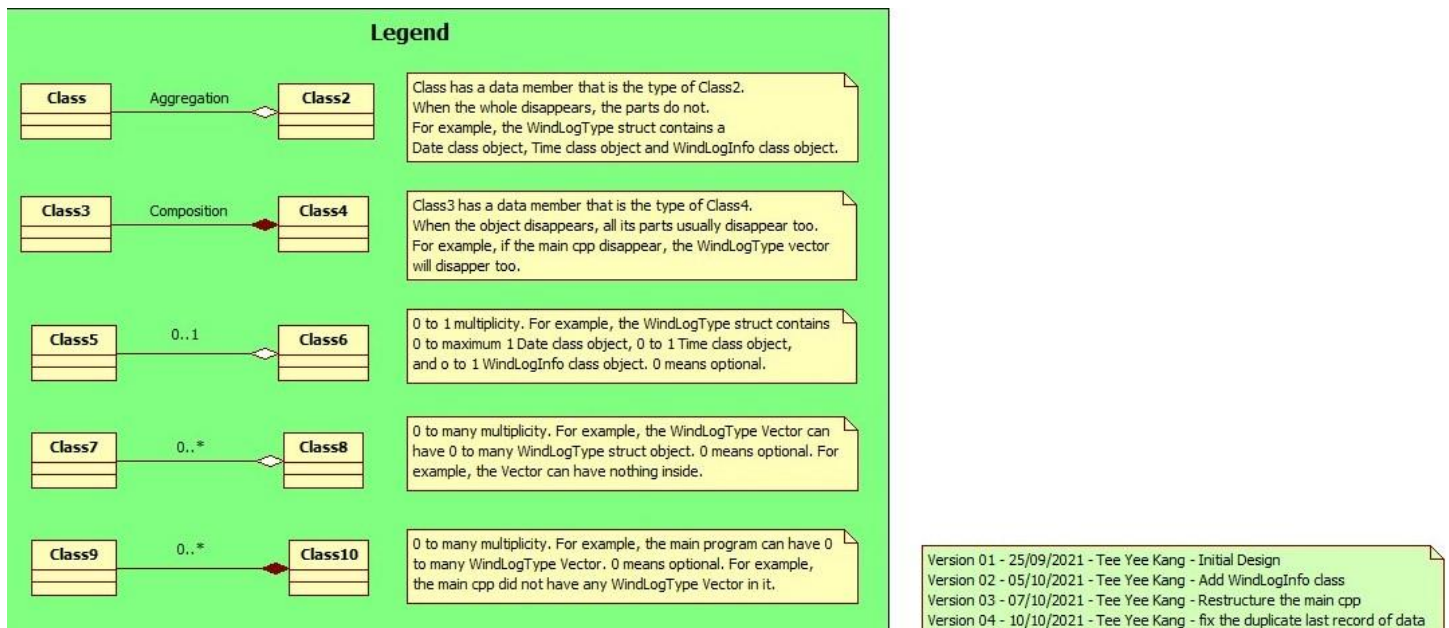


1. UML diagram (legend, low key and high key)

Legend



Date Class

Date
-m_day: int -m_month: int -m_year: int
+Date() +Date(day: const int, month: const int, year: const int) +Clear(): void +GetDate(): const void +SetDate(day: int, month: int, year: int): void +GetDay(): const int +SetDay(newDay: int): void +GetMonth(): const int +SetMonth(newMonth: int): void +GetYear(): const int +SetYear(newYear: int): void +GetMonthInString(): const string +ValidDate(day: int, month: int, year: int): bool +IsLeapYear(year: int): bool +operator <<(os: ostream &, D: const Date &): ostream & +operator >>(input: istream &, D: Date &): istream &

Time Class

Time
-m_hour: int -m_minute: int
+Time() +Time(hour: const int, minute: const int) +Clear(): void +GetHour(): const int +SetHour(hour: int): void +GetMinute(): const int +SetMinute(minute: int): void +operator <<(os: ostream &, T: const Time &): ostream & +operator >>(input: istream &, T: Time &): istream &

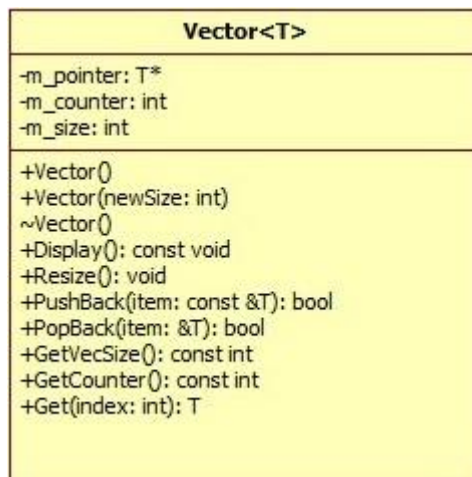
WindLogInfo Class

WindLogInfo
-m_speed: float -m_solar: float -m_air: float
+WindLogInfo() +WindLogInfo(speed: const float, solar: const float, air: const float) +Clear(): void +GetSpeed(): const float +SetSpeed(newSpeed: float): void +GetSolar(): const float +SetSolar(newSolar: float): void +GetAir(): const float +SetAir(newAir): void +operator <<(os: ostream &, W: const WindLogInfo &): ostream & +operator >>(input: istream &, W: WindLogInfo &): istream &

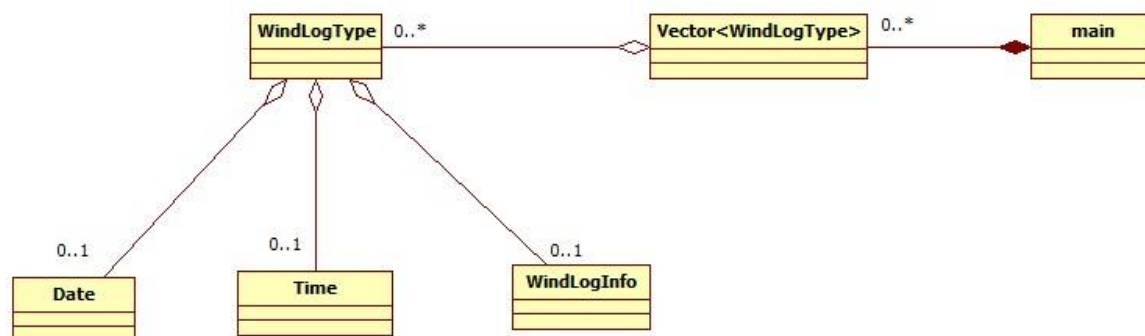
Struct – WindLogType

WindLogType
+d: Date +t: Time +w: WindLogInfo

Vector Class Template



High Level Diagram



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.

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

Vector Class Attributes

Name	Type	Protection	Description	Rationale
Vector				Simulate the STL Vector
m_pointer	Pointer	- (Private)	Pointer of type T which point to a dynamic array of type T. (type T is depends on user)	Template Vector class. T is the template data type. Based on the question requirement, this Vector class is behaved like array. Therefore, use the pointer to point at a dynamic array to store elements. Use private protection because did not want the user to direct access the member variables of the Time class object.
m_counter	integer	- (Private)	Represent the allocation of Vector	User can use this to know how many elements are currently in the array.
m_size	integer	- (Private)	The size of dynamic array store in the vector class	Use this member variables to represent the total size of the dynamic array.

Vector Class Methods

Name	Type	Protection	Description	Rationale
Vector()	procedure	+ (Public)	Constructor of the Vector class	This is essential for the class object. User can create Vector class object with default values (member variables).
Vector(int newSize)	procedure	+ (Public)	Constructor of the Vector class with parameter	This is essential for the class object. User can create Vector class object with his/her desire dynamic array's size.
~Vector()	virtual	+ (Public)	Destructor of the Vector class	Because the class contains pointer variables. That's why I implement the destructor for the class.
Display() const	void	+ (Public)	Display the elements store in the dynamic array of Vector class	I implement this function for the user to display all the elements in the dynamic array. User does not require to write the for loop, can just use this function to display everything in the array of Vector object.
Resize()	void	+ (Public)	Increase the size of dynamic array of Vector class	This function will be implemented inside the Pushback() function and the program will automatically increase the size of array once meet some requirement. I create this function separately, so that the user can resize the dynamic array manually as well (function reuse purpose)

Vector Class Methods Continue ...

Name	Type	Protection	Description	Rationale
PushBack(const T &item)	boolean	+ (Public)	Add new item into the dynamic array of Vector class	Represent the push function of STL vector class.
PopBack(T &item)	Boolean	+ (Public)	Remove existing item from the dynamic array of Vector class	Represent the pop function of STL vector class. The function will remove the last element in the array. The item to be removed will store in the pass in variable.
GetVecSize() const	integer	+ (Public)	Get the size of dynamic array of Vector class	Simple getter functions
GetCounter() const	integer	+ (Public)	Get the counter/allocation of Vector class	Simple getter functions
Get(int index)	T	+ (Public)	Get the element of dynamic array at specific position/index	Represent the at function of STL vector class.

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

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 Vector<WindLogType> &windlog);	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 Vector<WindLogType> &windlog, int year, int month);	float		A function used to calculate average wind speed store in Vector 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 Vector object.
GetAverageAir(const Vector<WindLogType> &windlog, int year, int month);	float		A function used to calculate average ambient air temperature store in Vector object 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 Vector object.

Name	Type	Protection	Description	Rationale
GetTotalSolar(const Vector<WindLogType> &windlog, int year, int month);	float		A function used to calculate total solar radiation store in Vector 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 Vector object.
option1(const Vector<WindLogType> &windlog);	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 Vector<WindLogType> &windlog);	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 Vector<WindLogType> &windlog);	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 Vector<WindLogType> &windlog);	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.
yearToFind();	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.
monthToFind();	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(Vector<WindLogType> &windlog);	void		A function used to read data from file and create corresponding class object and pass into the vector.	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 Vector object. Hence, this function can be used for another Vector 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.

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
END
```

2. controller(const Vector<WindLogType> &windlog)

- 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(Vector)
  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 (Vector<WindLogType> &windlog)

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

readDataFromFile (Vector)

Set variable theSpeed, theSolar, theAir to float type

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

Set endOfFile to boolean

Create WindLogType object – wObject

Open file "MetData_Mar01-2014-Mar01-2015-ALL.csv"

Read the first header line from file

WHILE (endOfFile is false)

 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 eind 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)

 Set endOfFile to true

 ELSE

 Convert wind speed to float type

 Convert solar radiation to float type

 Convert ambient air temperature to float type

 Set wind speed

 Set solar radiation

 Set ambient air temperature

 Store WindLogType object to pass in Vector

 ENDIF

 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 = 0
    Set speedAvg = 0
    Set counter = 0
    FOR (idx=0; idx<Vector counter; idx++)
        Retrieve Vector element at position idx
        IF (Vector element's year and month both = to pass in year and month)
            sumOfSpeed += Vector element's wind speed * 3.6
            counter increment by 1
        ENDIF
    Calculate average wind speed and store in speedAvg
    Return speedAvg
END
```

5. GetAverageAir(const Vector<WindLogType> &windlog, int year, int month)

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

```
GetAverageAir (Vector, year, month)
    Set sumOfAir = 0
    Set airAvg = 0
    Set counter = 0
    FOR (idx=0; idx<Vector counter; idx++)
        Retrieve Vector element at position idx
        IF (Vector element's year and month both = to pass in year and month)
            sumOfAir += Vector element's ambient air temperature
            counter increment by 1
        ENDIF
    Calculate average ambient air temperature and store in speedAvg
    Return airAvg
END
```

6. GetTotalSolar(const Vector<WindLogType> &windlog, int year, int month)

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

GetTotalSolar (Vector, year, month)

Set sumOfSolarRadiation = 0

Set hourlySolarRadiation = 0

Set finalSolarRadiation = 0

Set counter = 0

FOR (idx=0; idx<Vector counter; idx++)

Retrieve Vector element at position idx

IF (Value of Vector element's solar radiation > 100)

sumOfSolarRadiation += Vector element's solar radiation

counter increment by 1

ENDIF

sumOfSolarRadiation / 6 and store in hourlySolarRadiation

hourlySolarRadiation / 1000 and store in finalSolarRadiation

Return finalSolarRadiation

END

7. option1(const Vector<WindLogType> &windlog)

- The function is the controller of option 1.

Option1 (Vector)

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

Get the month to display in English form

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 Vector<WindLogType> &windlog)

- The function is the controller of option 2.

option2 (Vector)

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 Vector<WindLogType> &windlog)

- This function is the controller for option 3

option3 (Vector)

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 Vector<WindLogType> &windlog)

- This function is the controller for option 4

option4 (Vector)

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 (idx=0; idx<Vector's counter; increment idx by 1)

Retrieve Vector element at position idx

IF (Vector element's year == user's year)

Set hasData to true

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

ELSE

Write no data to output file

ENDIF

Close the output file

Print successful message to the screen

END

11. int yearToFind();

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

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

12. int monthToFind();

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

```
monthToFind()  
    Set month  
    Prompt user for month  
    Read month from user  
    Return month  
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
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
```

Vector.cpp

1. Vector()

- Vector class constructor (template)

```
Vector()  
    Set member variable size to 0  
    Create a pointer and point to a dynamic array  
    Set member variable counter to 0  
END
```

2. Vector(int newSize)

- Vector class constructor with parameters (template)

```
Vector(size)  
    Set member variable size to size  
    Create a pointer and point to a dynamic array with the passed in size  
    Set member variable counter to 0  
END
```

3. ~Vector()

- Vector class destructor

```
~Vector()  
    Delete pointer (clean up memory heap)  
END
```

4. Resize()

- This function is used to increase the size of dynamic array

```
Resize()  
    Create a new pointer call temp  
    Double the member variable size  
    Set pointer to point at a new dynamic array with double size  
    FOR (idx=0; idx < member variable counter; increment idx by 1)  
        Copy the elements in existing array to new create array  
    Delete pointer (clean up memory heap)  
    Set pointer to point at new create array  
END
```

5. PushBack(const T& item)

- This function is used to add new item into the array of Vector class

```
PushBack(item)
  IF(member variable counter >= member variable size / 2)
    Call the Resize() function to increase the array size
  ENDIF
  IF(member variable counter < member variable size)
    Add item into the array
    Increase member variable counter by 1
    Return true
  ELSE
    Return false
  ENDIF
END
```

6. PopBack(T &item)

- This function is used to remove item from the array of Vector class

```
PopBack(T &item)
  IF(member variable pointer == 0)
    Print no elements to the screen
    Return false
  ELSE
    Set item = last array element
    Decrease member variable counter by 1
    Return true
  ENDIF
END
```

7. GetVecSize ()

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

```
GetVecSize()
  Return member variable size
END
```

8. GetCounter ()

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

```
GetCounter()
  Return member variable counter
END
```

9. Get (int index)

- This function is used to return the element at the passed in position

```
Get(index)
  TRY
    IF(index > member variable size-1)
      Throw(index)
    ENDIF
  CATCH (index)
    Print index out of bound message
    IF (index >= 0 AND index < member variable counter)
      Return the element
    ENDIF
  END
```

10. Display ()

- This function is used to display all elements in the array of Vector class object

```
Display()
  FOR (idx = 0; idx < member variable counter; increment idx by 1)
    Print the element at position idx to the screen
  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
```

4. Test Plan

Test	Description	Change Code	Expected Output	Passed
Date Class				
1.	Check if the default constructor for Date class can successfully create with the default value. Get function also tested here when displaying.	Date date01; date01.GetDate();	0/0/0	Pass
2.	Check if the constructor with parameter can successfully create with the passed in value. Get function also tested here when displaying.	Date date02(3,1,1999); date02.GetDate();	3/1/1999	Pass
3.	Check if the set function is successfully change the value of member variable.	date01.SetDate(5,2,2021); date01.GetDate();	5/2/2021	Pass
4.	Check if the SetDay() function can successfully change the value of day.	date01.SetDay(11); date01.GetDate();	11/2/2021	Pass
5.	Check if the user has entered a leap year date.	Date date03(29,2,2024); date03.GetDate();	29/2/2024	Pass
6.	Check if the SetMonth() function can successfully change the value of month.	date01.SetMonth(12); date01.GetDate();	20/12/2021	Pass
7.	Check if the Setyear() function can successfully change the value of year.	date01.SetYear(1890); date01.GetDate();	20/12/1890	Pass
8.	Check if the << operator can successfully print out the output with the month in string form.	cout<<date01<<endl;	Date: 20 December 1890	Pass
9.	Check for invalid date value	date02.SetMonth(20);	Invalid date	Pass
10.	Check if the program can successfully read data from "dateFile.txt" can write those information into the "output.txt" file	-	(Check the "output.txt" file in the folder)	Pass

Test	Description	Change Code	Expected Output	Passed
Time Class				
11.	Check if the default constructor can successfully create with default value. The << operator also check here. It should print out the value of Time class object	Time t1; cout<<t1<<endl;	Time: 0:0	Pass
12.	Check if the constructor with parameters can successfully create with passed in value. The << operator also check here. It should print out the value of Time class object	Time t2(5, 25); cout<<t2<<endl;	Time: 5:25	Pass
13.	Check if the setHour() function can successfully set new value of hour.	t1.SetHour(12); cout<<t1<<endl;	Time: 12:0	Pass
14.	Check if the setMinute() function can successfully set new value of minute.	t1.SetMinute(35); cout<<t1<<endl;	Time: 12:35	Pass
15	Check if the GetHour() function can successfully return the hour of Time class object.	cout<<t1.GetHour()<<endl;	12	Pass
16.	Check if the GetMinute () function can successfully return the minute of Time class object.	cout<<t1.GetMinute()<<endl;	35	Pass
17.	Check if the program can successfully read data from "TimeFile.txt" can write those information into the "output.txt" file	-	(Check the "output.txt" file in the folder)	Pass

Test	Description	Change Code	Expected Output	Passed
Vector Class				
18.	Check if the Vector class can successfully create with int data type. To complete this test, I use for loop to enter numerical value into the Vector by using the Pushback() function. To see the result, I use Display() function to output all the numbers. Therefore, Display() and Pushback() functions are also tested here.	<pre> Vector<int> intvec; for(int idx=1; idx<20; idx++) { vec.PushBack(idx); } Intvec.Display() </pre>	1 2 3 4 5 6 7 ... until 19	Pass
19.	Check if the PopBack(T &item) function can successfully remove the element from the end of the array. To see the result, I use the Get() function with for loop to print out the elements. Inside the for loop also contain the GetCounter() function. Therefore, PopBack(T &item), Get() and GetCount() functions are tested here.	<pre> vec.PopBack(num1); vec.PopBack(num2); cout << "Test for the Get() function" << endl; for(int i=0; i<vec.GetCounter(); i++) { cout<<vec[i]<< " "; } </pre>	The deleted numbers are 19 and 18. After pop Test for the Get() function 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17	Pass

Test	Description	Change Code	Expected Output	Passed
20.	Check if the Get() function can successfully return the element at the specific position. Example here I try to return the element at position 2.	cout<<vec.Get(2)<<endl;	3	Pass
21.	Check is the GetSize() function can successfully return the current size of the dynamic array of Vector class object . Example here size is 40, because the program will automatically increase the size of the array if the counter is greater than size/2.	cout<<"\nTest for get size function: <<vec.GetVecSize()<<endl;	Test for get size function: 40	Pass
22.	Check if the Vector class can successfully create with Date class data type. To complete this test create 5 Date objects and use the Pushback() function to store the Date class objects into the vector. Then, use the Display() to see the result.	Date d1(3, 1, 1999); Date d2(3, 2, 1999); Date d3(3, 3, 1999); Date d4(3, 4, 1999); Date d5(3, 5, 1999); vec.PushBack(d1); vec.PushBack(d2); vec.PushBack(d3); vec.PushBack(d4); vec.PushBack(d5); vec.Display();	Date: 3 January 1999 Date: 3 February 1999 Date: 3 March 1999 Date: 3 April 1999 Date: 3 May 1999	Pass

Test	Description	Change Code	Expected Output	Passed
WindLogInfo Class				
23.	Check if the default constructor can successfully create with default value. The << operator also check here. It should print out the value of WindLogInfo class object	WindLogInfo wObject; cout<<wObject<<endl;	Speed: 0 Solar radiation: 0 Ambient air temperature: 0	Pass
24.	Check if the SetSpeed() function can successfully set the value of wind speed to the passed in value. To check the result, I used the GetSpeed() to return the value. Therefore, GetSpeed () function also tested here.	wObject.SetSpeed(5); cout<<wObject.GetSpeed() <<endl;	5	Pass
25.	Check if the SetSolar() function can successfully set the value of solar radiation to the passed in value. To check the result, I used the GetSolar() to return the value. Therefore, GetSolar() function also tested here.	wObject.SetSolar(10); cout<<wObject.GetSolar() <<endl;	10	Pass

Test	Description	Change Code	Expected Output	Passed
26.	Check if the SetAir() function can successfully set the value of ambient air temperature to the passed in value. To check the result, I used the GetAir() to return the value. Therefore, GetAir() function also tested here.	wObject.SetAir(15); cout<<wObject.GetAir()<<endl;	15	Pass
27.	Check if the program can successfully read data from "WindLogFile.txt" can write those information into the "output.txt" file	-	(Check the "output.txt" file in the folder)	Pass

Test	Description	Change Code	Expected Output	Passed
main.cpp				
28.	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 5 to stop execute.	Input 1: 9 Input 2: 7 Input 3: 5 (Example here I just enter any option instead of 5, the program should keep displaying the menu until last input-5 then terminate and display goodbye message).	(Keep displaying menu until 5 is entered) Thank you bye ~	Pass
29.	The program provides 5 different options for user to perform different tasks. However, if the user enters any option apart from 1-5, 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
30.	Once the program is executed, the program will automatically read data from the input file. The user does not need to care about reading the data part, because it is done by the program itself. Example here, the file to be read is " MetData_Mar01-2014-Mar01-2015-ALL.csv"	No input required. (To check the result, I purposely implement the Display() function.)	Date: 1 March 2014 Time: 9:0 Speed: 8 Solar radiation: 616 Ambient air temperature: 29.56 ...until the end	Pass

Test	Description	Change Code	Expected Output	Passed
31.	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: 2014 Input 2: 5 (Example here the user tries to get the result of May 2014)	May 2014: 17.1 km/h, 16.5 degrees C	Pass
32.	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: 2014 Input 2: 1 (Example here the user tries to get the result of January 2014 which does not contains any data)	January 2014: No Data	Pass

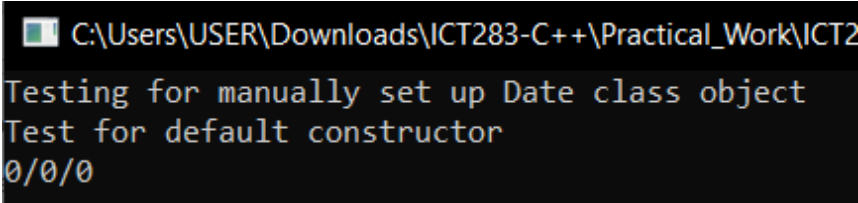
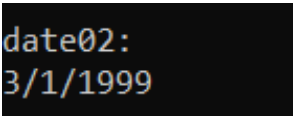
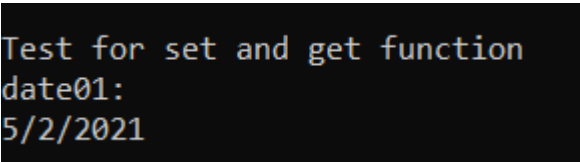

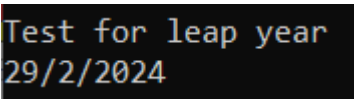
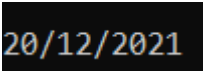
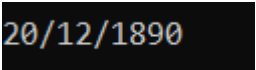
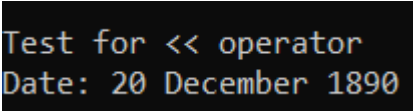
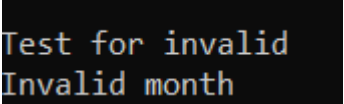
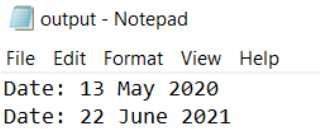
Test	Description	Change Code	Expected Output	Passed
33.	Option 2 allowed user to find the average wind speed and average ambient air temperature for each month of specific year. The program should prompt user to enter the year to find the result. The program should then display the result. If there is 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 : No Data 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
34.	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: 2020 (Example here, user tries to find the result for 2020. However, there is no record for 2020)	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

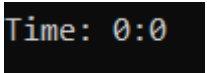
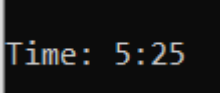
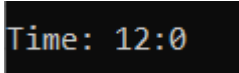
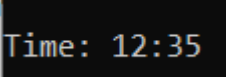
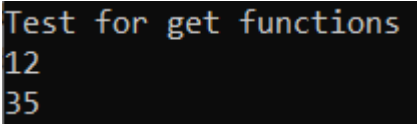
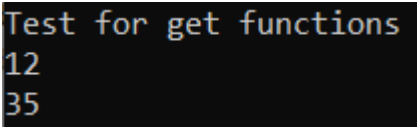
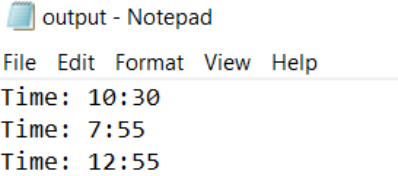
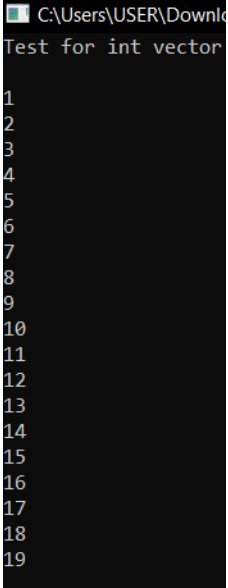
Test	Description	Change Code	Expected Output	Passed
35.	Option 3 allowed user to find the total solar radiation for each month of specific year. The program should prompt user to enter the year to find the result. The program should then display the result. If there is no data for a specific month, the 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)	2015 January : 254.7 kWh/m^2 February : 197.4 kWh/m^2 March : 0.9 kWh/m^2 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
36.	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
37.	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
38.	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
39.	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

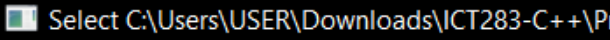
5. Output of test run(s) – (Based on the sequence number in test plan)

Test no.	Actual output
1.	
2.	
3.	
4.	
5.	
6.	
7.	
8.	
9.	
10.	


















Test no.	Actual output
11.	
12.	
13.	
14.	
15.	
16.	
17.	
18.	

Test no.	Actual output
19.	<pre> The deleted numbers are 19 and 18 After pop Test for the Get() operator 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 </pre>
20.	<pre> Test for get function 3 </pre>
21.	<pre> Test for get size function: 40 </pre>
22.	<pre> ----- Test for Date type vector Date: 3 January 1999 Date: 3 February 1999 Date: 3 March 1999 Date: 3 April 1999 Date: 3 May 1999 </pre>
23.	<pre> Speed: 0 Solar radiation: 0 Ambient air temperature: 0 </pre>
24.	<pre> 5 10 15 </pre>
25.	<pre> 5 10 15 </pre>
26.	<pre> 5 10 15 </pre>

Test no.	Actual output
27.	 output - Notepad File Edit Format View Help Speed: 6 Solar radiation: 200 Ambient air temperature: 20.75 Speed: 5 Solar radiation: 250 Ambient air temperature: 21.55 Speed: 6 Solar radiation: 300 Ambient air temperature: 22.65
28.	 C:\Users\USER\Downloads\ICT283-C++\Practical_Work\ICT283_Assignment1_Tee_Yee_Kang_34315323\bin\Debug\Assessment.exe ----- 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]: 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]: 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]: Exit the program Enter the option you want to perform: 5 Thank you bye ~ Process returned 0 (0x0) execution time : 2.792 s Press any key to continue.
29.	 C:\Users\USER\Downloads\ICT283-C++\Practical_Work\ICT283_Assignment1_Tee_Yee_Kang_34315323\bin\Debug\Assessment.exe ----- 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]: Exit the program Enter the option you want to perform: 9 Invalid option !

Test no.	Actual output
30.	 <pre> Date: 1 March 2014 Time: 9:0 Speed: 8 Solar radiation: 616 Ambient air temperature: 29.56 Date: 1 March 2014 Time: 9:10 Speed: 7 Solar radiation: 647 Ambient air temperature: 30.56 </pre>
31.	<pre> ----- 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]: Exit the program 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 </pre>
32.	<pre> ----- 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]: Exit the program Enter the option you want to perform: 1 Enter the year to search: 2014 Enter the month to search: 1 January 2014: No Data </pre>
33.	<pre> Enter the option you want to perform: 2 Enter the year to search: 2014 2014 January : No Data 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
34.	<pre> Enter the option you want to perform: 2 Enter the year to search: 2020 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 </pre>
35.	<pre> 2015 January : 254.7 kWh/m^2 February : 197.4 kWh/m^2 March : 0.9 kWh/m^2 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>
36.	<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																																																																														
37.	<table><tr><td>2</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td>2014</td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td>March</td><td>20.3</td><td>22.8</td><td>183.5</td><td></td></tr><tr><td>5</td><td>April</td><td>13.7</td><td>19.3</td><td>137.4</td><td></td></tr><tr><td>6</td><td>May</td><td>17.1</td><td>16.5</td><td>86.3</td><td></td></tr><tr><td>7</td><td>June</td><td>4.7</td><td>13.2</td><td>79.5</td><td></td></tr><tr><td>8</td><td>July</td><td>12.7</td><td>13.5</td><td>84.1</td><td></td></tr><tr><td>9</td><td>August</td><td>19</td><td>15.5</td><td>112.3</td><td></td></tr><tr><td>10</td><td>September</td><td>20.6</td><td>16.1</td><td>145</td><td></td></tr><tr><td>11</td><td>October</td><td>18.9</td><td>17.8</td><td>200.6</td><td></td></tr><tr><td>12</td><td>November</td><td>20.3</td><td>19</td><td>220.1</td><td></td></tr><tr><td>13</td><td>December</td><td>21.7</td><td>21.2</td><td>268.6</td><td></td></tr><tr><td>14</td><td></td><td></td><td></td><td></td><td></td></tr></table>	2						3	2014					4	March	20.3	22.8	183.5		5	April	13.7	19.3	137.4		6	May	17.1	16.5	86.3		7	June	4.7	13.2	79.5		8	July	12.7	13.5	84.1		9	August	19	15.5	112.3		10	September	20.6	16.1	145		11	October	18.9	17.8	200.6		12	November	20.3	19	220.1		13	December	21.7	21.2	268.6		14					
2																																																																															
3	2014																																																																														
4	March	20.3	22.8	183.5																																																																											
5	April	13.7	19.3	137.4																																																																											
6	May	17.1	16.5	86.3																																																																											
7	June	4.7	13.2	79.5																																																																											
8	July	12.7	13.5	84.1																																																																											
9	August	19	15.5	112.3																																																																											
10	September	20.6	16.1	145																																																																											
11	October	18.9	17.8	200.6																																																																											
12	November	20.3	19	220.1																																																																											
13	December	21.7	21.2	268.6																																																																											
14																																																																															
38.	<div><div>AutoSave Off   </div><div>WindTempSolar</div><div>FileHomeInsertPage LayoutFormulasDataReviewViewHelp</div><div><div> Paste  Cut  Copy  Format Painter</div><div>Clipboard</div><div>Calibri11A⁺A⁻ BBIUA</div><div>Font</div><div> Wrap Te  Merge 8</div><div>Alignment</div></div><div><div> POSSIBLE DATA LOSS Some features might be lost if you save this workbook in the comma-delimited (.csv) format.</div></div><div><div>A1</div><div></div><table><tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th><th>E</th><th>F</th><th>G</th><th>H</th><th>I</th></tr><tr><td>1</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>2</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>3</td><td>2021</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr><tr><td>4</td><td>No data</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr></table></div></div>		A	B	C	D	E	F	G	H	I	1										2										3	2021									4	No data																																				
	A	B	C	D	E	F	G	H	I																																																																						
1																																																																															
2																																																																															
3	2021																																																																														
4	No data																																																																														
39.	<div>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</div>																																																																														

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: