

**Student ID: 34315323**

**Student name: Tee Yee Kang**

**Class: FT MUR T321 ICT283 A**

**Project name: Assignment 2**

# 1. Rationale

The data structures I used for the assignment 2 program are the template binary search tree (bst) and STL map. The vector is not used in assignment 2.

In assignment 1, I used the vector to store all data after reading the value from the data file. However, in assignment 2, I change the data structure to map instead of vector.

## The way I store data in the STL map:

- I create a struct that contains a Date class object and a Time class object. (called it dateTime)
- Store the dateTime struct object as the key of the map
- A WindLogInfo class object (weather) contains the value of average wind speed, average air temperature and solar radiation.
- The value of the map will be the WindLogInfo class object.

One of the reasons I used the STL map to store all data is to fulfil the requirement of assignment 2. Another reason is that the STL map does not allow duplicate key. Therefore, the map will handle all the duplicate records for me. In addition, each key (dateTime) will have a WindLogInfo class object (value) that contains all the required information. When calculating the required data for options 1-5, once I found the same dateTime object (key), I can simply retrieve the corresponding value. I think this approach/structure is more suitable and easier to understand and compile than the bst. That's also the reason why I decided to use STL map to store data instead of bst.

## The way I use bst:

- I used bst in the calculation of average wind speed, average air temperature and total solar radiation (GetAverageSpeed(), GetAverageAir() and GetTotalSolar()). I will pass in the STL map to the methods that do the calculation. After I prompt and get the date from the user, I will use for loop to search through the pass-in map and find the same key. Once I found the same key, I retrieved and stored the required data into a bst. My traversal method will call the function pointer and calculate the running total of all nodes and find max value (option 5) in the bst. I used bst to store and calculate the data because I wanted to make use of the method of bst, which is the function pointer and traversal. For example, I have a function called GetTotal() in CollectData class, which I will pass to the inorder traversal to calculate the running total of every node in bst. It is because this approach/structure is easier and faster when calculation the running total. In addition, my program does not need to use a loop to go through every record and calculate the running total because my bst will handle it for me. After completing the running total calculation, I will conduct other calculations and return the final value.

### **Another way I use bst:**

- The requirement also requires that our program must have the ability to ignore duplicate data files. Therefore, I create a bst with string data type in the ReadDataFromFile() method. After I read and open a particular CSV file in the data folder, I will first check if my bst contains this file name. If the file name is not found in the bst (means no duplicate), I will store the particular file name into the bst and continue other operations. If found in the bst file (means duplicate), my program will skip everything in the if statement. I use bst because the searching performance of bst is  $O(\log_2 n)$  time. In worst case, the time it takes to search an element is  $O(n)$ . Which I think is better than using the array.
- I use the file.eof() and break in the middle of the code inside my program instead of the beginning. It is because the eof() will always read an extra line even though there is no more data inside the met\_index.txt file. Therefore, my program will check if there is more data in the file during the process instead of the beginning.
- I use struct to store Date class object and Time class object (dateTime) is because I consider this is just a combine package of Date and Time class object.

### **The way I handle global variable issue:**

- Initially I created a function in main program to calculate the sum of all value in bst and another function to compare and find the largest value in bst. However, as mentioned in the lecturer, global variable is considered not good. Therefore, to resolve this issue I follow the given example in the testTree file which I create 2 new separate class to retrieve value from bst and calculate the require data. The first class is called Value. This class object contains a member variable, and it is used to store the value retrieve from bst. For example, I create a Value type bst and Value class object in GetAverageSpeed () function. Once I found the same key record, I will first set the value of wind speed to the value of Vector class object by using the SetValue() method. After that I will store this Value class object in the Value type bst. Therefore, my bst will contains many Value class objects for later use.
- Once the looping of map is complete, I will use the inorderTraversal() function and pass in the Sum() function in CollectData class object. The Sum() function will take in a Value class object and calculate the running total by retrieving the value of Value class object (using the GetValue() function). The bst will traverse and continue this operation, to calculate the running total and store it in the member variable m\_total. Once the traversal is completed, I will simply use the GetTotal() function to retrieve the value of running total. After each call is completed, the destructor of Value class and CollectData class will set the member variable back to 0, so that the Value and CollectData class object can reuse.

- The calculation of average air temperature and total solar radiation are same.
- The calculation of highest solar radiation (option 5) is similar to this approach. The collectData class also contains another member variable called m\_max and static function called Max. Which is used to find and store the max solar radiation. The Max() function will take in a Value class object and compare with the member variable m\_max to find the larger value. Once the program found a larger value, the program will store the larger value to the m\_max. Once the comparison is completed, the program will use the GetMax() function to return the highest solar radiation.

## 2. Data Dictionary

### Date Class Attributes

Name	Type	Protection	Description	Rationale
Date				Simulate the date
m_day	integer	- (Private)	The day of the date in Date class	The value of day is read from the given csv file. The format also based on the given csv file. Use integer data type because the value of the date is limited. Use private protection because did not want the user to direct access the member variables of the Date class object.
m_month	integer	- (Private)	The month of the date in Date class	The value of month is read from the given csv file. The format also based on the given csv file. Use integer data type because the value of the date is limited. Use private protection because did not want the user to direct access the member variables of the Date class object.
m_year	integer	- (Private)	The year of the date in Date class	The value of year is read from the given csv file. The format also based on the given csv file. Use integer data type because the value of the date is limited. Use private protection because did not want the user to direct access the member variables of the Date class object.

## Date Class Methods

Name	Type	Protection	Description	Rationale
Date()	procedure	+ (Public)	Constructor of the Date class	This is essential for the class object. User can create Date class object with default values (member variables)
Date(const int day, const int month, const int year)	procedure	+ (Public)	Constructor of the Date class with parameter	This is essential for the class object. User can create Date class object with passed in values (member variables)
Clear()	void	+ (Public)	Functions used to clear/set default value for Date class object.	Create a separate function to set default values because other functions can reuse this function also, not only constructor can use.
GetDate() const	void	+ (Public)	Get the date of Date class object	The user can use this function to return all the value of Date class object, instead of separate day, month and year functions.
SetDate(int day, int month, int year)	void	+ (Public)	Set all values for Date class object	The user can use this function to set all the values of member variables, instead of setting on by one.
GetDay() const	integer	+ (Public)	Get the day of Date class object	Simple getter functions
SetDay(int newDay)	void	+ (Public)	Set the day of Date class object	Simple setter functions
GetMonth() const	integer	+ (Public)	Get the month of Date class object	Simple getter functions
SetMonth(int newMonth)	void	+ (Public)	Set the month of Date class object	Simple setter functions
GetYear() const	integer	+ (Public)	Get the year of Date class object	Simple getter functions
SetYear(int newYear)	void	+ (Public)	Set the year of Date class object	Simple setter functions
GetMonthInString() const	string	+ (Public)	Get the month of Date class object with description	Can output the value of month in English word form instead of numerical value – look nicer.
ValidDate(int day, int month, int year)	boolean	+ (Public)	Check if the date passed in is valid.	This is an essential function which use to ensure the date is valid. I create a separate function so that this function can be reuse in the constructor and all setter methods.

### Date Class Methods Continue ...

Name	Type	Protection	Description	Rationale
IsLeapYear(int year)	boolean	+ (Public)	Check for leap year.	This is also essential function which use to check for leap year. I create in separate function so that the ValidDate() function not be too long or too complicated.
GetDateObj()	Date	+ (Public)	Return Date class object in Date data type	This getter function is used to return the whole Date class object with Date data type. The previous few functions only return day of Date, month of Date, etc.

### Time Class Attributes

Name	Type	Protection	Description	Rationale
Time				Simulate the time
m_hour	integer	- (Private)	The hours of Time class object	The value of hour is read from the given csv file. The format also based on the given csv file. Use integer data because the value of the time is limited. Use private protection because did not want the user to direct access the member variables of the Time class object.
m_minute	integer	- (Private)	The minutes of Time class object	The value of minute is read from the given csv file. The format also based on the given csv file. Use integer data type because the value of the time is limited. Use private protection because did not want the user to direct access the member variables of the Time class object.

### Time Class Methods

Name	Type	Protection	Description	Rationale
Time()	procedure	+ (Public)	Constructor of the Time class	This is essential for the class object. User can create Time class object with default values (member variables)
Time(const int hour, const int minute)	procedure	+ (Public)	Constructor of the Time class with parameter	This is essential for the class object. User can create Time class object with passed in values (member variables)
Clear()	void	+ (Public)	Functions used to clear/set default value for Time class object.	Create a separate function to set default values because other functions can reuse this function also, not only constructor can use.
GetHour() const	integer	+ (Public)	Get the hour of Time class object	Simple getter functions
SetHour(int hour)	void	+ (Public)	Set the hour of Time class object	Simple setter functions
GetMinute() const	integer	+ (Public)	Get the minute of Time class object	Simple getter functions
SetMinute(int minute)	void	+ (Public)	Set the minute of Time class object	Simple setter functions



### WindLogInfo Class Attributes

Name	Type	Protection	Description	Rationale
WindLogInfo				Aggregate all require data and keep in a single class. Looks cleaner and more concise
m_speed	float	- (Private)	The speed of WingLogInfo class object	The value of speed is read from the given csv file. Data type float is mentioned in the question. Use private protection because did not want the user to direct access the member variables of the WingLogInfo class object.
m_solar	float	- (Private)	The solar radiation of WingLogInfo class object	The value of solar radiation is read from the given csv file. Use float data type because it is easier for calculation of average. Use private protection because did not want the user to direct access the member variables of the WingLogInfo class object.
m_air	float	- (Private)	The ambient air temperature of WingLogInfo class object	The value of ambient air temperature is read from the given csv file. Use float data type because it contains floating value. Use private protection because did not want the user to direct access the member variables of the WingLogInfo class object.

## WindLogInfo Class Methods

Name	Type	Protection	Description	Rationale
WindLogInfo()	procedure	+ (Public)	Constructor of the Time class	This is essential for the class object. User can create WindLogInfo class object with default values (member variables)
WindLogInfo(const float speed, const float solar, const float air)	procedure	+ (Public)	Constructor of the WindLogInfo class with parameter	This is essential for the class object. User can create WindLogInfo class object with passed in values (member variables)
Clear()	void	+ (Public)	Functions used to clear/set default value for WindLogInfo class object.	Create a separate function to set default values because other functions can reuse this function also, not only constructor can use.
GetSpeed() const	float	+ (Public)	Get the speed of WindLogInfo object	Simple getter functions
SetSpeed(float newSpeed)	void	+ (Public)	Set the speed of WindLogInfo object	Simple setter functions
GetSolar() const	float	+ (Public)	Get the solar radiation of WindLogInfo object	Simple getter functions
SetSolar(float newSolar)	void	+ (Public)	Set the solar radiation of WindLogInfo object	Simple setter functions
GetAir() const	float	+ (Public)	Get the ambient air temperature of WindLogInfo object	Simple getter functions
SetAir(float newAir)	void	+ (Public)	Set the ambient air temperature of WindLogInfo object	Simple setter functions

### Value Class Attributes

Name	Type	Protection	Description	Rationale
Value				Used to retrieve and temporary store the value from the bst by using with the traversal and function pointer.
m_value	float	- (Private)	The value retrieve from the bst.	The program will store the WindLogInfo (weather) data into the bst. Then the program will retrieve and store the value in to this class during the calculation of average wind speed and so on.

### Value Class Methods

Name	Type	Protection	Description	Rationale
~Value()	procedure	+ (Public)	Reset the value of member variable m_value	This function is used to set the value of member variable m_value to 0; So that the class object can be reuse.
GetValue()	float	+ (Public)	Get the value of member variable m_value.	Simple getter functions
SetValue(float number)	void	+ (Public)	Set the value of member variable m_value.	Simple setter functions

### CollectData Class Attributes

Name	Type	Protection	Description	Rationale
CollectDate				This class is used to calculate and store the running total of all value in bst. In addition, this method of this class also used to find and store the largest value in bst.
m_total	float	- (Private)	The running total of all value in bst	To calculate the running total of all value in bst will require a function (pass as function pointer) and traversal method of bst. This class is implemented so that the program can calculate the running total without using the global variable. This member variable will used to temporarily store the value running total.
m_max	float	- (Private)	The largest value in the bst.	This member variable is used to temporarily store the largest value store in the bst during the comparison and the traversal.

### CollectData Class Methods

Name	Type	Protection	Description	Rationale
~CollectData();	procEDURE	+ (Public)	Destructor of the class. Reset the value of both member variables.	This method is used to set the value of both member variables to the default value, so that the class object can be use.
Sum(Value &data)	void	+ (Public)	Calculate the running total.	This method will add the pass in value (during the traversal) and store the running total to the member variable m_total.
Max(Value &data)	Void	+ (Public)	Find and retrieve the largest value.	This method is use to find and retrieve the largest value by comparing the pass in value with the member variable m_max. Once found a larger value, set member variable m_max to the largest value.
GetTotal()	Void	+ (Public)	Get the value of m_total.	Simple getter method.
GetMax()	Void	+ (Public)	Get the value of m_max.	Simple getter method.

## bst Class Attributes

Name	Type	Protection	Description	Rationale
bst				This is the template binary search tree (function pointer approach)
root	Node<T> pointer	- (Private)	The node of bst.	This is the Node of the bst. Encapsulate so that the user cannot directly access the node.
CopyTree(Node<T>* &copiedTreeRoot, Node<T>* otherTreeRoot)	void	- (Private)	Copy constructor of bst class	This is essential for bst class. Because the bst class contains pointer.
Destroy(Node<T>* &p)	void	- (Private)	Clear all value in the bst.	This is used to remove/clear all value in the bst. This is also a essential method for bst class.
Inorder(funcP fp, Node<T>*p)	void	- (Private)	Private version of the inorder traversal method.	This is also essential for the bst class. Purpose is used to avoid user directly access the private member variable node.
Preorder(funcP fp, Node<T>*p)	void	- (Private)	Private version of the preorder traversal method.	This is also essential for the bst class. Purpose is used to avoid user directly access the private member variable node.
Postorder(funcP fp, Node<T>*p)	void	- (Private)	Private version of the postorder traversal method.	This is also essential for the bst class. Purpose is used to avoid user directly access the private member variable node.
SearchElement(Node<T>* nd, const T& item)	void	- (Private)	Private version of the search method.	This is also essential for the bst class. Purpose is used to avoid user directly access the private member variable node. This method is used to search an item from bst.
InsertElement(Node<T>* &nd, T item)	void	- (Private)	Private version of the insert method.	This is also essential for the bst class. This method is used to insert item into the bst.

## bst Class Methods

Name	Type	Protection	Description	Rationale
(*funcP)(T &)	Void	+ (Public)	Function pointer	This is the function pointer for the traversal methods, which is used to perform some task during the traversal.
InsertItem(const T& insertItem)	void	+ (Public)	Public version of the insert method.	This is an essential method for bst. This is function is for the user to insert new item into the bst.
SearchItem(const T& searchItem)	void	+ (Public)	Public version of the search method.	This is an essential method for bst. This is function is for the user to search for an item from the bst.
DestroyTree()	void	+ (Public)	Public version of the destroy method.	This is an essential method for bst. This is function allowed user to remove all items in the bst.
InorderTraversal(funcP fp)	void	+ (Public)	Inorder traversal method for bst.	This is an essential method for bst. This is function allowed user to traverse (inorder sequence) through all items in the bst.
PreorderTraversal(funcP fp)	void	+ (Public)	Preorder traversal method for bst.	This is an essential method for bst. This is function allowed user to traverse (preorder sequence) through all items in the bst.
PostorderTraversal(funcP fp)	void	+ (Public)	Postorder traversal method for bst.	This is an essential method for bst. This is function allowed user to traverse (postorder sequence) through all items in the bst.
binaryTreePointer()	procedure	+ (Public)	Constructor of the bst	This is an essential method for bst. This method allowed user to create an empty bst.
~binaryTreePointer()	procedure	+ (Public)	Destructor of the bst	This is an essential method for bst.
binaryTreePointer(const binaryTreePointer<T>& otherTree)	procedure	+ (Public)	Copy constructor of bst.	This is an essential method for bst. This method allowed user to create a bst by copying the value from other bst.

## Main.cpp methods

Name	Type	Protection	Description	Rationale
main				main execution file
Menu()	integer		Display the menu description and prompt user for option to be performed	Implement a separate function so that the main class can look cleaner (not many lines of code about the menu within the main class).
Controller( const map<dateTime, WindLogInfo> &mp);	void		The main controller of the whole program	Use this function to call all the other function. Therefore, inside the main class just have to call this function in order to run the whole program. The main class can look cleaner as well.
GetAverageSpeed(const map<dateTime, WindLogInfo> &mp, int year, int month);	float		A function used to calculate average wind speed store in map object for specific year and month.	Option 1, 2 and 4 all required to calculate average wind speed. Therefore, this separate function can be reuse instead of writing the code to calculate in every option. In addition, it can also be reused for another map object.
GetAverageAir(const map<dateTime, WindLogInfo> &mp, int year, int month)	float		A function used to calculate average ambient air temperature store in map for specific year and month.	Option 1, 2 and 4 all required to calculate average ambient air temperature. Therefore, this separate function can be reuse instead of writing the code to calculate in every option. In addition, it can also be reused for another map object.



Name	Type	Protection	Description	Rationale
GetTotalSolar(const map<dateTime, WindLogInfo> &mp, int year, int month)	float		A function used to calculate total solar radiation store in map object for specific year and month.	Both option 3 and 4 required to calculate total solar radiation. Therefore, this separate function can be reuse instead of writing the code to calculate in every option. In addition, it can also be reused for another map object.
Option1(const map<dateTime, WindLogInfo> &mp)	void		Controller for option 1 – Calculate average wind speed for a specified month and year	A separate function to perform option 1. Therefore, the function for calculate average wind speed can be reuse (because the printout result part only contains in this function). The whole function can look cleaner and more concise.
Option2(const map<dateTime, WindLogInfo> &mp)	void		Controller for option 2 – Calculate average wind speed and average ambient air temperature for each month of a specified year	A separate function to perform option 2. Therefore, the function for calculate average wind speed and average ambient air temperature can be reuse (because the printout result part and for-loop to loop through each month only contains in this function). The whole function can look cleaner and more concise.
Option3(const map<dateTime, WindLogInfo> &mp)	void		Controller for option 3 – Calculate total solar radiation for each month of a specified year	A separate function to perform option 3. Therefore, the function for calculate total solar radiation can be reuse (because the printout result part and for-loop to loop through each month only contains in this function). The whole function can look cleaner and more concise.

Name	Type	Protection	Description	Rationale
Option4(const map<dateTime, WindLogInfo> &mp);	void		Controller for option 4 – Output all require data into a csv file	A separate function to perform option 4. Therefore, all the functions for calculation can be reused. The whole function can look cleaner and more concise.
GetUserYear()	integer		This function is used to prompt and read which year the user wanted to use for further calculations.	A separate function to get the year from the user. Therefore, this function can be used for other task or option that will be required in the future.
GetUserMonth()	integer		This function is used to prompt and read which month the user wanted to use for further calculations.	A separate function to get the year from the user. Therefore, this function can be used for other task or option that will be required in the future.
ReadDataFromFile(map<dateTime, WindLogInfo> &mp)	void		A function used to read data from file and create corresponding class object and pass into the map.	A separate function for reading data from file. Therefore, this function can be reuse if needed. After read data, all objects will store into the pass in map. Hence, this function can be used for another map object as well. Inside the function body, I implement my own boolean variable endOfFile to check for the end of csv file. The reason why is because if I only use the .eof at the beginning, the last record of data will read twice/duplicate. Therefore, my while loop will use my own boolean to check the looping condition.

Name	Type	Protection	Description	Rationale
Option5(const map<dateTime, WindLogInfo> &mp)	void		Controller for option 5 – Show the times for the highest solar radiation for a specific Date.	A separate function to perform option 5. Therefore, all the functions for calculation can be reused. The whole function can look cleaner and more concise.
isNumeric(string input)	bool		Used to check if the user enters a valid input (numeric option).	A separate function to check for invalid input. Therefore, this function can be reuse and the Menu() function can look cleaner and more concise.
GetUserDate()	Date		Get the date to perform some tasks from the user.	This function is for option 5. It is used to prompt and get a data from user. A separate method so that it can be reuse if needed for any additional option.

### 3. Algorithm (Pseudocode) **\*\*red color is pseudocode**

#### main.cpp

#### 1. Menu()

- Used to display the description of each option and prompt user for the number of option to be performed.

```
menu
  Display description of menu
  Prompt user for option
  Read option from user
  Return option
  Check if it is valid input
END
```

#### 2. Controller(const map<dateTime, WindLogInfo> &mpg)

- The main controller of the whole program. Used to call all other functions. Will keep execute until the user enter option 5 to terminate the program.

```
controller(map)
  Set userChoice to 0;
  DO
    CASE OF (userChoice)
      1. Execute option 1
      2. Execute option 2
      3. Execute option 3
      4. Execute option 4
      5. Terminate the program
      Other: Display invalid message
    WHILE (userChoice != 5)
  ENDDO
END
```

### 3. ReadDataFromFile (map<dateTime, WindLogInfo> &mp)

- This function is used to read data from the csv file and create corresponding class object and struct and store into the passed in map.

readDataFromFile (map)

Set variable theSpeed, theSolar, theAir to float type

Set name, day, month, year, hour, minute, speedInString, solarInString, airInString, dummy

FileToRead, filename, strDate to string type

Set err to 0, invalidFile 0, numOfFileRead;

Set and initialize NA to "NA"

Create WindLogInfo object – wObject

Create struct dateTime – dt

Create a bst with string type

Open file "data/met\_index.txt"

WHILE (true)

Read file to read

Set file name

IF( reach end of file)

Break

ENDIF

Open file to read

IF( file to read is open)

IF(file to read not duplicate)

Increase numOfFileRead by 1

Insert file name to bst

Skip first header line

WHILE (true)

Read day from file

Read month from file

Read year from file

Read hour from file

Read minute from file

Convert day to int data type

Convert month to int data type

Convert year to int data type

Convert hour to int data type

Convert minute to int data type

Set Date class object

Set Time class object

FOR (idx=0; idx < 10; idx increment by 1)

Read wind speed from file

END FOR LOOP

Read solar radiation from file

FOR (idx=0; idx < 5; idx increment by 1)

Read unuse data from file

END FOR LOOP

```
Read ambient air temperature from file
IF (reach end of file)
    break
ELSE
    IF(speed no contain NA)
        Convert wind speed to float type
    ELSE
        Set wind speed to 0
    ENDIF
    IF(solar radiation no contain NA)
        Convert solar radiation to float type
    ELSE
        Set solar radiation to 0
    ENDIF
    IF(ambient air temperature no contain NA)
        Convert ambient air temperature to float type
    ELSE
        Set ambient air temperature to 0
    ENDIF
    Set wind speed
    Set solar radiation
    Set ambient air temperature
    Store WindLogInfo object to pass in map
ENDIF
Print message
Close file
END
```

4. `GetAverageSpeed(const Vector<WindLogType> &windlog, int year, int month);`
- This function is used to calculate and return the average wind speed for specific month and year.

```
GetAverageSpeed (Vector, year, month)
    Set sumOfSpeed to 0
    Set speedAvg to 0
    Set counter to 0
    Create a bst with Value type
    Create a Value class object
    Create a CollectData class object
    FOR (auto temp:mp)
        IF (temp year and month both = to pass in year and month)
            Set wind speed to Value class object
            Store value class object to the bst
            counter increment by 1
        ENDIF
    Calculate average wind speed and store in speedAvg
    Destroy the bst
    Return speedAvg
END
```

5. GetAverageAir(const map<dateTime, WindLogInfo> &mp, int year, int month)

- This function is used to calculate and return the average air temperature for specific month and year.

GetAverageAir (map, year, month)

Set sumOfAir to 0

Set airAvg to 0

Set counter to 0

Create a bst with Value type

Create a Value class object

Create a CollectData class object

FOR (auto temp:mp)

IF (temp year and month both = to pass in year and month )

Set air temperature to Value class object

Store value class object to the bst

counter increment by 1

ENDIF

Calculate average ambient air temperature and store in speedAvg

Destroy the bst

Return airAvg

END



6. GetTotalSolar(const map<dateTime, WindLogInfo> &mp, int year, int month)

- This function is used to calculate and return the total solar radiation for specific month and year.

GetTotalSolar (map, year, month)

Set sumOfSolarRadiation to 0

Set hourlySolarRadiation to 0

Set finalSolarRadiation to 0

Create a bst with Value type

Create a Value class object

Create a CollectData class object

FOR (auto temp:mp)

IF (Value of temp solar radiation > 100)

Set solar radiation to Value class object

Store value class object to the bst

ENDIF

sumOfSolarRadiation / 6 and store in hourlySolarRadiation

hourlySolarRadiation / 1000 and store in finalSolarRadiation

Destroy the bst

Return finalSolarRadiation

END

7. option1(const map<dateTime, WindLogInfo> &mp)

- The function is the controller of option 1.

Option1 (map)

Get year to calculate from user

Get month to calculate from user

Get average wind speed for the year and month get from user

Get average ambient air temperature for the year and month get from user

Create a Date class object and set the month

IF (average wind speed and average ambient air temperature are > 0)

Print out the output base on requirement

ELSE

Print out the no data output

ENDIF

END

8. option2(const map<dateTime, WindLogInfo> &mp)

- The function is the controller of option 2.

Option2 (map)

```
Get year to calculate from user
Create a Date class object
Set month
For (month =1; month < 13; increment month by 1)
    Get average wind speed for specific month of year
    Get average ambient air temperature for specific month of year
    Get month in English form
    IF ( average wind speed and average ambient air temperature >0)
        Print the output based on the requirement
    ELSE
        Print no data output
    ENDIF
END
```

9. option3(const map<dateTime, WindLogInfo> &mp)

- This function is the controller for option 3

Option3 (map)

```
Get year to calculate from user
Create a Date class object
Set month
For (month =1; month < 13; increment month by 1)
    Get total solar radiation for specific month of year
    Get month in English form
    IF ( total solar radiation > 0)
        Print the output based on the requirement
    ELSE
        Print no data output
    ENDIF
END
```

#### 10. Option4(const map<dateTIme, WindLogInfo> &mp)

- This function is the controller for option 4

##### Option4 (map)

```
Open the output file to write output
Set decimal point to 1
Get year to calculate from user
Set month
Set hasData to false
FOR (auto temp:mp)
    IF (temp year == user's year)
        Set hasData to true
    ENDIF
ENDIF
IF(hasData == true)
    FOR (month=1; month<13; increment month by 1)
        Get average wind speed for specific month of year
        Get average ambient air temperature for specific month of year
        Get total solar radiation for specific month of year
        Get month in English form
        IF(average wind speed, ambient air temperature and solar radiation >0)
            Write the require output to the output file
        ENDIF
    ENDIF
ELSE
    Write no data to output file
ENDIF
Close the output file
Print successful message to the screen
END
```

11. int GetUserYear();

- This function is used to get the year to calculate data from user

```
GetUserYear ()  
    Set year  
    Prompt user for year  
    Read year from user  
    Return year  
END
```

12. int GetUserMonth ();

- This function is used to get the month to calculate data from user

```
GetUserMonth ()  
    Set month  
    Prompt user for month  
    Read month from user  
    Return month  
END
```

13. int GetUserDate ();

- This function is used to get the Date in Date data type

```
GetUserDate ()  
    Create Date object  
    Prompt user for Date in specific format  
    Read Date from user  
    Return Date  
END
```

#### 14. Option5(const map<dateTime, WindLogInfo> &mp)

- This function is the controller for option 5

##### Option5 (map)

```
Create time class object
Set boolean exist to false
Get Date to calculate from user
Create a Date class object
Create a bst with Value type
Create a Value class object
Create a CollectData class object
For (auto temp:mp)
    IF( temp's date is equal to user date AND temp's solar radiation > 0)
        Set wind speed to Value class object
        Store value class object to the bst
    ENDIF
Find max solar radiation
Display output
For (auto temp:mp)
    IF(temp's date equal to user data AND temp's solar is equal to max solar)
        Display time of the record
        Set exist to true
    ENDIF
IF (exist == false)
    Display no record message
ENDIF
Destroy the bst
END
```

## Date.cpp

### 1. Date()

- Date class constructor

```
Date()
    Call clear() function to set member variables to default value
END
```

### 2. Date(const int date, const int month, const int year)

- Date class constructor with parameters

```
Date(day, month year)
    IF ( is valid date )
        Set member variable day to day
        Set member variable month to month
        Set member variable year to year
    ELSE
        Call clear() function to set member variables to default value
    ENDIF
END
```

### 3. Clear()

- This function is used to set all member variables to default value

```
Clear()
    Set member variable day to 0
    Set member variable month to 0
    Set member variable year to 0
END
```

### 4. GetDate()

- This function is used to print out the date in a proper format

```
GetDate()
    Print the date to the screen
END
```

#### 5. SetDate(int date, int month, int year)

- This function is the setter function to set Date class object to new passed in value

```
SetDate(day, month, year)
  IF ( is valid date )
    Set member variable day to day
    Set member variable month to month
    Set member variable year to year
  ELSE
    Print invalid date message
  ENDIF
END
```

#### 6. SetDay(int newDay)

- This function is the setter function, used to set new value for member variables day

```
SetDay (day)
  IF (is valid date)
    Set member variable day to day
  ELSE
    Print error message to screen
  ENDIF
END
```

#### 7. SetMonth(int newMonth)

- This function is the setter function and used to set new value for member variables month

```
SetMonth(month)
  IF (is valid date)
    Set member variable month to month
  ELSE
    Print error message to screen
  ENDIF
END
```

#### 8. SetYear(int newYear)

- This function is the setter function and used to set new value for member variables year

```
SetYear (year)
    Set member variable year to year
END
```

#### 9. GetDay()

- This function is the getter function, used to get the value of day

```
GetDay ()
    Return the day of Date class object
END
```

#### 10. GetMonth()

- This function is the getter function, used to get the value of month

```
GetMonth ()
    Return the month of Date class object
END
```

#### 11. GetYear()

- This function is the getter function, used to get the value of year

```
GetYear ()
    Return the year of Date class object
END
```

#### 12. IsLeapYear(int year)

- This function is used to check for leap year.

```
IsLeapYear(year)
    Set leapYear = false
    IF (year % 4 == 0) AND (year % 100 != 0) OR (year % 400 == 0)
        Set leapYear = true;
    ENDIF
    Return leapYear
END
```



### 13. GetMonthInString() const

- This function is used to return the month in string and English form

GetMonthInString()

Set monthDesc

CASE OF (member variable month)

1. Set monthDesc to "January"
2. Set monthDesc to "February"
3. Set monthDesc to "March"
4. Set monthDesc to "April"
5. Set monthDesc to "May"
6. Set monthDesc to "June"
7. Set monthDesc to "July"
8. Set monthDesc to "August"
9. Set monthDesc to "September"
10. Set monthDesc to "October"
11. Set monthDesc to "November"
12. Set monthDesc to "December"

DEFAULT: Set monthDesc to "Invalid data"

Return monthDesc

END

#### 14. ValidDate(int day, int month, int year)

- This function is used to check whether the date passed in by the user is valid.

```
ValidDate(day, month, year)
  Set valid = false
  IF (month == 1 OR 3 OR 5 OR 7 OR 8 OR 10 OR 12)
    IF (day >=0 AND day <=31)
      Set valid = true
    ENDIF
  ELSE IF (month == 4 OR 6 OR 9 OR 11)
    IF (day >=0 AND day <=30)
      Set valid = true
    ENDIF
  ELSE IF (month == 2)
    IF (year is leap year)
      IF (day >=0 AND day <=29)
        Set valid = true
      ENDIF
    ELSE
      IF (day >=0 AND day <=28)
        Set valid = true
      ENDIF
    ENDIF
  ENDIF
ENDIF
END
```

#### 15. GetDateObj() const

- This function is used to return the Date object.

```
ValidDate(day, month, year)
  Create a Date class object
  Return Date class object
END
```

## Time.cpp

### 1. Time()

- Time class constructor

```
Time()  
    Call clear() function to set member variables to default value  
END
```

### 2. Time(const int hour, const int minute)

- Time class constructor with parameters

```
Time(hour, minute)  
    IF (hour>=0 AND hour<=24 AND minute>=0 AND minute <=59)  
        Set member variable hour to passed in hour  
        Set member variable minute to passed in minute  
    ELSE  
        Call clear() function to set member variables to default value  
        Print error message to the screen  
    END
```

### 3. Clear()

- This function is used to set all member variables to default value

```
Clear()  
    Set member variable hour to 0  
    Set member variable minute to 0  
END
```

### 4. GetHour() const

- This is the getter function and used to return hour of Time class object

```
GetHour()  
    Return member variable hour  
END
```

#### 5. SetHour(int hour)

- This is the setter function and used to set new value for member variable hour

```
SetHour(hour)
  IF (hour >= 0 AND hour <=23)
    Set member variable hour to hour
  ELSE
    Print error message to the screen
  ENDIF
END
```

#### 6. GetMinute() const

- This is the getter function and used to return minute of Time class object

```
GetHour()
  Return member variable minute
END
```

#### 7. SetMinute(int minute)

- This is the setter function and used to set new value for member variable minute

```
SetMinute(minute)
  IF (minute >= 0 AND minute <=59)
    Set member variable hour to hour
  ELSE
    Print error message to the screen
  ENDIF
END
```

## WindLogInfo.cpp

### 1. WindLogInfo()

- WindLogInfo constructor

WindLogInfo()

Call clear() function to set member variables to default value  
END

### 2. WindLogInfo()

- WindLogInfo constructor with parameters

WindLogInfo (speed, solar, air)

Set member variable speed to passed in speed  
Set member variable solar to passed in solar  
Set member variable air to passed in air  
END

### 3. Clear()

- This function is used to set all member variables to default value

Clear()

Set member variable speed to 0  
Set member variable solar to 0  
Set member variable air to 0  
END

### 4. GetSpeed()

- This function is the getter function and used to return the member variable speed

GetSpeed()

Return member variable speed  
END

#### 5. SetSpeed(float newSpeed)

- This function is the setter function and used to set new value for the member variable wind speed

```
SetSpeed(speed)
    Set member variable speed to passed in speed
END
```

#### 6. GetSolar()

- This function is the getter function and used to return the member variable solar

```
GetSolar()
    Return member variable solar
END
```

#### 7. SetSolar(float newSolar)

- This function is the setter function and used to set new value for the member variable solar radiation

```
SetSolar(solar)
    Set member variable solar to passed in solar
END
```

#### 8. GetAir()

- This function is the getter function and used to return the member variable air

```
GetAir()
    Return member variable air
END
```

#### 9. SetAir(float newAir)

- This function is the setter function and used to set new value for the member variable ambient air temperature

```
SetAir(air)
    Set member variable air to passed in air
END
```

## CollectData.cpp

### 1. ~CollectData()

- This is the destructor or CollectData class

```
~CollectData()  
    Set m_total to 0  
    Set m_max to 0  
END
```

### 2. Sum (Value & data)

- Calculate the running total

```
Sum (Value & data)  
    Add the value of pass in Value object  
END
```

### 3. Max (Value & data)

- Find the largest value

```
Max (Value & data)  
    IF(data's value is greater than the member variable m_max)  
        Set m_max to the largest value  
    ENDIF  
END
```

### 4. GetTotal() const

- Getter function

```
GetTotal()  
    Return m_total  
END
```

### 5. GetMax () const

- Getter function

```
GetMax ()  
    Return m_max  
END
```

## Value.cpp

### 1. ~Value ()

- This is the destructor or Value class

```
~CollectData()  
    Set m_value to 0  
END
```

### 2. GetValue ()

- Getter method

```
GetValue ()  
    Set m_value to 0  
END
```

### 3. SetValue (float number)

- Setter method

```
SetValue (float number)  
    Set m_value to number  
END
```



## bst.h

### 1. BinaryTreePointer()

- Constructor of bst class

```
BinaryTreePointer()  
    Set root to null pointer  
END
```

### 2. BinaryTreePointer (const BinaryTreePointer <T>& otherTree)

- Copy constructor

```
BinaryTreePointer(const BinaryTreePointer<T>& otherTree)  
    IF(otherTree root is equal to null pointer)  
        Set member variable root to null pointer  
    ELSE  
        Copy otherTree to this tree  
    ENDIF  
END
```

### 3. BinaryTreePointer <T>& operator=(const BinaryTreePointer & otherTree)

- Overload assignment operator

```
BinaryTreePointer<T>& operator=(const BinaryTreePointer& otherTree)  
    IF(both tree are different)  
        Destroy this tree  
        Create a new Node and assign to otherTree node  
        Copy otherTree to this tree  
    ENDIF  
    Return this node  
END
```

4. void CopyTree(Node<T>\* &copiedTreeRoot, Node<T>\* otherTreeRoot)

- Deep copy method

```
void CopyTree(Node<T>* &copiedTreeRoot, Node<T>* otherTreeRoot)
    IF(otherTree is no null pointer)
        Set copiedTree to null pointer
    ELSE
        Assign copiedTree to new Node
        Copy otherTree info to copiedTree
        Copy left side of node
        Copy right side of node
    ENDIF
END
```

5. ~BinaryTreePointer()

- Destructor of bst

```
~BinaryTreePointer()
    Delete node
END
```

6. void InorderTraversal(funcP fp) const

- Public version inorder traversal method

```
void InorderTraversal(funcP fp) const
    Call private version Inorder traversal method
END
```

7. void PreorderTraversal(funcP fp) const

- Public version preorder traversal method

```
void PreorderTraversal(funcP fp) const
    Call private version preorder traversal method
END
```

8. void PostorderTraversal(funcP fp) const

- Public version postorder traversal method

```
void PostorderTraversal(funcP fp) const
    Call private version postorder traversal method
END
```

9. void Inorder(funcP fp, Node<T> \*p) const

- Private version of inorder traversal method

```
void Inorder(funcP fp, Node<T> *p) const
    IF(p is not null pointer)
        Call this function recursively - left link
        Call the function pointer and pass in node info
        Call this function recursively - right link
    ENDIF
END
```

10. void Preorder (funcP fp, Node<T> \*p) const

- Private version of Preorder traversal method

```
void Preorder(funcP fp, Node<T> *p) const
    IF(p is not null pointer)
        Call the function pointer and pass in node info
        Call this function recursively - left link
        Call this function recursively - right link
    ENDIF
END
```

11. void Postorder (funcP fp, Node<T> \*p) const

- Private version of Postorder traversal method

```
void Postorder(funcP fp, Node<T> *p) const
    IF(p is not null pointer)
        Call this function recursively - left link
        Call this function recursively - right link
        Call the function pointer and pass in node info
    ENDIF
END
```

## 12. void Destroy(Node<T>\* &p)

- Delete the node pointer – private version

```
void Destroy(Node<T>* &p)
    IF(p is not null pointer)
        Call this function recursively - left link
        Call this function recursively - right link
        Delete pointer p
        Set p to null pointer
    ENDIF
END
```

## 13. void DestroyTree ()

- Delete the node pointer – public version

```
void DestroyTree ()
    Destroy the tree
END
```

## 14. bool SearchItem(const T& searchItem) const

- Search for specific item in the bst – public version

```
SearchItem(const T& searchItem) const
    IF( item is found)
        Return true
    ELSE
        Return false
    END
```

15. bool SearchElement(Node<T>\* nd, const T& item) const

- Search for specific item in the bst – private version

```
SearchItem(const T& searchItem) const
    IF( nd equal to null pointer)
        Return false
    ENDIF
    IF (found the item)
        Return true
    ENDIF
    IF(item is greater than the mode info)
        Search for right tree
    ELSE
        Search for left tree
    END
END
```

16. void InsertItem(const T& insertItem)

- Used to insert item to the bst – public version

```
Void InsertItem(const T& insertItem)
    Insert the pass in item
END
```

17. void InsertElement(Node<T>\*& nd, T item)

- Search for specific item in the bst – private version

```
void InsertElement(Node<T>*& nd, T item)
    IF( nd equal to null pointer)
        Create new Node
        Insert new item
        Set left link to null pointer
        Set right link to null pointer
    ELSE
        IF (item < node info)
            Insert item to left tree
        ELSE
            Insert item to right tree
        ENDIF
    ENDIF
END
```

## 4. Test Plan

Test	Description	Test data/Values	Expected Output	Passed
main.cpp				
1.	Once the user start executes the program, the program should keep execute and display the menu to allow the user to perform different task. The program should terminate only when the user enters 6 to stop execute.	Input 1: 9 Input 2: 7 Input 3: 6  (Example here I just enter any option instead of 6, the program should keep displaying the menu until last input-6 then terminate and display goodbye message).	(Keep displaying menu until 6 is entered)  Thank you bye ~	Pass
2.	The program provides 6 different options for user to perform different tasks. However, if the user enters any option apart from 1-6, the program should display an error message to remind the user.	Input 1: 9  (Enter 9 because 9 is an invalid option)	Invalid option !	Pass
3.	Once the program is executed, the program will automatically read data from the multiple data files from the data folder. The user does not need to care about reading the data part, because it is done by the program itself.	No input required.	All data loaded. Then the program will display the number of file read, number of file fail to read and the menu. Can execute option 1-5 to check for all data.	Pass

Test	Description	Change Code	Expected Output	Passed
4.	The option menu provides 6 options for the user to perform different task. The user needs to enter the number of tasks to perform the particular task. If the user enters a non-numeric value, the program will display the error message to remind the user.	Input 1: b  (Example here I used “b” because “b” is not numerical value)	Please enter a numerical value !	Pass
5.	Option 1 allow user to display the average wind speed and average ambient air temperature for specific year and month. The program should prompt user to enters the year and month he/she wish to find the result. The value should be one decimal point.	Input 1: 1 Input 2: 2014 Input 3: 5  (Example here the user tries to get the result of May 2014)	May 2014: 17.1 km/h, 16.5 degrees C	Pass
6.	Option 1 allow user to display the average wind speed and average ambient air temperature for specific year and month. However, if the user enters the date which does not contains any data, the program should display “No data message to inform the user there is not data for the specific date.	Input 1: 1 Input 2: 2000 Input 3: 1  (Example here the user tries to get the result of January 2000 which does not contains any data)	January 2000: No Data	Pass



Test	Description	Change Code	Expected Output	Passed
7.	Option 2 allowed user to find the average wind speed and average ambient air temperature for each month of specific year. The program should prompt user to enter the year to find the result. The program should then display the result. If there is no data for a specific month, the program should display "No data" for the month. Example here, the file to be read is " MetData_Mar01-2014-Mar01-2015-ALL.csv"	Input 1: 2 Input 2: 2014  (Example here, the user tries to get the result of year 2014)	2014 January : 9.9 km/h, 17.0 degrees C February : No Data March : 20.3 km/h, 22.8 degrees C April : 13.7 km/h, 19.3 degrees C May : 17.1 km/h, 16.5 degrees C June : 4.7 km/h, 13.2 degrees C July : 12.7 km/h, 13.5 degrees C August : 19.0 km/h, 15.5 degrees C September : 20.6 km/h, 16.1 degrees C October : 18.9 km/h, 17.8 degrees C November : 20.3 km/h, 19.0 degrees C December : 21.7 km/h, 21.2 degrees C	Pass
8.	Option 2 allowed user to find the average wind speed and average ambient air temperature for each month of specific year. However, if the user enters a year which do not contains any data ,the program should display "No data" for each month instead of crashing the program.	Input 1: 2 Input 2: 2000  (Example here, user tries to find the result for 2000. However, there is no record for 2000)	2020 January : No Data February : No Data March : No Data April : No Data May : No Data June : No Data July : No Data August : No Data September : No Data October : No Data November : No Data December : No Data	Pass

Test	Description	Change Code	Expected Output	Passed
9.	Option 3 allowed user to find the total solar radiation for each month of specific year. The program should prompt user to enter the year to find the result. The program should then display the result. If there is no data for a specific month, the program should display "No data" for the month. Example here, the file to be read is " MetData_Mar01-2014-Mar01-2015-ALL.csv"	Input 1: 3 Input 2: 2015  (Example here the user tries to get the result of 2015)	2012 January: 219.4 kWh/m <sup>2</sup> February: 201.7 kWh/m <sup>2</sup> March: 209.2 kWh/m <sup>2</sup> April: 128.0 kWh/m <sup>2</sup> May: 103.5 kWh/m <sup>2</sup> June: 62.3 kWh/m <sup>2</sup> July: 100.7 kWh/m <sup>2</sup> August: 119.4 kWh/m <sup>2</sup> September: 155.6 kWh/m <sup>2</sup> October: 201.5 kWh/m <sup>2</sup> November: 223.2 kWh/m <sup>2</sup> December: 244.7 kWh/m <sup>2</sup>	Pass
10.	Option 3 allowed user to find the total solar radiation for each month of specific year. However, if the user enters a year which do not contains any data ,the program should display "No data" for each month instead of crashing the program.	Input 1: 3 Input 2: 2018	2018 January : No Data February : No Data March : No Data April : No Data May : No Data June : No Data July : No Data August : No Data September : No Data October : No Data November : No Data December : No Data	Pass

Test	Description	Change Code	Expected Output	Passed
11.	Option 4 allowed user to output the average wind speed (km/h), average ambient air temperature and total solar radiation in kWh/m2 for each month of a specified year to a csv file call "WindTempSolar.csv". If there is no data for a specific month, the program will not output "No data". Instead, the program will skip the month which has no data.	Input 1: 4 Input 2: 2014  (Example her the user writes the data of year 2014 to the csv file)	All data has been successfully output to the csv file  +  "WindTempSolar.csv" contains all the record.	Pass
12.	Option 4 allowed user to output the average wind speed (km/h), average ambient air temperature and total solar radiation in kWh/m2 for each month of a specified year to a csv file call "WindTempSolar.csv". If there is no data for the whole year, the program will only output 1 line of "No data" with the corresponding year into the output file	Input 1: 4 Input 2: 2021  (Example her the user writes the data of year 2021 to the csv file which does not contain any data)	All data has been successfully output to the csv file  +  "WindTempSolar.csv" output "No data"	Pass

Test	Description	Change Code	Expected Output	Passed
13.	Check if the user enters an invalid month during option 1 progress. The month can only be between 1 to 12. If the user enters an invalid month, the program should display invalid message to the user.	Input 1: 1 Input 2: 2014 Input 3: 99  (Example here, the user enters 99 for month, which is an invalid input)	Invalid month  Invalid date 2014: No Data	Pass
14.	The program will read data from multiple data files from the data folder/ However, the program will ignore those duplicate data file instead of reading the same data file multiple times. Example here I have 11 data files in the met_index.txt. However, there are 2 duplicate files. The program should only read 9 data files instead of 11 (ignore the 2 duplicate files)	No input required	Total file read: 9 Reading file error: 0	Pass

Test	Description	Change Code	Expected Output	Passed
15.	Menu option 5 will prompt user to enter a date in d/m/yyyy format and calculate, find and display the time for the highest solar radiation for the specific date.	Input 1: 5 Input 2: 1/1/2014	Date: 1/1/2016 High solar radiation for the day: 1172 W/m2  Time: 12:0	Pass
16.	Menu option 5 will prompt user to enter a date in d/m/yyyy format and calculate, find and display the time for the highest solar radiation for the specific date. There might have same data in the data file and the program should ignore those same data. Example here, the data file contains 2 x records of 12.30 and 2 x records of 12.50. The program should ignore those duplicates record.	Input 1: 5 Input 2: 2/2/2014	Date: 2/2/2016 High solar radiation for the day: 1080 W/m2  Time: 12:30 12:50	Pass

Test	Description	Change Code	Expected Output	Passed
17.	Menu option 5 will prompt user to enter a date in d/m/yyyy format and calculate, find and display the time for the highest solar radiation for the specific date. If there are not record for the specific date, the program will display no record. Example here, there is no data for year 2000.	Input 1: 5 Input 2: 1/1/2000	Date: 1/1/2000 High solar radiation for the day: 0 W/m2  Time: No record	Pass
18.	<p>This test is to manually calculate the value from excel file and compare with the value calculated by the program. This test will check the value of average wind speed for January 2007.</p> <p>The column for January 2007 in excels file is K2 to k4406. The L column is manually inserted. It is used to calculate the wind speed value * 3.6. Then I will use the format =SUM(L2:L4406) to calculate the total of all wind speed in January 2007. Lastly, use the total wind speed / 4406 to find the value.</p>	Input 1: 1 Input 2: 2007 Input 3: 1	Program value: 22.4 km/h Manually calculate: 22.43297	Pass

Test	Description	Change Code	Expected Output	Passed
19.	<p>This test is to manually calculate the value from excel file and compare with the value calculated by the program. This test will check the value of average ambient air temperature for January 2008.</p> <p>The column for January 2008 in excels file is R2 to R3603. I will use the format =SUM(R2:R3603) to calculate the total of value in January 2008. Lastly, use the total / 3601 to find the value.</p>	Input 1: 1 Input 2: 2008 Input 3: 1	Program value: 24.9 km/h Manually calculate: 24.91803	Pass

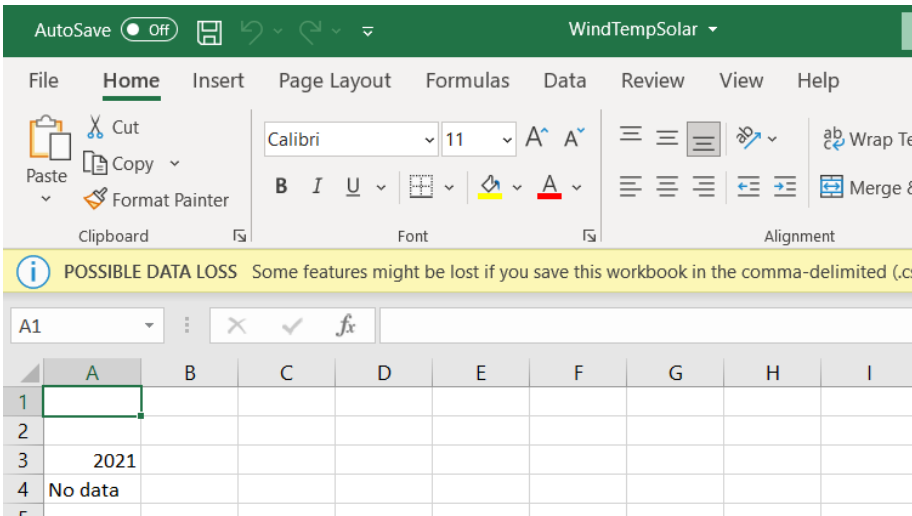
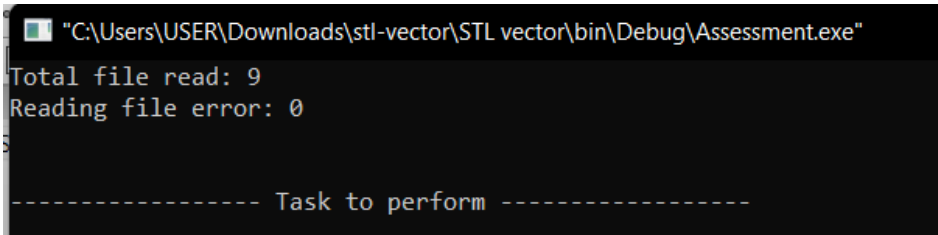
## 5. Actual Output of test run(s) – (Based on the sequence number in test plan)

Test no.	Actual output
1.	<pre> "C:\Users\USER\Downloads\stl-vector\STL vector\bin\Debug\AssessmentExe" Total file read: 9 Reading file error: 0  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: 9  Invalid option !  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: 7  Invalid option !  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: 6  Thank you bye ~  Process returned 0 (0x0)   execution time : 3.565 s Press any key to continue. </pre>
2.	<pre> Enter the option you want to perform: 9  Invalid option !  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: </pre>
3.	<pre> Total file read: 9 Reading file error: 0  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: </pre>



Test no.	Actual output
4.	<pre> Enter the option you want to perform: b  Please enter a numerical value !  ----- Task to perform -----  [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]: Output avg wind speed, avg ambient air temperature and total solar radiation for each month of year to a file [5]: The time for the highest solar radiation for a specific date [6]: Exit the program  Enter the option you want to perform: _ </pre>
5.	<pre> Enter the option you want to perform: 1 Enter the year to search: 2014 Enter the month to search: 5  May 2014: 17.1 km/h, 16.5 degrees C  ----- Task to perform ----- </pre>
6.	<pre> Enter the option you want to perform: 1 Enter the year to search: 2000 Enter the month to search: 1  January 2000: No Data </pre>
7.	<pre> Enter the option you want to perform: 2 Enter the year to search: 2014  2014 January: 9.9 km/h, 17.0 degrees C February: No Data March: 20.3 km/h, 22.8 degrees C April: 13.7 km/h, 19.3 degrees C May: 17.1 km/h, 16.5 degrees C June: 4.7 km/h, 13.2 degrees C July: 12.7 km/h, 13.5 degrees C August: 19.0 km/h, 15.5 degrees C September: 20.6 km/h, 16.1 degrees C October: 18.9 km/h, 17.8 degrees C November: 20.3 km/h, 19.0 degrees C December: 21.7 km/h, 21.2 degrees C </pre>

Test no.	Actual output
8.	<pre> Enter the option you want to perform: 2 Enter the year to search: 2000  2000 January: No Data February: No Data March: No Data April: No Data May: No Data June: No Data July: No Data August: No Data September: No Data October: No Data November: No Data December: No Data </pre>
9.	<pre> Enter the option you want to perform: 3 Enter the year to search: 2012  2012 January: 219.3 kWh/m^2 February: 201.7 kWh/m^2 March: 209.1 kWh/m^2 April: 128.0 kWh/m^2 May: 103.5 kWh/m^2 June: 62.3 kWh/m^2 July: 100.7 kWh/m^2 August: 119.4 kWh/m^2 September: 155.5 kWh/m^2 October: 201.4 kWh/m^2 November: 223.1 kWh/m^2 December: 244.6 kWh/m^2 </pre>
10.	<pre> Enter the option you want to perform: 3 Enter the year to search: 2018  2018 January : No Data February : No Data March : No Data April : No Data May : No Data June : No Data July : No Data August : No Data September : No Data October : No Data November : No Data December : No Data </pre>

Test no.	Actual output																																																																	
11.	<table><tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th></tr><tr><td>1</td><td>2014</td><td></td><td></td><td></td></tr><tr><td>2</td><td>January</td><td>9.9</td><td>17</td><td>1.3</td></tr><tr><td>3</td><td>March</td><td>20.3</td><td>22.8</td><td>183.5</td></tr><tr><td>4</td><td>April</td><td>13.7</td><td>19.3</td><td>137.3</td></tr><tr><td>5</td><td>May</td><td>17.1</td><td>16.5</td><td>86.3</td></tr><tr><td>6</td><td>June</td><td>4.7</td><td>13.2</td><td>79.4</td></tr><tr><td>7</td><td>July</td><td>12.7</td><td>13.5</td><td>84</td></tr><tr><td>8</td><td>August</td><td>19</td><td>15.5</td><td>112.3</td></tr><tr><td>9</td><td>September</td><td>20.6</td><td>16.1</td><td>144.9</td></tr><tr><td>10</td><td>October</td><td>18.9</td><td>17.8</td><td>200.5</td></tr><tr><td>11</td><td>November</td><td>20.3</td><td>19</td><td>220</td></tr><tr><td>12</td><td>December</td><td>21.7</td><td>21.2</td><td>268.5</td></tr></table>		A	B	C	D	1	2014				2	January	9.9	17	1.3	3	March	20.3	22.8	183.5	4	April	13.7	19.3	137.3	5	May	17.1	16.5	86.3	6	June	4.7	13.2	79.4	7	July	12.7	13.5	84	8	August	19	15.5	112.3	9	September	20.6	16.1	144.9	10	October	18.9	17.8	200.5	11	November	20.3	19	220	12	December	21.7	21.2	268.5
	A	B	C	D																																																														
1	2014																																																																	
2	January	9.9	17	1.3																																																														
3	March	20.3	22.8	183.5																																																														
4	April	13.7	19.3	137.3																																																														
5	May	17.1	16.5	86.3																																																														
6	June	4.7	13.2	79.4																																																														
7	July	12.7	13.5	84																																																														
8	August	19	15.5	112.3																																																														
9	September	20.6	16.1	144.9																																																														
10	October	18.9	17.8	200.5																																																														
11	November	20.3	19	220																																																														
12	December	21.7	21.2	268.5																																																														
12.																																																																		
13.	<pre>Enter the option you want to perform: 1 Enter the year to search: 2014 Enter the month to search: 99 Invalid month Invalid date 2014: No Data</pre>																																																																	
14.																																																																		



Test no.	Actual output																																																																																																																																																																																																																																																																																																																																																																										
16.	<div>Date: 2/2/2016 High solar radiation for the day: 1080 W/m2  Time: 12:30 12:50</div> <div><ul style="list-style-type: none"><li>Manually check</li></ul><table><tr><td>4619</td><td>02/02/2016 10:30</td><td>3</td><td>112</td><td>46</td><td>1913.64</td><td>1015.3</td><td>1018.8</td><td>1018.9</td><td>21.7</td><td>28</td><td>6</td><td>972</td><td>22.9</td></tr><tr><td>4620</td><td>02/02/2016 10:40</td><td>1.2</td><td>165</td><td>29</td><td>1914.21</td><td>1015.2</td><td>1018.7</td><td>1018.8</td><td>21.7</td><td>25.1</td><td>6</td><td>1003</td><td>23</td></tr><tr><td>4621</td><td>02/02/2016 10:50</td><td>1.5</td><td>146</td><td>28</td><td>1914.74</td><td>1015</td><td>1018.5</td><td>1018.6</td><td>21.7</td><td>24.5</td><td>5</td><td>999</td><td>23.1</td></tr><tr><td>4622</td><td>02/02/2016 11:00</td><td>2.1</td><td>168</td><td>24</td><td>1915.33</td><td>1014.9</td><td>1018.3</td><td>1018.5</td><td>21.7</td><td>24.6</td><td>6</td><td>1026</td><td>23.2</td></tr><tr><td>4623</td><td>02/02/2016 11:10</td><td>2.5</td><td>166</td><td>106</td><td>1915.81</td><td>1014.8</td><td>1018.2</td><td>1018.4</td><td>21.7</td><td>25.5</td><td>4</td><td>1037</td><td>23.4</td></tr><tr><td>4624</td><td>02/02/2016 11:20</td><td>1.1</td><td>161</td><td>29</td><td>1916.4</td><td>1014.7</td><td>1018.2</td><td>1018.3</td><td>21.7</td><td>23.6</td><td>6</td><td>1033</td><td>23.5</td></tr><tr><td>4625</td><td>02/02/2016 11:30</td><td>1</td><td>175</td><td>29</td><td>1917.01</td><td>1014.6</td><td>1018</td><td>1018.2</td><td>21.7</td><td>22.1</td><td>6</td><td>1045</td><td>23.7</td></tr><tr><td>4626</td><td>02/02/2016 11:40</td><td>0.9</td><td>143</td><td>40</td><td>1917.56</td><td>1014.5</td><td>1017.9</td><td>1018.1</td><td>21.7</td><td>22.4</td><td>5</td><td>1049</td><td>23.8</td></tr><tr><td>4627</td><td>02/02/2016 11:50</td><td>0.1</td><td>169</td><td>36</td><td>1918.19</td><td>1014.3</td><td>1017.7</td><td>1017.9</td><td>21.7</td><td>20.7</td><td>6</td><td>1078</td><td>24</td></tr><tr><td>4628</td><td>02/02/2016 12:00</td><td>-0.2</td><td>152</td><td>44</td><td>1918.81</td><td>1014.2</td><td>1017.6</td><td>1017.8</td><td>21.7</td><td>20.6</td><td>6</td><td>1052</td><td>24.2</td></tr><tr><td>4629</td><td>02/02/2016 12:10</td><td>-0.4</td><td>159</td><td>35</td><td>1919.46</td><td>1014</td><td>1017.4</td><td>1017.6</td><td>21.7</td><td>18.9</td><td>6</td><td>1075</td><td>24.4</td></tr><tr><td>4630</td><td>02/02/2016 12:20</td><td>-0.8</td><td>184</td><td>38</td><td>1920.11</td><td>1013.9</td><td>1017.3</td><td>1017.5</td><td>21.7</td><td>18.3</td><td>6</td><td>1079</td><td>24.6</td></tr><tr><td>4631</td><td>02/02/2016 12:30</td><td>0.4</td><td>184</td><td>38</td><td>1920.69</td><td>1013.7</td><td>1017.1</td><td>1017.3</td><td>21.7</td><td>19.9</td><td>5</td><td>1080</td><td>24.7</td></tr><tr><td>4632</td><td>02/02/2016 12:40</td><td>0.4</td><td>179</td><td>44</td><td>1921.28</td><td>1013.6</td><td>1017</td><td>1017.2</td><td>21.7</td><td>19.3</td><td>5</td><td>1069</td><td>24.9</td></tr><tr><td>4633</td><td>02/02/2016 12:50</td><td>0.9</td><td>163</td><td>41</td><td>1921.86</td><td>1013.5</td><td>1016.9</td><td>1017.1</td><td>21.7</td><td>20.2</td><td>5</td><td>1080</td><td>25.2</td></tr><tr><td>4634</td><td>02/02/2016 13:00</td><td>4.4</td><td>213</td><td>47</td><td>1922.46</td><td>1013.4</td><td>1016.8</td><td>1017</td><td>21.7</td><td>26.3</td><td>6</td><td>1057</td><td>25.4</td></tr><tr><td>4635</td><td>02/02/2016 13:10</td><td>5.4</td><td>247</td><td>48</td><td>1923.09</td><td>1013.3</td><td>1016.7</td><td>1016.9</td><td>21.7</td><td>29.4</td><td>7</td><td>1049</td><td>25.6</td></tr><tr><td>4636</td><td>02/02/2016 13:20</td><td>6.8</td><td>219</td><td>35</td><td>1923.68</td><td>1013.3</td><td>1016.7</td><td>1016.9</td><td>21.7</td><td>33.5</td><td>7</td><td>1043</td><td>25.8</td></tr><tr><td>4637</td><td>02/02/2016 13:30</td><td>8.1</td><td>233</td><td>34</td><td>1924.31</td><td>1013.2</td><td>1016.6</td><td>1016.8</td><td>21.7</td><td>35.5</td><td>8</td><td>1031</td><td>26</td></tr><tr><td>4638</td><td>02/02/2016 13:40</td><td>7.6</td><td>235</td><td>32</td><td>1924.93</td><td>1013.2</td><td>1016.6</td><td>1016.8</td><td>21.7</td><td>36.2</td><td>8</td><td>1019</td><td>26.3</td></tr><tr><td>4639</td><td>02/02/2016 13:50</td><td>8</td><td>234</td><td>36</td><td>1925.54</td><td>1013</td><td>1016.4</td><td>1016.6</td><td>21.7</td><td>37.3</td><td>8</td><td>1005</td><td>26.5</td></tr><tr><td>4640</td><td>02/02/2016 14:00</td><td>8.8</td><td>180</td><td>67</td><td>1926.1</td><td>1012.9</td><td>1016.3</td><td>1016.5</td><td>21.7</td><td>37.6</td><td>7</td><td>1002</td><td>26.7</td></tr><tr><td>4641</td><td>02/02/2016 14:10</td><td>8.5</td><td>241</td><td>36</td><td>1926.83</td><td>1012.8</td><td>1016.2</td><td>1016.4</td><td>21.7</td><td>35.6</td><td>10</td><td>970</td><td>26.9</td></tr><tr><td>4642</td><td>02/02/2016 14:20</td><td>8.1</td><td>229</td><td>37</td><td>1927.42</td><td>1012.7</td><td>1016.1</td><td>1016.3</td><td>21.7</td><td>38.1</td><td>8</td><td>964</td><td>27.1</td></tr><tr><td>4643</td><td>02/02/2016 14:30</td><td>8.1</td><td>235</td><td>34</td><td>1928.01</td><td>1012.7</td><td>1016.1</td><td>1016.3</td><td>21.7</td><td>37.5</td><td>8</td><td>930</td><td>27.3</td></tr></table></div>													4619	02/02/2016 10:30	3	112	46	1913.64	1015.3	1018.8	1018.9	21.7	28	6	972	22.9	4620	02/02/2016 10:40	1.2	165	29	1914.21	1015.2	1018.7	1018.8	21.7	25.1	6	1003	23	4621	02/02/2016 10:50	1.5	146	28	1914.74	1015	1018.5	1018.6	21.7	24.5	5	999	23.1	4622	02/02/2016 11:00	2.1	168	24	1915.33	1014.9	1018.3	1018.5	21.7	24.6	6	1026	23.2	4623	02/02/2016 11:10	2.5	166	106	1915.81	1014.8	1018.2	1018.4	21.7	25.5	4	1037	23.4	4624	02/02/2016 11:20	1.1	161	29	1916.4	1014.7	1018.2	1018.3	21.7	23.6	6	1033	23.5	4625	02/02/2016 11:30	1	175	29	1917.01	1014.6	1018	1018.2	21.7	22.1	6	1045	23.7	4626	02/02/2016 11:40	0.9	143	40	1917.56	1014.5	1017.9	1018.1	21.7	22.4	5	1049	23.8	4627	02/02/2016 11:50	0.1	169	36	1918.19	1014.3	1017.7	1017.9	21.7	20.7	6	1078	24	4628	02/02/2016 12:00	-0.2	152	44	1918.81	1014.2	1017.6	1017.8	21.7	20.6	6	1052	24.2	4629	02/02/2016 12:10	-0.4	159	35	1919.46	1014	1017.4	1017.6	21.7	18.9	6	1075	24.4	4630	02/02/2016 12:20	-0.8	184	38	1920.11	1013.9	1017.3	1017.5	21.7	18.3	6	1079	24.6	4631	02/02/2016 12:30	0.4	184	38	1920.69	1013.7	1017.1	1017.3	21.7	19.9	5	1080	24.7	4632	02/02/2016 12:40	0.4	179	44	1921.28	1013.6	1017	1017.2	21.7	19.3	5	1069	24.9	4633	02/02/2016 12:50	0.9	163	41	1921.86	1013.5	1016.9	1017.1	21.7	20.2	5	1080	25.2	4634	02/02/2016 13:00	4.4	213	47	1922.46	1013.4	1016.8	1017	21.7	26.3	6	1057	25.4	4635	02/02/2016 13:10	5.4	247	48	1923.09	1013.3	1016.7	1016.9	21.7	29.4	7	1049	25.6	4636	02/02/2016 13:20	6.8	219	35	1923.68	1013.3	1016.7	1016.9	21.7	33.5	7	1043	25.8	4637	02/02/2016 13:30	8.1	233	34	1924.31	1013.2	1016.6	1016.8	21.7	35.5	8	1031	26	4638	02/02/2016 13:40	7.6	235	32	1924.93	1013.2	1016.6	1016.8	21.7	36.2	8	1019	26.3	4639	02/02/2016 13:50	8	234	36	1925.54	1013	1016.4	1016.6	21.7	37.3	8	1005	26.5	4640	02/02/2016 14:00	8.8	180	67	1926.1	1012.9	1016.3	1016.5	21.7	37.6	7	1002	26.7	4641	02/02/2016 14:10	8.5	241	36	1926.83	1012.8	1016.2	1016.4	21.7	35.6	10	970	26.9	4642	02/02/2016 14:20	8.1	229	37	1927.42	1012.7	1016.1	1016.3	21.7	38.1	8	964	27.1	4643	02/02/2016 14:30	8.1	235	34	1928.01	1012.7	1016.1	1016.3	21.7	37.5	8	930	27.3
4619	02/02/2016 10:30	3	112	46	1913.64	1015.3	1018.8	1018.9	21.7	28	6	972	22.9																																																																																																																																																																																																																																																																																																																																																														
4620	02/02/2016 10:40	1.2	165	29	1914.21	1015.2	1018.7	1018.8	21.7	25.1	6	1003	23																																																																																																																																																																																																																																																																																																																																																														
4621	02/02/2016 10:50	1.5	146	28	1914.74	1015	1018.5	1018.6	21.7	24.5	5	999	23.1																																																																																																																																																																																																																																																																																																																																																														
4622	02/02/2016 11:00	2.1	168	24	1915.33	1014.9	1018.3	1018.5	21.7	24.6	6	1026	23.2																																																																																																																																																																																																																																																																																																																																																														
4623	02/02/2016 11:10	2.5	166	106	1915.81	1014.8	1018.2	1018.4	21.7	25.5	4	1037	23.4																																																																																																																																																																																																																																																																																																																																																														
4624	02/02/2016 11:20	1.1	161	29	1916.4	1014.7	1018.2	1018.3	21.7	23.6	6	1033	23.5																																																																																																																																																																																																																																																																																																																																																														
4625	02/02/2016 11:30	1	175	29	1917.01	1014.6	1018	1018.2	21.7	22.1	6	1045	23.7																																																																																																																																																																																																																																																																																																																																																														
4626	02/02/2016 11:40	0.9	143	40	1917.56	1014.5	1017.9	1018.1	21.7	22.4	5	1049	23.8																																																																																																																																																																																																																																																																																																																																																														
4627	02/02/2016 11:50	0.1	169	36	1918.19	1014.3	1017.7	1017.9	21.7	20.7	6	1078	24																																																																																																																																																																																																																																																																																																																																																														
4628	02/02/2016 12:00	-0.2	152	44	1918.81	1014.2	1017.6	1017.8	21.7	20.6	6	1052	24.2																																																																																																																																																																																																																																																																																																																																																														
4629	02/02/2016 12:10	-0.4	159	35	1919.46	1014	1017.4	1017.6	21.7	18.9	6	1075	24.4																																																																																																																																																																																																																																																																																																																																																														
4630	02/02/2016 12:20	-0.8	184	38	1920.11	1013.9	1017.3	1017.5	21.7	18.3	6	1079	24.6																																																																																																																																																																																																																																																																																																																																																														
4631	02/02/2016 12:30	0.4	184	38	1920.69	1013.7	1017.1	1017.3	21.7	19.9	5	1080	24.7																																																																																																																																																																																																																																																																																																																																																														
4632	02/02/2016 12:40	0.4	179	44	1921.28	1013.6	1017	1017.2	21.7	19.3	5	1069	24.9																																																																																																																																																																																																																																																																																																																																																														
4633	02/02/2016 12:50	0.9	163	41	1921.86	1013.5	1016.9	1017.1	21.7	20.2	5	1080	25.2																																																																																																																																																																																																																																																																																																																																																														
4634	02/02/2016 13:00	4.4	213	47	1922.46	1013.4	1016.8	1017	21.7	26.3	6	1057	25.4																																																																																																																																																																																																																																																																																																																																																														
4635	02/02/2016 13:10	5.4	247	48	1923.09	1013.3	1016.7	1016.9	21.7	29.4	7	1049	25.6																																																																																																																																																																																																																																																																																																																																																														
4636	02/02/2016 13:20	6.8	219	35	1923.68	1013.3	1016.7	1016.9	21.7	33.5	7	1043	25.8																																																																																																																																																																																																																																																																																																																																																														
4637	02/02/2016 13:30	8.1	233	34	1924.31	1013.2	1016.6	1016.8	21.7	35.5	8	1031	26																																																																																																																																																																																																																																																																																																																																																														
4638	02/02/2016 13:40	7.6	235	32	1924.93	1013.2	1016.6	1016.8	21.7	36.2	8	1019	26.3																																																																																																																																																																																																																																																																																																																																																														
4639	02/02/2016 13:50	8	234	36	1925.54	1013	1016.4	1016.6	21.7	37.3	8	1005	26.5																																																																																																																																																																																																																																																																																																																																																														
4640	02/02/2016 14:00	8.8	180	67	1926.1	1012.9	1016.3	1016.5	21.7	37.6	7	1002	26.7																																																																																																																																																																																																																																																																																																																																																														
4641	02/02/2016 14:10	8.5	241	36	1926.83	1012.8	1016.2	1016.4	21.7	35.6	10	970	26.9																																																																																																																																																																																																																																																																																																																																																														
4642	02/02/2016 14:20	8.1	229	37	1927.42	1012.7	1016.1	1016.3	21.7	38.1	8	964	27.1																																																																																																																																																																																																																																																																																																																																																														
4643	02/02/2016 14:30	8.1	235	34	1928.01	1012.7	1016.1	1016.3	21.7	37.5	8	930	27.3																																																																																																																																																																																																																																																																																																																																																														
17.	<div>Date: 1/1/2000 High solar radiation for the day: 0 W/m2  Time: No record</div> <div>----- Task to perform -----</div>																																																																																																																																																																																																																																																																																																																																																																										

Test  
no.

Actual output

18.

AutoSave Metdata-Jan-Dec2007

File Home Insert Page Layout Formulas Data Review View Help

Clipboard Font Alignment Number Styles Cells Editing Analysis

POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (csv) format. To preserve these features, save it in an Excel file format.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	WAST	DP	Dts	EV	QFE	QFF	QNH	RF	RH	S	SR	ST1	ST2	ST3	ST4	Sx	T					
2	01/01/2007 9:00	15.5	200	26	44.27	1007	1010.4	1010.6	0	63.6	6	21.6	664	26.1	29.2	28.2	26	10	22.75	Start col	L2	
3	01/01/2007 9:10	15.7	203	30	44.56	1007	1010.4	1010.6	0	62.2	5	18	702	26.1	29.2	28.2	26	8	23.3	End col	L4406	
4	01/01/2007 9:20	15.9	226	28	44.9	1006.8	1010.2	1010.4	0	62.7	7	25.2	711	26.2	29.1	28.2	26	15	23.38			
5	01/01/2007 9:30	16.1	233	29	45.38	1006.8	1010.2	1010.4	0	62.9	8	38.8	754	26.3	29.1	28.2	26	14	23.56	Total	98794.8	
6	01/01/2007 9:40	16.1	234	26	45.69	1006.9	1010.3	1010.5	0	63.1	9	32.4	780	26.4	29	28.2	26	16	23.57			
7	01/01/2007 9:50	16.7	237	24	46.08	1006.9	1010.3	1010.5	0	63.1	8	28.8	809	26.6	29	28.2	26	16	24.16	Average	22.43297	
8	01/01/2007 10:00	16.6	219	29	46.44	1006.9	1010.3	1010.5	0	63.5	7	25.2	837	26.7	28.9	28.2	26	14	24			
9	01/01/2007 10:10	16.8	223	32	46.82	1006.9	1010.3	1010.5	0	62.4	7	25.2	868	26.9	28.9	28.2	26	13	24.42			
10	01/01/2007 10:20	17.2	240	28	47.25	1006.8	1010.2	1010.4	0	63.4	9	32.4	893	27.1	28.9	28.2	26	17	24.57			
11	01/01/2007 10:30	17.4	228	34	47.65	1006.7	1010.1	1010.3	0	64.5	8	38.8	918	27.3	28.8	28.2	26	15	24.55			
12	01/01/2007 10:40	16.8	235	32	48.04	1006.8	1010.2	1010.4	0	61.9	7	25.2	946	27.6	28.8	28.2	26	14	24.55			
13	01/01/2007 10:50	16.6	240	26	48.51	1006.8	1010.2	1010.4	0	60.4	9	32.4	963	27.8	28.8	28.2	26	16	24.77			
14	01/01/2007 11:00	16.1	239	34	48.96	1006.8	1010.2	1010.4	0	59	8	28.8	979	28.1	28.8	28.2	26	14	24.61			
15	01/01/2007 11:10	15.6	1	65	49.41	1006.8	1010.2	1010.4	0	59.2	8	28.8	1004	28.3	28.7	28.2	26	16	24.09			
16	01/01/2007 11:20	16.3	232	30	49.87	1006.6	1010	1010.2	0	58.3	8	28.8	1008	28.6	28.7	28.2	26	14	25.08			
17	01/01/2007 11:30	15.8	235	27	50.4	1006.6	1010	1010.2	0	57.8	10	36	1030	28.9	28.7	28.2	26	16	24.67			
18	01/01/2007 11:40	16.3	232	29	50.88	1006.6	1010.1	1010.2	0	57.1	8	38.8	1045	29.2	28.7	28.2	26	15	25.37			
19	01/01/2007 11:50	15.2	220	29	51.38	1006.6	1010.1	1010.2	0	54.4	8	28.8	1048	29.5	28.6	28.2	26	15	25.08			
20	01/01/2007 12:00	14.5	231	30	51.93	1006.6	1010	1010.2	0	52.4	9	32.4	1055	29.8	28.6	28.2	26	17	24.87			
21	01/01/2007 12:10	13.9	226	28	52.5	1006.7	1010.1	1010.3	0	50.4	9	32.4	1067	30.1	28.6	28.2	26	17	24.95			
22	01/01/2007 12:20	15.1	232	32	53.11	1006.7	1010.1	1010.3	0	51.1	10	36	1073	30.4	28.6	28.1	26	17	25.96			
23	01/01/2007 12:30	12.9	240	25	53.81	1006.7	1010.1	1010.3	0	45.8	11	39.6	1067	30.7	28.6	28.2	26	17	25.48			
24	01/01/2007 12:40	14	270	77	54.35	1006.6	1010	1010.2	0	50.2	8	28.8	1073	31	28.6	28.1	26	17	25.11			
25	01/01/2007 12:50	14.4	227	35	54.91	1006.6	1010	1010.2	0	51.5	9	32.4	1060	31.3	28.5	28.1	26	16	25.08			
26	01/01/2007 13:00	14.5	237	29	55.51	1006.6	1010	1010.2	0	51.5	10	36	1061	31.6	28.6	28.1	26	18	25.25			
27	01/01/2007 13:10	14.1	236	30	56.1	1006.6	1010	1010.2	0	51.6	10	36	1058	31.9	28.5	28.1	26	19	24.76			

19.

AutoSave Metdata-Jan-Dec2008

File Home Insert Page Layout Formulas Data Review View Help

Clipboard Font Alignment Number Styles Cells Editing Analysis

POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (csv) format. To preserve these features, save it in an Excel file format.

	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	Dts	EV	QFE	QFF	QNH	RF	RH	S	SR	ST1	ST2	ST3	ST4	Sx	T								
2	112	21	63.18	1008.4	1011.8	1012	0	37.8	6	1011	26.2	28	27.7	25.4	10	26.28			START	R2			
3	108	24	63.68	1008.4	1011.8	1012	0	37.8	7	487	26.5	27.9	27.7	25.4	13	26.94			END	R3603			
4	98	20	64.16	1008.3	1011.7	1011.9	0	37	6	608	26.7	27.9	27.7	25.4	11	26.45			TOTAL	89729.83			
5	132	22	64.69	1008.1	1011.5	1011.7	0	36.4	6	887	27	27.9	27.7	25.4	9	27.23			AVG	24.91803			
6	127	26	65.32	1008	1011.5	1011.6	0	35.5	7	1098	27.4	27.8	27.7	25.4	10	27.94							
7	121	23	65.88	1007.8	1011.2	1011.4	0	35.1	6	963	27.7	27.8	27.7	25.4	9	28.26							
8	133	25	66.41	1007.7	1011	1011.3	0	32.5	7	338	28	27.8	27.7	25.4	10	28.52							
9	127	27	67.01	1007.4	1010.8	1011	0	33.3	8	519	28.3	27.8	27.7	25.4	11	28.53							
10	117	20	67.79	1007.3	1010.6	1010.9	0	31.8	9	1132	28.6	27.7	27.7	25.4	15	29.06							
11	103	20	68.52	1007.1	1010.4	1010.7	0	31.2	9	805	29	27.7	27.7	25.4	14	29.38							
12	116	21	69.37	1006.9	1010.2	1010.5	0	29.6	10	1029	29.3	27.7	27.7	25.4	16	29.73							
13	119	24	70.18	1006.8	1010.1	1010.4	0	28.8	9	1102	29.6	27.7	27.7	25.4	12	29.94							
14	132	23	71.01	1006.5	1009.8	1010.1	0	27.3	9	1050	29.9	27.7	27.7	25.4	17	31.27							
15	127	25	71.78	1006.4	1009.7	1010	0	26.7	8	1036	30.2	27.7	27.7	25.4	14	31.03							
16	104	21	72.6	1006.2	1009.5	1009.8	0	25.9	9	896	30.5	27.7	27.7	25.4	17	31.21							
17	102	26	73.44	1006	1009.3	1009.6	0	25.7	9	942	30.9	27.6	27.7	25.4	19	32.13							
18	124	27	74.1	1005.9	1009.2	1009.5	0	25	7	610	31.2	27.6	27.7	25.4	15	32.09							
19	109	25	74.83	1005.7	1009	1009.3	0	24	8	631	31.5	27.6	27.7	25.4	17	31.62							
20	112	24	75.58	1005.6	1008.9	1009.2	0	24.6	8	757	31.9	27.7	27.7	25.4	12	32.12							
21	97	24	76.29	1005.6	1008.9	1009.2	0	23.6	7	810	32.2	27.7	27.6	25.4	11	32.48							
22	85	33	77.02	1005.7	1009	1009.3	0	22.9	8	572	32.5	27.7	27.6	25.4	13	32.4							
23	108	26	77.68	1005.6	1008.9	1009.2	0	23.6	7	531	32.9	27.7	27.6	25.4	11	32.33							
24	81	18	78.46	1005.5	1008.8	1009.1	0	23.3	9	499	33.1	27.7	27.6	25.4	13	32.45							
25	100	18	79.17	1005.7	1009	1009.3	0	23	8	476	33.4	27.7	27.6	25.4	12	32.12							
26	83	27	79.82	1005.7	1009	1009.3	0	22.8	7	429	33.7	27.7	27.6	25.4	8	32.53							
27	106	18	80.47	1005.7	1009	1009.3	0	22.8	7	462	33.9	27.8	27.6	25.4	10	31.9							

# ICT283Assignment 1

<This sheet is given to the student via LMS as written feedback. A copy is sent to the unit coordinator.>

**Student Name: Tee Yee Kang**

Components	Comments
UML diagram (High level and Low level)	
Written rationale for the design with Data Dictionary	
Non-programming language specific algorithm <i>(no marks for word processed code as the program code is easier to read and understand than code that is messed up using a word processor)</i>	
Program that builds and works (includes coding, coding style including readability, doxygen comments, C++ classes). <i>Marks not allocated if program does not build or doxygen output is not provided.</i>	
Non-STL Vector class implementation and usage. <i>Marks not allocated if STL data structures/algorithms used.</i>	
Evaluation, Test plan and testing. <i>Marks only if evaluation.txt is also provided.</i>	

**Other advice (if any):**

**Number of days late:**

**Late penalty (10 marks a day):**

**Final Grade:**