Software Requirements Specification

for

Temperature Registration System

Version 2.0 approved

Prepared by
Team iAlpha!

Kaki Wong – 9230971
Samson Xian – 6163378
Zhaoyu Zhang – 9097260

Concordia University

Novemeber 12th 2010

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Kaki Wong, Samson Xian	Oct 9, 2010	Document creation	1.0
KaKi Wong	Nov 12, 2010	Stack, Queue bi-directional linked list and Binary tree data structures added	2.0

1. Introduction

1.1 Purpose

The document defines the operational requirement for the temperature registration system (TRS). This SRS describes the software functional requirements for release 1.0 of the system. This document is intended to be used by the members of the development teams and testing teams.

1.2 Project Scope and Features

The system is an application that permits users to manage the weather condition by means of data insertion, deletion, sorting, and display etc. The application is also a demonstration of how the information related to weather condition is handled by basic data structures of arrays, unidirectional linked lists, and bidirectional linked lists.
The application features in providing a friendly text-based user interface which allows user to select data structure and operation they want to work with.

1.3 References

- COEN 345 – Software Testing and Validation(Fall 2010) Project description
  https://moodle.concordia.ca/moodle/file.php/35223/Project.pdf

- General Outline of a SRS
  http://en.wikipedia.org/wiki/Software_Requirements_Specification

- Software Requirement Specification Template
  http://www.processimpact.com/process_assets/srs_template.doc

2. Overall Description

2.1 Product Requirements

The list of requirements that the system is expected to fulfilled is included in this section. Each of the requirements has a unique identifier for testing purpose.

2.1.1 Data Structure Selections

R-1 - Array data structure can be selected to store data.

R-2 - Unidirectional Linked list data structure can be selected to store data.

R-3 - Bi-directional Linked list data structure can be selected to store data

R-4 - Stack Bi-directional Linked list data structure can be selected to store data

R-5 - Queue Bi-directional Linked list data structure can be selected to store data

R-6 - Binary tree structure can be selected to store data.

2.1.2 User Allowed Actions

R-7 - User can enter new entry to the list.

R-8 - User can enter multiple entries to the list

R-9 - User can delete entry from the list

R-10 - User can delete multiple entries from the list

R-11 - By entering a date, the system display the temperature store for that date

R-12 - The list can be sorted in ascending (older entry on top) or descending order (newest entry on top)

R-13 - The order of the entries can be shuffled in random order

R-14 - The current list of entries can be displayed on the screen

R-15 - By entering 2 dates, the system outputs the date with the highest temperature

R-16 - The system can convert temperature from Celsius to Fahrenheit

R-17 - When the user enter a reference temperature, the system output the number of date which the temperature is above that reference point

R-18 - The user enters 2 temperatures to the system and the system counts the number of days having temperature in between these 2 reference points. The system expects t1 to be lower than t2.

2.1.3 System Restrictions

R-19 - The users navigate through the program using the text-based interface.

R-20 - The user cannot input a date that does not exist in the calendar

R-21 - The user has to enter date in integer format as follows: YYYYMMDD

R-22 - Date will older than Jan 1, 1900 and later than Dec 31, 2020 will not be allowed

R-23 - The user should only enter the temperature in integer format ranging from -60°C to 60°C

R-24 - The user cannot enter duplicate date in system

R-25 - In Array data structure, the user cannot enter more than 500 entries.

2.2 Operating Environment

OE-1: The application should be executable on Windows OS without the support of third-party software.

OE-2: The application shall be complied with Visual Studio on Windows environment or Eclipse on Linux.

2.3 Design and Implementation Constraints

The Temperature Registration System shall be designed under the following constraints.

CT-1: TRS software must be version labeled both in source and binary form.

CT-2: The UI should be comprehensive and consistent with the functional requirements.

CT-3: All three data structures must provide the same functionalities as described.

2.4 User Documentation

No user documentation is provided with the software; operating instructions are built in the system text-based menu

2.5 Assumptions and Dependencies

AS-1: It is assumed that any use of the application will occur in an environment with full compliance to this specification.

AS-2: It is assumed that the application only accept the date entry from year 1990 to 2100.

AS-3: It is assumed that temperature is in range of -60℃ to 60℃.

AS-4: It is assumed that the Celsius-Fahrenheit conversion respects the standard formula of

$$F = C \times (9/5) + 32$$

3. Specific Requirements

3.1 System Functions

The list of functions included in the system is listed below with its description, user input and expected system output. Each section is broken down by list of classes and files.

3.1.1 main.cpp

F-1 - `void submenu(int ch)`

This function is to generate a submenu on the screen after the user select which data structure the want to use to store the data. The input requires is the choice of data structure in integer format. The output will generate a text-based menu and store this selection for subsequent operations.

F-2 - `double convertCtoF(int c)`

This function will convert the temperature from Celsius to Fahrenheit. The user input is the date (index) to find the corresponding temperature stored in the system. The system will use the temperature in Celsius and generate an output in Fahrenheit.

F-3 - `bool checkDate ( int date )`

This function checks if the date is within the predefined range of 19000101 and 21001231. It also checks to see if the user's input is a valid date in the calendar. The user has to input be in YYYYMMDD format and a valid calendar date. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

F-4 - `bool checkTemp ( int temp )`

This function checks if the user input for temp is within the range of -60 and +60. The input will be passed to the function as an integer. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

3.1.2 Array - iArray.cpp

F-5 - `iArray::iArray(void)`

This function set the counter of the Array to 0; it is also the constructor function for iArray class.

F-6 - `bool iArray::add(int date, int temperature)`

The function performs an addition to the array with the 2 parameters: date and temperature. The 2 inputs required by the user are date and temperature both in integer type. The output of the is an addition to the array with provided data.

F-7 - `bool iArray::add(weather w)`

This function is called at the same time as the previous function when an element is being inserted into the array. Each time the function is called, it will increase the array counter by one, therefore pointing to an empty array for data insertion.

F-8 - `bool iArray::addset()`

This function performs the same way as “add” function, except it allows entry of multiple elements. The inputs from the user are the date as the index and temperature as the data. When the user input 0 for both index and temperature, the input requests terminate. The output of the function will store multiple items in the array.

F-9 - `bool iArray::remove(int idate)`

The remove function will delete an entry from the array list based on the index date. The user inputs the date to function. The output is the removal of the entry from the array.

F-10 - `bool iArray::removeset()`

The removeset functions remove multiple entries at once. The system expects a date input in integer or a “0” to terminate the function. The output is removal of the entries from the array.

F-11 - `bool iArray::shuffle()`

The shuffle function puts the order of the array in random order. No user input is needed other than calling the function. The output of the function will rearrange the array in a random order.

F-12 - `bool iArray::sort(char type)`

The sort function will re-order the array in an ascending or descending order depending on user's input. The user input needed is the type of sorting; either ascending or descending order. The output of the function will rearrange the array in the specified order.

F-13 - `bool iArray::display()`

The display function will show the data (date and temperature) on screen. No user input is needed other than calling the function. The output of the function will display the current entries stored in the array.

F-14 - `int iArray::highesttemperature(int date1,int date2)`

The highesttemperature function will compare the temperature of 2 specific dates and display the one that has a higher temperature registered. The user inputs required are the 2 dates from the existing array. The output will be the date which has a higher temperature.

F-15 - `int iArray::numberofdaysovertemperature(int tempe)`

The numberofdaysovertemperature function will display the number of days that has a temperature above a threshold given by the user. The user is required input a temperature to the function. The output of the function will give a count number of days having temperature higher than the threshold.

F-16 - `int iArray::numberofdaystemperaturebetween(int temp1,int temp2)`

This function takes 2 temperatures value and output the number of day having registered temperature within the range. The input required from the users is 2 temperature values where the

first one is the minimum temperature and the second one is the maximum temperature. The output of the function will give a count number of days having temperature between this range.

F-17 - `int iArray::getTemperature(int date)`

This function will output the temperature given the date provided by the user. The user is required to input the date. The output of the function will output the temperature of the specified date in Celsius.

F-18 - `bool iArray::dateExist(int date)`

This function check if the date given by the user is already stored in the array. The input required from the user is the date. The output of the function will display a Boolean true or false depending on the input.

F-19 - `bool iArray::checkDate ( int date )`

This function checks if the date is within the predefined range of 19000101 and 21001231. It also checks to see if the user's input is a valid date in the calendar. The user has to input be in YYYYMMDD format and a valid calendar date. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

F-20 - `bool iArray::checkTemp ( int temp )`

This function checks if the date is within the predefined range of 19000101 and 21001231. It also checks to see if the user's input is a valid date in the calendar. The user has to input be in YYYYMMDD format and a valid calendar date. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

3.1.3 Unidirectional Linked-List – uLinkList.cpp

F-21 - `void uLinkList::insert(int index,int data)`

The function performs an insertion to the linked list with the 2 parameters: date and temperature. The 2 inputs required by the user are date and temperature both in integer type. The output of the function will insert the entry into the linked list.

F-22 - `void uLinkList::remove( int index )`

The remove function will delete an entry from the linked list based on the index date. The user inputs the date to function. The output is the removal of the entry from the array.

F-23 - `int uLinkList::search(int index)`

This function will output the temperature given the date provided by the user. The user is required to input the date as search index. The output of the function will output the temperature of the specified date in Celsius.

F-24 - `void uLinkList::sortByIndex(char order)`

The sort function will re-order the list in an ascending or descending order depending on user's input. The user input needed is the type of sorting; either ascending or descending order. The output of the function will rearrange the list in the specified order.

F-25 - `void uLinkList::display()`

The display function will show the data (date and temperature) on screen. No user input is needed other than calling the function. The output of the function will display the current entries stored in the array.

F-26 - `int uLinkList::getLength()`

This function is used to obtain the length of the list. No user input is needed other than calling the function. The output of the function is the length of the linked list. It is used in conjunction with shuffle and sort function.

F-27 - `bool uLinkList::nodeExisted(int index)`

This function check if the date given by the user is already stored in the array. The input required from the user is the date as index. The output of the function will display a Boolean true or false depending on the input.

F-28 - `void uLinkList::shuffle()`

The shuffle function puts the order of the list in random order. No user input is needed other than calling the function. The output of the function will rearrange the array in a random order.

F-29 - `int uLinkList::indexOfLargestBetween(int minIndex, int maxIndex)`

The indexOfLargestBetween function will compare the temperature of 2 specific dates and display the one that has a higher temperature registered. The user inputs required are the 2 dates from the existing array. The output will be the date which has a higher temperature.

F-30 - `int uLinkList::numOfNodeGt(int t)`

The numOfNodeGt function will display the number of days that has a temperature above a threshold given by the user. The user is required input a temperature to the function. The output of the function will give a count number of days having temperature higher than the threshold.

F-31 - `int uLinkList::numOfNodeBtw(int min, int max)`

This function takes 2 temperatures value and output the number of day having registered temperature within the range. The input required from the users is 2 temperature values where the first one is the minimum temperature and the second one is the maximum temperature. The output of the function will give a count number of days having temperature between this range.

3.1.4 Bidirectional Linked List – iBiDirectionalLinkedList.cpp

F-32 - `bool iBiDirectionalLinkedList::add(int date, int temperature)`

The function performs an insertion to the linked list with the 2 parameters: date and temperature. The 2 inputs required by the user are date and temperature both in integer type. The output of the function will insert the entry into the linked list.

F-33 - `bool iBiDirectionalLinkedList::addset()`

This function performs the same way as “add” function, except it allows entry of multiple elements. The inputs from the user are the date as the index and temperature as the data. When the user input 0 for both index and temperature, the input requests terminate. The output of the function will store multiple items in the array.

F-34 - `bool iBiDirectionalLinkedList::remove(int date)`

The remove function will delete an entry from the linked list based on the index date. The user inputs the date to function. The output is the removal of the entry from the array.

F-35 - `bool iBiDirectionalLinkedList::removeset()`

The removeset functions remove multiple entries at once. The system expects a date input in integer or a “0” to terminate the function. The output is removal of the entries from the array.

F-36 - `bool iBiDirectionalLinkedList::shuffle()`

The shuffle function puts the order of the list in random order. No user input is needed other than calling the function. The output of the function will rearrange the array in a random order.

F-37 - `bool iBiDirectionalLinkedList::sort(char type)`

The sort function will re-order the list in an ascending or descending order depending on user’s input. The user input needed is the type of sorting; either ascending or descending order. The output of the function will rearrange the list in the specified order.

F-38 - `bool iBiDirectionalLinkedList::display()`

The display function will show the data (date and temperature) on screen. No user input is needed other than calling the function. The output of the function will display the current entries stored in the array.

F-39 - `bool iBiDirectionalLinkedList::dateExist(int date)`

This function check if the date given by the user is already stored in the list. The input required from the user is the date. The output of the function will display a Boolean true or false depending on the input.

F-40 - `int iBiDirectionalLinkedList::highesttemperature(int date1, int date2)`

The highesttemperature function will compare the temperature of 2 specific dates and display the one that has a higher temperature registered. The user inputs required are the 2 dates from the existing array. The output will be the date which has a higher temperature.

F-41 - `int iBiDirectionalLinkedList::numberofdaysovertemperature(int tempe)`

The numberofdaysovertemperature function will display the number of days that has a temperature above a threshold given by the user. The user is required input a temperature to the function. The output of the function will give a count number of days having temperature higher than the threshold.

F-42 - `int iBiDirectionalLinkedList::numberofdaystemperaturebetween(int temp1, int temp2)`

This function takes 2 temperatures value and output the number of day having registered temperature within the range. The input required from the users is 2 temperature values where the first one is the minimum temperature and the second one is the maximum temperature. The output of the function will give a count number of days having temperature between this range.

3.1.5 Stack Bidirectional Linked List – iStack.cpp

F-43 - `void iStack::push(int date, int temperature)`

The push functions are solely implemented for Stack and Queue. It takes the integer input date and temperature and add the data at the end of the stack.

F-44 - `bool iStack::pop()`

The pop functions are solely implemented for Stack and Queue. It takes no input from the user and it removes the last element of the Stack (pointed by “tail”).

3.1.6 Queue Bidirectional Linked List – iQueue.cpp

F-45 - `void iQueue::push(int date,int temperature)`

The push functions are solely implemented for Stack and Queue. It takes the integer input date and temperature and add the data at the end of the Queue.

F-46 - `bool iQueue::pop()`

The pop functions are solely implemented for Stack and Queue. It takes no input from the user and it removes the first element of the Queue (pointed by “head”).

3.1.7 Binary Tree – binaryTree.cpp

F-47 - `void binaryTree::insert(int date,int temperature)`

The insert function is to add a new element to the binary tree. It takes the date and temperature from user's input and add the data at the end of the binary tree as a new leaf.

F-48 - `bool binaryTree::insertSet()`

The insertSet function will loop continuously for multiple insertions to the binary tree. The user inputs require are the date and temperature of each data set entered individually and "0" as the end of insertion character. The output is multiple additions of leaves to the binary tree.

F-49 - `void binaryTree::remove(int date)`

The remove function takes the date and removes the data from the binary tree if it exists. The user input the date of day they wish to remove and the system remove the entry from the binary if found.

F-50 - `bool binaryTree::removeSet()`

The removeSet function takes dates inputted by the user and removes all the leaves if they are found. The system expects the user to enter 0 to stop removeSet function. The output is the removal of leaves from the binary tree.

F-51 - `int binaryTree::getTemperature(int date)`

The getTemperature function returns the value of temperature stored in the date given by the user. The input is the date and the output is the display of the temperature data in Celsius and Fahrenheit of the specified date.

F-52 - `void binaryTree::sortByIndex(char ch)`

The sortByIndex function will sort the tree depending on the sorting command given by the user. A "A" for ascending order or a "D" for descending order is expected to be inputted by the user. The output is the sorted tree based on the command given.

F-53 - `void binaryTree::shuffle()`

The shuffle function will store the content of the tree to a linked list and shuffle it in a random order and re-apply the data back to the same tree. No user input is required except calling the function. The expected output is tree in random order.

F-54 - `void binaryTree::display()`

The display function will show the content of the tree. No user input is required except calling the function. The expected output is tree in order.

F-55 - `double binaryTree::highesttemperature(int date1,int date2)`

The highesttemperature function will compare the temperature of 2 specific dates and display the one that has a higher temperature registered. The user inputs required are the 2 dates from the existing array. The output will be the date which has a higher temperature.

F-56 - `int binaryTree::numberofdaysovertemperature(int t)`

The numberofdaysovertemperature function will display the number of days that has a temperature above a threshold given by the user. The user is required input a temperature to the function. The output of the function will give a count number of days having temperature higher than the threshold.

F-57 - `int binaryTree::numberofdaystemperaturebetween(int temp_min,int temp_max)`

This function takes 2 temperatures value and output the number of day having registered temperature within the range. The input required from the users is 2 temperature values where the first one is the minimum temperature and the second one is the maximum temperature. The output of the function will give a count number of days having temperature between this range.

F-58 - `bool binaryTree::isEmpty()`

The isEmpty function checks if a binary tree is empty. It is only used by other member function for validate before doing other operations such as shuffle and remove.

F-59 - `bool binaryTree::dateExist(int date)`

The dateExist function checks if a given exist already in the binary tree. It is only used by other member function for validate before doing other operations such as insert.

F-60 - `bool binaryTree::checkDate ( int date )`

This function checks if the date is within the predefined range of 19000101 and 21001231. It also checks to see if the user's input is a valid date in the calendar. The user has to input be in YYYYMMDD format and a valid calendar date. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

F-61 - `bool binaryTree::checkTemp ( int temp )`

This function checks if the date is within the predefined range of 19000101 and 21001231. It also checks to see if the user's input is a valid date in the calendar. The user has to input be in YYYYMMDD format and a valid calendar date. The output of the function will generate a true or false. In the case of false, a request for input is generated again.

F-62 - `int binaryTree::getSize(treeNode* root)`

This function returns the size of the tree. It is only used by other member function such as insert to add a new leaf.

F-63 - `void binaryTree::inorder(treeNode* root)`

This function sorts the list in ascending order. It is only called by other member function "sort".

F-64 - `void binaryTree::preorder(treeNode* root)`

This function sorts the list in current order. It is only called by other member function "display".

F-65 - `void binaryTree::treeToQueue(treeNode* root)`

This function transfer the content of the tree to a Queue linked list for. It is only called by the "shuffle" function where the input will pull the data from the tree and shuffle around a linked list. The output is data restored in the original tree in random order.

F-66 - `void binaryTree::postorder(treeNode* root)`

This function sorts the list in descending order. It is only called by other member function "sort".

F-67 - `double binaryTree::findMax(treeNode* tPtr,int date1, int date2)`

This function returns the maximum temperature of the 2 dates given by user inputs and member function. It is only called by other member function "`highesttemperature`".

F-68 - `int binaryTree::countNodeGT(treeNode* tPtr,int t)`

This function counts the number of dates with temperature higher than a given. It is only called by other member function "`numberofdaysovertemperature`".

F-69 - `int binaryTree::countNodeBtw(treeNode* tPtr,int t_min, int t_max)`

This function counts the number of days between the 2 dates given by user inputs and member function. It is only called by other member function "`numberofdaystemperaturebetween`".

3.2 User Interfaces

The system provides friendly user interface. The UI contains two text-based menus that provide several options to be selected.

When the system initializes, the menu should be displayed as the figure below:

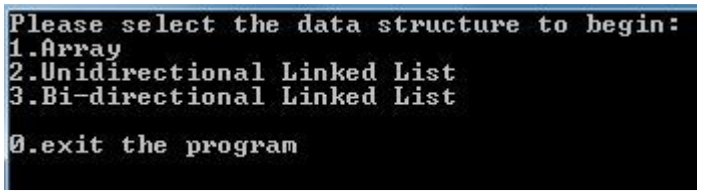

Figure 3.2.1 Main Menu

Once user selects one of data structure option, the submenu would appear as illustrated below:

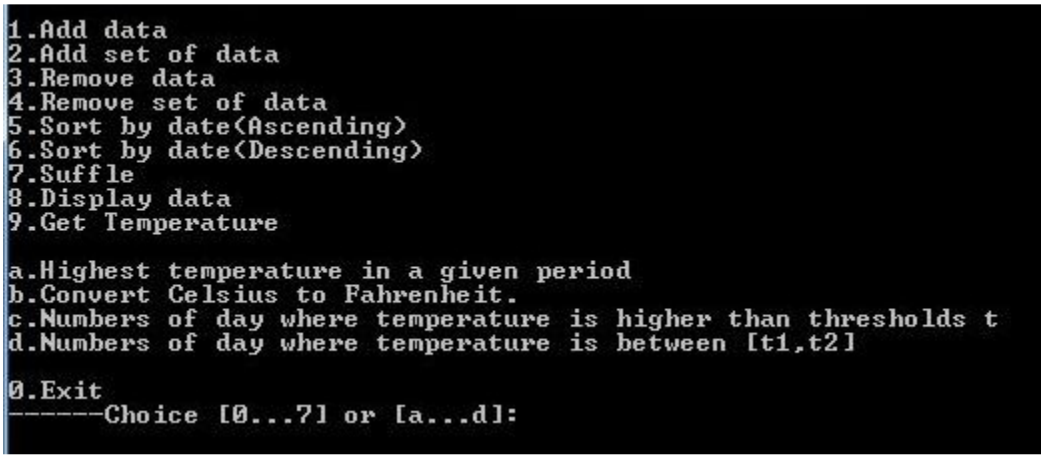

Figure 3.2.2 Submenu

Description of options

@ **Add data:** Prompt user for input date (YYYYMMDD) and temperature(C)

@ **Add set of data:** Prompt user for multiple inputs date (YYYYMMDD) and temperature(C)
*Note: the input method may vary depending on data structure

@ **Remove data:** Prompt user for input date and delete corresponding information

@ **Remove set of data:** Prompt user for multiple inputs date and delete corresponding information

@ **Sort by date (Ascending):** Sort data by Ascending order. Oldest data will be first displayed
*Note: this function only executes internal sorting. No display after sorting

@ **Sort by date (Descending):** Sort data by Descending order. Latest data will be first displayed
*Note: this function only executes internal sorting. No display after sorting

@ **Shuffle:** Randomly shuffle data
*Note: this function only executes internal shuffle. No display after shuffle

@ **Display Data:** Display entire list of data

@ **Get temperature:** Prompt user for input date and display the corresponding temperature

@ **Highest temperature in a given period:** Prompt user for input a start date and end date and output the maximum temperature in that period

@ **Covert Celsius to Fahrenheit:** Prompt user for input a temperature in Celsius and output in Fahrenheit

@ **Numbers of day where temperature is higher than threshold t:** Prompt user for input threshold temperature and output the dates where temperature are higher than the threshold

@ **Numbers of day where temperature is between [t1, t2]:** Prompt user for input the min temperature and max temperature and output the dates where temperature are in that range

@ **Exit:** Exit the program

3.3 Software Interfaces

The system is independent program that is executable under Windows OS environment. It doesn't require any third-party software support. However, user should have installed IDE in order to compile the program. The program is compliable both with Visual Studio in Windows OS and Eclipse in Linux OS.

Appendix A: Traceability Matrix

Test ID	Requirement ID	Function ID
	R1	F1
	R2	F1
	R3	F1
	R4	F1
	R5	F1
	R6	F1
	R7	F6, F21, F32, F43, F45, F47
	R8	F7, F21, F33, F48
	R9	F9, F22, F34, F44, F46, F49
	R10	F10, F22, F35, F50
	R11	F17, F23, F43, F51
	R12	F12, F24, F37, F52
	R13	F11, F28, F36, F53
	R14	F13, F25, F38, F54
	R15	F14, F29, F40, F55
	R16	F2
	R17	F15, F30, F41, F56
	R18	F16, F31, F42, F57
	R19	main(), F1
	R20	F3, F19, F44, F60
	R21	F3, F19, F44, F60
	R22	F3, F19, F44, F60
	R23	F4, F20, 45, F61
	R24	F18, F27, F39, F59
	R25	F7