiation for the day: 0.151000W/m2 Time: 09:20 09:30	Pass
14	Check that function 6 can quit the program	*Program exits	Pass
15	Check that inputting any function other than 1-6 will return in error and repeatedly display the menu	Error: Unknown command. Only numbers 1-6 accepted.	

Output of test run(s)

Date Unit Test

1)

```
Default Constuctor
Day: -1
Month: -1
Year: -1
```

```
Constructor
Day: 1
Month: 1
Year: 1
```

2)

```
Test Day setter
Day: 3
```

3)

```
Test Month setter
Month: 3
```

4)

```
Test Year setter
Year: 3
```

Time Unit Test

1)

```
Default Constructor
Hour: -1
Minute: -1
```

2)

```
Constructor
Hour: 1
Minute: 1
```

3)

```
Test Hour setter
Hour: 3
```

4)

```
Test Minute setter
Minute: 3
```

5)

```
Test < operator
Test Time 1: 05:05
Test Time 2: 03:03
Compare testTime1 < testTime2: 0
```

6)

```
Test > operator
Compare testTime1 > testTime2: 1
```

Weather Unit Test

1)

```
Check constructor
Date: -1/-1/-1
Time: 0-1:0-1
Wind Speed: -1
Solar Radiation-1
Air Temperature: -1
```

2)

```
Test Date setter
Date: 16/3/2016
```

3)

```
Test Time setter
Time: 09:15
```

4)

```
Test Wind Speed Setter
Wind Speed: 23.1
```

5)

```
Test Solar radiation Setter
Solar Radiation53.15
```

6)

```
Test Air temp setter
Air Temperature: 53.25
```

7)

```
Test < operator
Time of weather 1: 0-1:0-1
Time of weather 2: 09:15
Compare weather1 < weather2: 1
```

8)

```
Test > operator
Time of weather 1: 0-1:0-1
Time of weather 2: 09:15
Compare weather1 > weather2: 0
```

BstWeatherHelper Unit Test

1)

```
Test GetWeatherWindSpeed
23.1
```

2)

```
Test GetWeatherAirTemp
53.25
```

3)

```
Test GetWeatherSolarRad
53.15
```

4)

```
Test GetWeatherTimeBySolarRad
Found match: 09:15

Match not found:
```

BST Unit Test

1)

```
Test Constructor
0
```

2)

```
Test Copy constructor
Tree1 values: 1
Tree2 values after copying Tree1 and deleting Tree1: 1
```

3)

```
Test deconstructor
Tree1 values:
Tree2 values: 1
Tree2 values after decontructed:
```

4)

```
Test DeleteTree function
Tree1 values:
Tree1 values after deleting tree:
```

5)

```
Test = operator
Tree1 values: 1
Tree2 values after copying Tree1 and deleting Tree1: 1
```

6)

```
Test Search function
Value found: 1
Value not found: 0
```

7)

```
Test Insert function
Tree1 values:
After insert
Tree1 values: 1 2 3 4 5
```

8)

```
Test DeleteNode function
Tree1 values: 1 2 3 4 5
Deletes node
Tree1 values after delete: 2 3 4 5
```

9)

```
Test TreeNodeCount function
Count: 4
```

10)

```
Test InOrderTraversal function
1 2 3 4 5
```

11)

```
Test PreOrderTraversal function
2 1 5 4 3
```

12)

```
Test PostOrderTraversal function
2 1 5 4 3
```

13)

```
Test InOrderGetMaxFloat function
5
Test InOrderGetMaxFloat with function as parameter
33.1
```

14)

```
Test GetSumValueFloat function
15
Test GetSumValueFloat with function as parameter
69.3
```

15)

```
Test GetStringByFloatValue with function as parameter
09:15
09:15
09:15
```

LogicHelper Unit Test

1)

```
Test LoadMultiDataFileCSV function
Loading file: dataFile1.csv
Loaded data file: dataFile1.csv
Loading file: dataFile2.csv
Loaded data file: dataFile2.csv
1/1/2012 08:20
1/1/2012 08:30
1/1/2012 08:40
1/1/2012 08:50
1/1/2012 09:00
```

2)

```
Test LoadDataFileCSV function
17/8/2010 19:10
17/8/2010 19:20
17/8/2010 19:30
17/8/2010 19:40
17/8/2010 19:50
```

3)

```
Test InsertWeatherToMap function
16/3/2016 09:15
```

4)

```
Test RemoveLeadingZeros(string)
Test RemoveLeadingZeros(08): 8
Test RemoveLeadingZeros(08.9): 8.9
```

5)

```
Test IntMonthToString(unsigned int)
Test IntMonthToString(8): August
Test IntMonthToString(13):
Error converting int month to string
ERROR
```

6)

```
Test ConvertWindMStoKMH(float)
Test ConvertWindMStoKMH(10): 36
Test ConvertWindMStoKMH(10.5): 37.8
```

7)

```
Test ConvertSolarRadiationWMtoKWH(float)
Test ConvertSolarRadiationWMtoKWH(120): 0.02
Test ConvertSolarRadiationWMtoKWH(120.42): 0.02007
```

8)

```
Test AverageWindSpeedMonth(vector,int,int)
Test AverageWindSpeedMonth(TestLog,3,2016): 6.25
Test AverageWindSpeedMonth(TestLog,1,2020): -1
```

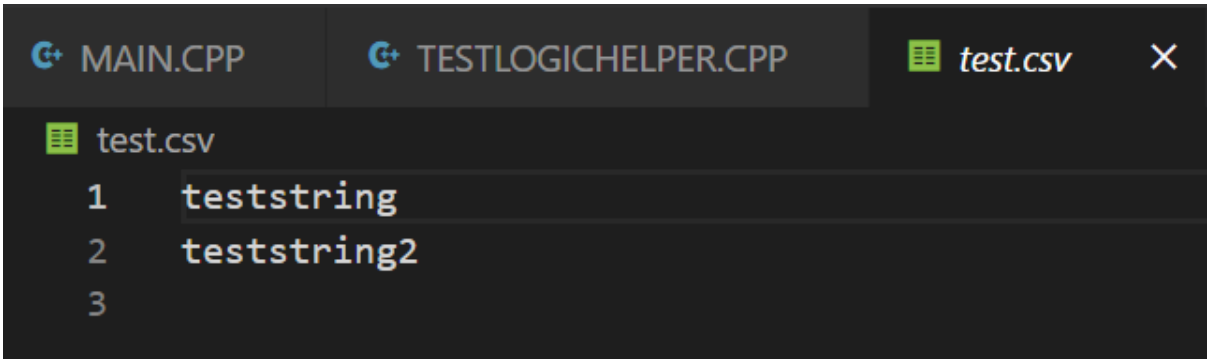
9)

```
Test AverageAirTemperatureMonth(vector,int,int)
Test AverageAirTemperatureMonth(TestLog,3,2016): 21.71
Test AverageAirTemperatureMonth(TestLog,1,2020): -1
```

10)

```
Test SumSolarRadiationMonth(vector,int,int)
Test SumSolarRadiationMonth(TestLog,3,2016): 86.84
Test SumSolarRadiationMonth(TestLog,1,2020): -1
```

11)



12)

```
Test HighestSolarRadiationDay function
906
```

13)

```
Test TimeOfSolarRadiationDay function
09:20
09:30
```

ValidationHelper Unit Test

1)

```
Checking IsValidDate(int day, int month, int year)
Positive case (29/2/2020): 1
Negative case, wrong day (31/2/2020):  0
Negative case, wrong month (29/13/2020): 0
Negative case, wrong year (29/2/99): 0
Negative case, not leap year (29/2/2022): 0
```

2)

```
Checking IsValidDate(int month, int year)
Positive case (2/2020): 1
Negative case, wrong month (13/2020): 0
Negative case, wrong year (2/99): 0
```

3)

```
Checking IsValidMonth(int month)
Positive case (2): 1
Negative case, wrong month (13): 0
Checking IsLeapYear(int year)
Positive case (2020): 1
Negative case (2022): 0
```

4)

```
Checking IsValid24HourTime(int hour, int minute)
Positive case (8,50): 1
Negative case, wrong hour (25,50): 0
Negative case, wrong minute (24,60): 0
```

5)

```
Checking IsNumber(string input)
Positive case (415): 1
Positive case (63.41): 1
Negative case, negative int (-1): 0
Negative case, negative float (-1.2): 0
Negative case, octal numbers (08): 0
```

6)

```
Checking IsInteger(string input)
Positive case (415): 1
Negative case (45.21): 0
Negative case (-1): 0
Negative case (08): 0
```

FunctionHelper Unit Test

1)

```
Test DisplayMenu function
1. The average wind speed and average ambient air temperature for a specified month and  year.
2. Average wind speed and average ambient air temperature for each month of a specified year.
3. Total solar radiation in kWh/m2 for each month of a specified year.
4. Write average wind speed (km/h), average ambient air temperature and total solar radiation in kWh/m2 for eac
h month of a specified year to CSV.
5. Show the times for the highest solar radiation for a date (d/m/yyy/)
6. Exit the program.
```

2)

```
Test FindAverageWindTempMonth function for option 1
January 2010: 22.500000 km/h, 21.709999 degrees C
```

3)

```
Test FindAverageWindTempMonth function for option 2
January 2010: 22.500000 km/h, 21.709999 degrees C
February 2010: No Data
March 2010: 21.600000 km/h, 18.940001 degrees C
April 2010: No Data
May 2010: No Data
June 2010: No Data
July 2010: No Data
August 2010: 7.200000 km/h, 11.366000 degrees C
September 2010: No Data
October 2010: No Data
November 2010: No Data
December 2010: No Data
```

4)

```
Test FindSumSolarRadMonth function for option 3
January 2010: 0.014473 kWh/m²
February 2010: No Data
March 2010: 0.015783 kWh/m²
April 2010: No Data
May 2010: No Data
June 2010: No Data
July 2010: No Data
August 2010: 0.009472 kWh/m²
September 2010: No Data
October 2010: No Data
November 2010: No Data
December 2010: No Data
```

5)

WindTempSolar.csv

```
2010
January,22.500000,21.709999,0.014473
March,21.600000,18.940001,0.015783
August,7.200000,11.366000,0.009472
```

6)

```
Test FindHighestSolarTimeByDate function for option 5
Date: 1/1/2010
High solar radiation for the day: 0.151000W/m2

Time:
09:20
09:30
```

Main SIT

1)

```
Error while opening file: notfound.csv
```

2)

```
Input month to search(1-12)
k
Error: invalid month. Expects integer.
```

3)

```
Input month to search(1-12)
13
Input year to search
9123
Error: invalid date.
```

4)

```
Input month to search(1-12)
1
Input year to search
2010
January 2010: 22.500000 km/h, 21.709999 degrees C
```

5)

```
Input month to search(1-12)
2
Input year to search
2010
February 2010: No Data
```

6)

```
Input year to search(YYYY)
rw
Error: invalid year. Expects integer.
```

7)

```
January 2010: 22.500000 km/h, 21.709999 degrees C
February 2010: No Data
March 2010: 21.600000 km/h, 18.940001 degrees C
April 2010: No Data
May 2010: No Data
June 2010: No Data
July 2010: No Data
August 2010: 7.200000 km/h, 11.366000 degrees C
September 2010: No Data
October 2010: No Data
November 2010: No Data
December 2010: No Data
```

8)

```
Input year to search
12345
Error: invalid year. Expects integer.
```

9)

```
January 2010: 0.014473 kWh/m²
February 2010: No Data
March 2010: 0.015783 kWh/m²
April 2010: No Data
May 2010: No Data
June 2010: No Data
July 2010: No Data
August 2010: 0.009472 kWh/m²
September 2010: No Data
October 2010: No Data
November 2010: No Data
December 2010: No Data
```

10)

```
Input year to search
ytr
Error: invalid year. Expects integer.
```

11)

WindTempSolar.csv

```
2010
January,22.500000,21.709999,0.014473
March,21.600000,18.940001,0.015783
August,7.200000,11.366000,0.009472
```

12)

```
Error: Invalid date
Error Data: 31/31/31
Accepted date format: dd/mm/yyyy
```

13)

```
Date: 1/1/2010
High solar radiation for the day: 0.151000W/m2

Time:
09:20
09:30
```

14)

```
Error: Unknown command. Only numbers 1-6 accepted.
```

Evaluation

Refer to txt document “Evaluation.txt”