# Data Dictionary

**Bst.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Node<T> | Struct |  | A Data type that represents a Node in a Binary Search Tree |  |
| value | T | + | The value the Node is holding |  |
| leftNode | Node<T>\* | + | The pointer of a Node | Represents the left node of a BST |
| rightNode | Node<T>\* | + | The pointer of a Node | Represents the right node of a BST |
| Bst<T> | Template |  | A template class that serves as a Binary Search Tree |  |
| m\_root | T\* | - | pointer of root Node for the BST |  |
| Bst () |  | + | Constructor |  |
| Bst (const Bst<T>&) |  | + | Copy constructor, deep copies |  |
| ~Bst () |  | + | Destructor that will delete the pointers |  |
| DeleteTree () | void | + | Deletes entire tree |  |
| operator = (const Bst<T>&) | Bst<T>& | + | Same as the copy constructor, deep copies |  |
| Search (const T&) | bool | + | Search Tree to determine if a given Node's values exist. Returns true or false |  |
| Insert (const T&) | bool | + | Inserts a Node holding the given value into the tree |  |
| DeleteNode (const T&) | bool | + | Search through the tree and deletes a node based on a given value |  |
| TreeNodeCount () | int | + | Returns the count of Nodes in a tree |  |
| InOrderTraversal () | void | + | Traverse a tree in an in-order way |  |
| PreOrderTraversal () | void | + | Traverse a tree in a pre-order way |  |
| PostOrderTraversal () | void | + | Traverse a tree in a post order way |  |
| InOrderTraversal (void (\*visit) (T&)) | void | + | Traverse a tree in an in-order way. Takes in a function as parameter. | Used to get entire tree’s (Specific day) Weather's data |
| PreOrderTraversal (void (\*visit) (T&)) | void | + | Traverse a tree in a pre-order way. Takes in a function as parameter. | Used to get entire tree’s (Specific day) Weather's data |
| PostOrderTraversal (void (\*visit) (T&)) | void | + | Traverse a tree in a post order way. Takes in a function as parameter. | Used to get entire tree’s (Specific day) Weather's data |
| GetSumValueFloat () | float | + | Returns the sum of float values contained in tree | Kept allowing user to use function for float trees |
| GetSumValueFloat (float (\*visit) (T&)) | float | + | returns the total of float values contained in tree. Takes in a function as parameter. | Used to find the total value of a float data in Weather in the entire tree. Which data to get is determined by function |
| template <typename Func> InOrderTraversalFloat (Func visit) | float | + | Return the result of applying the custom function to each node's value in the BST. | Performs an in-order traversal of the BST and applies a custom function to each node's value, returning a float.  Need for calculating standard deviation. |
| CopyTree (Node<T>\* &, Node<T>\*) | void | - | Deep copies a given tree to another tree |  |
| Insert (Node<T>\* &, Node<T>\* &) | void | - | Inserts a Node holding the given value into the tree |  |
| Search (const Node<T>\*, const T&) | bool | - | Search Tree to determine if a given Node's values exist. Returns true or false |  |
| Destroy (Node<T>\* &) | void | - | Delete all Nodes downstream of a given node |  |
| DeleteFromTree (Node<T>\* &) | void | - | Search through the tree and deletes a node based on a given value |  |
| NodeCount (Node<T>\*) | int | - | Returns the count of Nodes in a tree |  |
| InOrder (Node<T>\*) | void | - | Traverse a tree in an in-order way |  |
| PreOrder (Node<T>\*) | void | - | Traverse a tree in a pre-order way |  |
| PostOrder (Node<T>\*) | void | - | Traverse a tree in a post order way |  |
| InOrder (Node<T>\*, void(\*visit) (T&)) | void | - | Traverse a tree in an in-order way. Takes in a function as parameter. |  |
| PreOrder (Node<T>\*, void (\*visit) (T&)) | void | - | Traverse a tree in a pre-order way. Takes in a function as parameter. |  |
| PostOrder (Node<T>\*, void (\*visit) (T&)) | void | - | Traverse a tree in a post order way. Takes in a function as parameter. |  |
| SumValueFloat (Node<T>\*, float&) | float | - | Returns the sum of float values contained in tree |  |
| SumValueFloat (Node<T>\*, float(\*visit) (T&), float&) | float | - | returns the total of float values contained in tree. Takes in a function as parameter. |  |
| template <typename Func>  InOrderFloat (Node<T>\*, Func visit) | float | - | Return the result of applying the custom function to each node's value in the BST from the given node | Performs an in-order traversal of the BST from a given node and applies a custom function to each node's value, returning a float.  Need for calculating standard deviation. |

**Date.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Date | class |  | A class for storing date | Each object is used to hold the date information after splitting them into the date components |
| m\_day | int | - | Stores day of the date |  |
| m\_month | int | - | Stores the month of the date |  |
| m\_year | int | - | Stores the year of the date |  |
| Date () |  | + | Constructor, sets m\_day, m\_month, m\_year to -1 |  |
| Date (unsigned int d, unsigned int m, unsigned int y) |  | + | Constructor, sets m\_day, m\_month, m\_year to what was given |  |
| ~Date () |  | + | Destructor, will not do anything as no pointers are used |  |
| GetDay () | int | + | Getter for m\_day |  |
| GetMonth () | int | + | Getter for m\_month |  |
| GetYear () | int | + | Getter for m\_year |  |
| SetDay (const unsigned int d) | void | + | Setter for m\_day |  |
| SetMonth (const unsigned int m) | void | + | Setter for m\_month |  |
| SetYear (const unsigned int y) | void | + | Setter for m\_year |  |
| DisplayFullDate () | string | + | Returns a string containing the current date stored in DD/MM/YYYY format |  |

**Time.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Time | class |  | A class for storing time data | Each object will store a time in hour and minute |
| m\_hour | int | - | Stores hour of time in 24-hour time |  |
| m\_minute | int | - | Stores minute of time |  |
| Time () |  | + | Constructor sets m\_hour and m\_minute to -1 |  |
| Time (unsigned int h, unsigned int m) |  | + | Constructor sets m\_hour and m\_minute to given value |  |
| ~Time () |  | + | Destructor does nothing as no pointers are used |  |
| GetHour () | int | + | Getter for m\_hour |  |
| GetMinute () | int | + | Getter for m\_minute |  |
| SetHour (const unsigned int h) | void | + | Setter for m\_hour |  |
| SetMinute (const unsigned int m) | void | + | Setter for m\_hour |  |
| Display24HTime () | string | + | Returns a string containing the current time stored in object in HH:MM format |  |
| operator < (const Time&, const Time&) | bool | + | Operator overload of <. Compares hours and minute of 2 Time objects. | Required for Inserting Weather object into BST. |
| operator > (const Time&, const Time&) | bool | + | Operator overload of >. Compares hours and minute of 2 Time objects. | Required for Inserting Weather object into BST. |
| operator == (const Time&, const Time&) | bool | + | Overloaded operator for ==. Equate 2 Time objects. | Required for changing the BST insert from void to bool. |

**Weather.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Weather | Class |  | Object comprising of Date, Time, WindSpeed (float), SolarRadiation (float), AmbientAirTemperature (float). | Each object is used to hold a row of data from log data file. Holds the data as-is so that recreating the data is possible and there will be no loss of information |
| m\_date | Date | - | The date of a Log record |  |
| m\_time | Time | - | The time of Log record |  |
| m\_windSped | float | - | The Wind speed of Log record |  |
| m\_solarRad | float | - | The Solar Radiation of Log record |  |
| m\_airTemp | float | - | Ambient Air Temperature of Log record |  |
| Weather () |  | + | Constructor |  |
| ~Weather () |  | + | Destructor |  |
| GetDate () | Date& | + | Getter for m\_date |  |
| GetTime () | Time& | + | Getter for m\_time |  |
| GetWindSpeed () | float | + | Getter for m\_windsped |  |
| GetSolarRadiation () | float | + | Getter for m\_solarRad |  |
| GetAirTemperature () | float | + | Getter for m\_airtemp |  |
| SetDate (const Date& date) | void | + | Setter for m\_date |  |
| SetTime (const Time& time) | void | + | Setter for m\_time |  |
| SetWindSpeed (const float windSpeed) | void | + | Setter for m\_windsped |  |
| SetSolarRadiation (const float solar) | void | + | Setter for m\_solarRad |  |
| SetAirTemp (const float temp) | void | + | Setter for m\_airtemp |  |
| operator < (const Weather&, const Weather&) | bool | + | Operator overload for <. Will call < operator of Time Class | Required for Inserting Weather object into BST. |
| operator > (const Weather&, const Weather&) | bool | + | Operator overload for >. Will call > operator of Time Class | Required for Inserting Weather object into BST. |
| operator == (const Weather&, const Weather&) | bool | + | Operator overload for ==. Will call == operator of Time Class | Required for changing BST insert from void to bool. |

**Validator.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Validator | helper |  | A helper class used for validating incoming data that are being stored |  |
| IsValidDate (const int& day, const int& month, const int& year) | bool | + | Checks if a given set of date is valid. Day is 1-31 only Month is 1-12 only Year is 1000-9999 only If year is a leap year, then February would be checked that day is 1-29 instead of 1-28 Returns true if valid and false if invalid |  |
| IsValidDate (const int& month, const int& year) | bool | + | Checks if a given set of month and year are valid Month is 1-12 only Year is 1000-9999 only Returns true if valid and false if invalid |  |
| IsValidMonth (const int& month) | bool | + | Checks if given int is a valid month. Month is 1-12 only |  |
| IsLeapYear (const int& year) | bool | + | Checks if a given year is a leap year Returns true if it is and false if not |  |
| IsValid24HourTime (const int& hour, const int& minute) | bool | + | Checks if a given time is valid. Hour 0-23 only Time 0-59 only Return true if valid and false if not |  |
| IsNumber (const string& input) | bool | + | Checks if a given string is a valid number A number is valid if it contains digits and at most 1 decimal point The decimal point must be after at least 1 digit Returns true if valid and false if not |  |
| IsInteger (const string& input) | bool | + | Checks if a given string is a valid integer A integer is valid if it contains only digits Returns true if valid and false if not |  |

**Logic.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| LogicHelper |  |  | A helper class used for holding functions relating to handling of logic | Class is used for things such as calculation, conversion, loading of data file and creating and writing of output file. This class will not deal with input and output to terminal |
| LoadMultiDataFileCSV (ifstream&, map<int, map<int, map<int, Bst<Weather>>>>&) | void | + | Allows for loading of multiple data files by taking in a file with a list of file names. Data is loaded into a map of map of map of BST of weather | file is expected to have data file name to be read, it will call LoadDataFileCSV function for each file. |
| LoadDataFileCSV (ifstream&, map<int, map<int, map<int, Bst<Weather>>>>&) | void | + | Takes in the file and a map It will loop through the file line by line and get only the date, time, wind speed, solar radiation, and air temperature data Each line of data will be stored in a weather object which will be stored in a BST All BST will be stored in map of map of map | The maps of map of map represents year, month, and day, mimicking how a calendar works. |
| InsertWeatherToMap (map<int, map<int, map<int, Bst<Weather>>>>&, Weather&) | void | + | Inserts tempLog into wind log |  |
| RemoveLeadingZeros (string input) | string | + | Removes leading 0s from a given string | This is to prevent octal integer error when storing data into integer variables |
| AverageWindSpeedMonth (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int) | float | + | Search and calculates the average wind speed of a month when given the month and year to search for |  |
| AverageAirTemperatureMonth (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int) | float | + | Search and calculates the average air temperature of a month when given the month and year to search for |  |
| StandardDeviationWindSpeed (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int) | float | + | Calculate and return standard deviation of wind speed for the specified month and year | Needed for option 1 |
| StandardDeviationAirTemperature (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int) | float | + | Calculate and return standard deviation of air temperature for the specified month and year | Needed for option 2 |
| SumSolarRadiationMonth (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int) | float | **+** | Search and calculates the total solar radiation of a month when given the month and year to search for |  |
| IntMonthToString (const int&) | string | + | Converts a given integer month to string month | For converting stored int month (1-12) to string (January-December) for output |
| ConvertWindMStoKMH (const float&) | float | + | Multiplies a given float by 3.6 | This will help convert stored windSpeed which is m/s to km/h |
| ConvertSolarRadiationWMtoKWH (const float&) | float | + | Divides a given float by 6 and 1000 | This will help convert stored solarRadiation which is in W/10mins to KWH |
| WriteAppendFile (const string&) | void | + | Appends a given string to "WindTempSolar.csv" |  |
| calculateSPCC (const Vector<float>&, const Vector<float>&) | float | + | This method calculates Sample Pearson Correlation Coefficient between two vectors of floating-point numbers. | Needed for new option 3 |
| extractMonthData (const map<int, map<int, map<int, Bst<Weather>>>>&, const int, const int, Vector <float>&, Vector <float>&, Vector <float>&) | void | + | This method extracts wind speeds, air temperatures, and solar radiation data for a specified month and year from the wind log | Needed for calculating SPCC of three different combinations needed for option 3 |

**FunctionHelper.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| FunctionHelper |  |  | A helper class with functions that will each represent a menu option + a display menu function |  |
| DisplayMenu () | string | + | Displays the menu |  |
| FindAverageWindSpeedAndStandardDeviationMonth (map<int, map<int, map<int, Bst<Weather>>>>&, int, int) | string | + | Returns the string result of option 1 of menu. Takes in the map of data, the month input and year input. |  |
| FindAverageAirTemperatureAndStandardDeviationMonth (map<int, map<int, map<int, Bst<Weather>>>>&, int, int) | string | + | Returns the result needed for option 2 of menu. Takes in the map of data, the month input and year input. |  |
| FindAverageTemperatureYear (map<int, map<int, map<int, Bst<Weather>>>>&, int) | string | + | Returns the string result of option 2 of menu. Takes in map of data and the year input. |  |
| FindsPCCForGivenMonth (map<int, map<int, map<int, Bst<Weather>>>>&,) | string | + | Returns the string result of option 3 of menu. Takes in map of data and the month input. |  |
| FindSumSolarRadMonth (map<int, map<int, map<int, Bst<Weather>>>>&, int) | string | + | Returns the result needed for option 4 of menu. Takes in map of data and the year input |  |
| OutputFileAverageWindTempSolarMonth (map<int, map<int, map<int, Bst<Weather>>>>&, int) | void | + | Outputs a CSV file of data result from option 4 of menu. Takes in map of data and the year input |  |
| ReloadData (map<int, map<int, map<int, Bst<Weather>>>>&) | void | + | Reload the data for data/data\_source.txt file and process data. |  |

**BstWeatherHelper.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| BstWeatherHelper |  |  | A class that holds the functions to be passed into BST as parameter for varies tasks | To keep other functions, separate for neatness |
| GetWeatherWindSpeed (Weather&) | float | + | Get wind speed data from a weather object |  |
| GetWeatherAirTemp (Weather&) | float | + | Get Air temperature from a weather object |  |
| GetWeatherSolarRad (Weather&) | float | + | Get solar radiation from a weather object |  |

**Vector.h**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name** | **Type** | **Protection** | **Description** | **Rationale** |
| Vector | helper |  | A wrapper class for std:vector. | A class need for the calculation of standard deviation and most importantly option 3 to find SPCC of 3 different combinations. |
| m\_data | vector<T> | - |  |  |
| Vector () |  | + | Default constructor |  |
| Vector (size\_t) |  | + | Constructor that creates a vector with a specified size |  |
| Vector (size\_t, const T&) |  | + | Constructor that creates a vector with a specified size and initial value. |  |
| Vector (const Vector<T>&) |  | + | Copy constructor |  |
| Vector (Vector<T>&&) noexcept |  | + | Move constructor |  |
| ~Vector () |  | + | destructor |  |
| operator= (const Vector<T>&) | Vector<T>& | + | Copy assignment operator |  |
| operator= (Vector<T>&&) noexcept | Vector<T>& | + | Move assignment operator |  |
| Operator [] (size\_t) | T& | + | Accesses the element at the specified index |  |
| Operator [] (size\_t) | const T& | + | Accesses the element at the specified index (const version) |  |
| Size () | size\_t | + | Returns the number of elements in the vector |  |
| capacity () | size\_t | + | Returns the capacity of the vector |  |
| empty () | bool | + | Checks if the vector is empty |  |
| push\_back (const T&) | void | + | Adds an element to the end of the vector |  |
| push\_back (T&&) | void | + | Adds an element to the end of the vector |  |
| pop\_back () | void | + | Removes the last element of the vector |  |
| Insert (size\_t, const T&) | void | + | Inserts an element at a specified position |  |
| Erase (size\_t) | void | + | Removes an element at a specified position. |  |
| at (size\_t) | T& | + | Accesses the element at a specified index |  |
| at (size\_t) | const T& | + | Accesses the element at a specified index (const version) |  |
| Reserve (size\_t) | void | + | Reserves storage for at least the specified number of elements |  |
| resize(size\_t) | void | + | Changes the size of the vector |  |
| resize (size\_t, const T&) | void | + | Changes the size of the vector and initializes new elements |  |
| begin () | iterator | + | Returns an iterator pointing to the first element of the vector |  |
| end () | iterator | + | Returns an iterator pointing to the past-the-end element of the vector |  |
| begin () | const\_iterator | + | Returns a const iterator pointing to the first element of the vector |  |
| end () | const\_iterator | + | Returns a const iterator pointing to the past-the-end element of the vector |  |
| cbegin () | const\_iterator | + | Returns a const iterator pointing to the first element of the vector |  |
| cend () | const\_iterator | + | Returns a const iterator pointing to the past-the-end element of the vector |  |
| Vector (std::initializer\_list<T>) |  | + | Constructor that initializes the vector with an initializer list |  |

# Algorithm

## High Level

1. Get .txt file with list of data file names to read (data/data\_source.txt)
2. Read Index File line by line.
   1. Get .csv file with weather data.
   2. Read .csv file line by line.
      1. Read column by column.
      2. Get DateTime (WAST)
         1. Split DateTime into data and time
         2. Store Date in Date class.
         3. Store Time in Time Class
      3. Get WindSpeed (S)
      4. Get Solar Radiation (SR)
      5. Get Ambient Air Temperature (T)
   3. Repeat for each line.
3. Repeat for each file name in Index File
4. Store each line of extracted data into Weather object.
5. For each Weather of a year
   1. Store in Map<Year, Map<Month, Map<Day, Bst<Weather>\* >\* >\* > (Bst sort by time)
6. Display menu.
7. If option 1 is chosen
   1. Ask user for month and year input.
   2. Go to corresponding year and month of Maps and get all Weather objects.
   3. Sum all the WindSpeed and count the number of Weather objects.
   4. Divide WindSpeed by count to get average and calculate the standard deviation.
   5. Convert windspeed from m/s to km/h.
   6. Print result.
8. If Option 2 is chosen
   1. Ask user for year input.
   2. Go to corresponding year of Maps and get all weather object.
   3. Sum all air temperature and count the number of Weather that matches.
   4. Divide air temperature by count to get average and calculate the standard deviation.
   5. Print result.
9. If option 3 is chosen
   1. Ask user for month input.
   2. Go to corresponding years of maps and get all weather object.
   3. Get average wind speed, average air temperature and average solar radiation.
   4. Calculate the SPCC of 3 different combinations of S\_T, S\_R, T\_R.
   5. Print result.
10. If option 4 is chosen
    1. Ask user for year input.
    2. Go to corresponding year of maps and get all weather object.
    3. Get average wind speed, standard deviation of wind speed, average air temperature, standard deviation of air temperature and total solar radiation.
    4. Print result to WindTempSolar.csv.
11. If option 5 is chosen
    1. Clear the existing data.
    2. Do step 1-6.
12. If option 6 is chosen
    1. Exit program.

# Rationale

Weather Log data are stored in map(year) of map(month) of map(day) of BST of Weather data that is inserted based on time. The rationale for the above structure is to emulate a calendar when storing data. A particular date will only be inserted into the maps only if a log data with that date exist. Since all the current functions are searching by date, we can improve search time of the functions by storing it separated by date at the cost of increasing data loading time.

Another method would be to store Weather Log data in a map with datetime as key while storing the datetime in BSTs with one BST per year. This will improve loading time at the cost of longer search time instead as the program would have to search through the BST to look for if data with that date exist before finding the actual Weather data in map by searching through the map for the date again.