Part B - Foundations   
  
**Dynamic Memory**

Workshop 3 (worth 3% of your final grade)

In this workshop, you are to process an array of objects where the user specifies the number of elements in the array at run-time.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities

* to allocate and deallocate dynamic memory for an array
* to use standard library functions to accept input data
* to use standard library functions to format output data

**SUBMISSION POLICY**

The “in-lab” section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the “in-lab” section along with your “at-home” section (a 20% late deduction will be assessed). The “at-home” portion of the lab is **due the day before you next scheduled workshop**

**IN-LAB SUBMISSION: ITEM Class (80%)**

Design and code a class named **Weather** that holds information about weather conditions. Place your class definition in a header file named **Weather.h** and your function definitions in an implementation file named **Weather.cpp**. Include in your coding all of the statements necessary for your code to compile under a standard C++ compiler.

The class **Weather** has the following two member variables:

* A 7-characters long C-style string (8 chars, including the NULL), holding the calendar date associated with the weather information. The date of formatted as follows: (Jan/21, Apr/1, …)
* A **double** value, representing the high temperature on the specified date
* A **double** value, representing the low temperature on the specified date

Your design includes the following member functions:

* **void set(const char\*, double, double) -** a modifier that stores valid data in a **Weather** object. The first parameter receives the address of a C-style string that holds the calendar date. The second and third parameters receive the low temperature, and high temperatures, respectively.
* **void display() const** - a query that displays the high and low temperature as well as the associated calendar date in the following format:
  + date left justified in a field of 10 using underscore '\_' instead of white space
  + high and low temperatures right justified in fields of 6 with 1 decimal place each using underscore '\_' instead of white space

[Hint: use the reference example in the notes to find the right manipulators.]

[Hint: **setw(int)** only affects the next item sent to the stream.]

**CLIENT MODULE**

Complete the following implementation file for the w3 main module. The missing regions of code are highlighted.

// OOP244 Workshop 3: Dynamic Memory

// File w3\_in\_lab.cpp

// Version 1.0

// Date 2015/05/30

// Author A.C. Coder

// Description

// This file is used to demonstrate dynamic memory in

// C++ and to process an array of objects of compound

// type where the user specifies the number of

// elements in the array at run-time.

/////////////////////////////////////////////////////

#include <iostream>

#include "Weather.h"

using namespace std;

using namespace sict;

int main(){

int n; //the count of days worth of weather

// initialize the weather pointer here

cout << "Weather Data\n";

cout << "=====================" << endl;

cout << "Days of Weather: ";

cin >> n;

cin.ignore();

// allocate dynamic memory here

for(int i = 0; i < n; i++){

char date\_description[7];

double high=0.0, low=0.0;

// ... add code to accept user input for

//weather

}

cout << endl;

cout << "Weather report:\n";

cout << "======================" << endl;

for(int i = 0; i < n; i++){

weather[i].display();

}

// deallocate dynamic memory here

return 0;

}

Output Example:

Weather Data

=====================

Days of Weather: 3

Enter date: Oct/1

Enter high: 15

Enter low : 10

Enter date: Nov/13

Enter high: 10

Enter low : 1.1

Enter date: Dec/15

Enter high: 5.5

Enter low : -6.5

Weather report:

======================

Oct/1\_\_\_\_\_\_\_15.0\_\_10.0

Nov/13\_\_\_\_\_\_10.0\_\_\_1.1  
Dec/15\_\_\_\_\_\_\_5.5\_\_-6.5

For submission instructions, see the [SUBMISSION](https://scs.senecac.on.ca/~oop244/pages/workshops/w2.html#sub) section below.

**In-Lab SUBMISSION**

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **Weather.h** and **Weather.cpp** and **w3\_in\_lab.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

**Sections SAA and SBB:  
~fardad.soleimanloo/submit\_w3\_in\_lab <ENTER>   
Section SCC and SDD:  
~ronald.burton/submit\_w3\_in\_lab <ENTER>**

and follow the instructions.

**AT-HOME SUBMISSION: Search (20%)**

After completing the solution described, copy w3\_in\_lab.cpp to w3\_at\_home.cpp and upgrade the definition of your class to include two more queries:

* **const char\* getDate() const,** this query returns the date of a **Weather** object.
* **double getLowTemp() const,** this query returns the low temperature of a **Weather** object.

Also, add the following global function to your client module, **w3\_at\_home.cpp**.

* **double findLow(const char\* qDate, Weather \*wData, int n);**

This function takes the following three arguments: The first parameter receives the address of a C-style string that represents the date of a Weather object. The second parameter receives the address of the array made in part one of the workshop. The third argument receives the size of the array made in part one of the workshop. This function searches for a date associated with a Weather object in the array and returns the low that day. If the date is not found in the data, return 0.0.

The main program that searches for the date in the array of Weather objects contains the following code.  The output from this program should exactly look like the following output example.

// OOP244 Workshop 3: Dynamic Memory

// File w3\_at\_home.cpp

// Version 1.0

// Date 2015/05/30

// Author A.C Coder

// Description

///////////////////////////////////////////////////////////

#include <iostream>

#include <cstring>

#include "Weather.h"

using namespace std;

using namespace sict;

double findLow(const char\* qDate, Weather \*wData, int n);

int main(){

int n; //the count of days worth of weather

// initialize the weather pointer here

cout << "Weather Data\n";

cout << "=====================" << endl;

cout << "Days of Weather: ";

cin >> n;

cin.ignore();

// allocate dynamic memory here

for(int i = 0; i < n; i++){

char date\_description[7];

double high=0.0, low=0.0;

// ... add code to accept user input for

//weather

}

cout << endl;

cout << "Weather report:\n";

cout << "======================" << endl;

for(int i = 0; i < n; i++){

weather[i].display();

}

//accept user input for the date to find

//(in this example stored in char query[7])

// and display the found low temprature.

double low = findLow(query, weather, n);

cout << "Low temperature : " << low << endl;

// deallocate dynamic memory here

return 0;

}

Output Example:

Weather Data

=====================

Days of Weather: 3

Enter date: Oct/1

Enter high: 15

Enter low : 10

Enter date: Nov/13

Enter high: 10

Enter low : 1.1

Enter date: Dec/15

Enter high: 5.5

Enter low : -6.5

Weather report:

======================

Oct/1\_\_\_\_\_\_\_15.0\_\_10.0

Nov/13\_\_\_\_\_\_10.0\_\_\_1.1  
Dec/15\_\_\_\_\_\_\_5.5\_\_-6.5

Enter the date you are looking for: Nov/13

Low temperature : 1.1

**AT-HOME SUBMISSION: REFLECTION (20%)**

Please provide brief answers to the following questions in a text file named **reflect.txt.**

1. What happens to dynamic memory if it is not deallocated?
2. What are the two methods of formatting IO stream data?
3. Why was dynamic memory required in the main() function of w3\_in\_lab.cpp to build the solution?

**SUBMISSION**

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **Weather.h**, **Weather.cpp, reflect.txt**  and **w3\_at\_home.cpp** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

**Sections SAA and SBB:  
~fardad.soleimanloo/submit\_w3\_at\_home <ENTER>   
Section SCC and SDD:  
~ronald.burton/submit\_w3\_at\_home <ENTER>**

and follow the instructions.