Part B – Foundations

Classes and Privacy

Workshop 3 (10 marks – 3% of your final grade)

In this workshop, you are to code a class definition with private data members and public member functions and implement the logic for those functions.

Learning Outcomes

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

- to define a class type
- to privatize data within the class type
- to instantiate an object of class type
- to access data within an object of class type through public member functions
- to use standard library facilities to format data inserted into the output stream
- to describe to your instructor what you have learned in completing this workshop

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 period. If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the in-lab portion after the period. If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (with a penalty; see below). The *at-home* portion of the lab is due on the day that is two days before your next scheduled workshop (23:59).

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible to back up your work regularly.

Late submission penalties:

- *In-lab* portion submitted late, with *at-home* portion: **0** for *in-lab*. Maximum of 7/10 for the entire workshop.
- If any of *in-lab*, *at-home* or *reflection* portions is missing, the mark for the workshop will be **0**/10.

IN-LAB (30%)

Design and code a module named `CRA_Account` that this workshop's application, listed below, can use. Store your class definition in a header file named `CRA_Account.h` and your function definitions in an implementation file named `CRA_Account.cpp`.

Include **compilation safeguards** in the header file and enclose all definitions within the `sict` namespace.

In the `CRA_Account.h` header file, predefine the following constants:

`max_name_length` with a value of 40. This value represents the maximum number of characters for any name of the account holder (not including the null byte `'\0'`).
`min_sin` with a value of 100000000. This is the smallest social insurance number.
`max_sin` with a value of 999999999. This is the largest social insurance number.

For this in-lab submission, your `CRA_Account` class considers a SIN valid if it is a 9-digit number between these predefined limits.

Create the `CRA_Account` class with private members that hold the following data:

- The family name on the account
- The given name on the account
- The social insurance number (SIN) on the account

Declare the following public member functions for your class:

`void set(const char* familyName, const char* givenName, int sin)`: checks if `sin` is a valid SIN. If so, this function stores the family and given names on the account along with the SIN. If `sin` is not a valid SIN, this function stores values that identify an empty state.

`bool isEmpty() const`: returns true if the object does not hold a valid SIN; false if the object holds a valid SIN.

`void display() const`: if the object is not empty, this function inserts into the standard output stream the content of the current instance in the following format:

```
Family Name: FAMILY_NAME<ENDL>
Given Name : GIVEN_NAME<ENDL>
CRA Account: SIN<ENDL>
```

If the object is empty, your function only inserts the following message:

```
Account object is empty!<ENDL>
```

The main program that uses your `CRA_Account` module contains the following code:

```
/*************************************************************
// Workshop 3: Classes and Privacy
// w3_in_lab.cpp
// Version 2.0
// 2017/09/09
// Chris Szalwinski
// Description
// This workshop demonstrates classes in C++ and
// how member variables are defined privately, but
// accessed through public member functions
//
// Revision History
////////////////////////////////////////////////////////////
// Name                   Date                       Reason
//
////////////////////////////////////////////////////////////
*************************************************************/

#include <iostream>
#include "CRA_Account.h"

using namespace std;
using namespace sict;

int main() {
    sict::CRA_Account myCRA;
    int sin;
    bool quit = false;
    char familyName[sict::max_name_length + 1];
```

```
    char givenName[sict::max_name_length + 1];

    cout << "Canada Revenue Agency Account App" << endl
         << "=================================" << endl << endl;
    cout << "Please enter your family name: ";
    cin >> familyName;
    cin.ignore();
    cout << "Please enter your given name: ";
    cin >> givenName;
    cin.ignore();

    do {
        cout << "Please enter your social insurance number (0 to quit): ";
        cin >> sin;
        cin.ignore();
        if (sin != 0) {
            myCRA.set(familyName, givenName, sin);
            if (myCRA.isEmpty())
                cout << "Invalid input! Try again." << endl;
            else
                quit = true;
        }
        else
            quit = true;
    } while (!quit);

    cout << "---------------------------------------" << endl;
    cout << "Testing the display function" << endl;
    cout << "---------------------------------------" << endl;
    myCRA.display();
    cout << "---------------------------------------" << endl;

    return 0;
}
```

Output Sample:

```
Canada Revenue Agency Account App
=================================

Please enter your family name: Doe
Please enter your given name: Jane
Please enter your social insurance number (0 to quit): 234
Invalid input! Try again.
Please enter your social insurance number (0 to quit): 1234567890
Invalid input! Try again.
Please enter your social insurance number (0 to quit): 193456787
---------------------------------------
Testing the display function
---------------------------------------
Family Name: Doe
Given Name : Jane
CRA Account: 193456787
---------------------------------------
```

IN-LAB SUBMISSION

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

Upload `CRA_Account.h`, `CRA_Account.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: (use your professor's Seneca `userid` to replace `profname.proflastname`)

`~profname.proflastname/submit 244_w3_lab``<ENTER>`

and follow the instructions.

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

AT-HOME (30%)

The *at-home* section of this workshop upgrades your `CRA_Account` module to track the balance owing or refund due on the account over several years and to improve the validation. Copy the original module to your at-home directory and do the following:

In the `CRA_Account.h` header file, predefine the following constant:

`max_yrs` with a value of 4. This value represents the maximum number of tax years that the object can store.

Add to the `CRA_Account` class the following attributes:

a) An array of size `max_yrs` representing the tax return years.
b) An array of size `max_yrs` representing balance owed or refund due on the account for each year.

c) The number of years for which tax return data is stored

Add a declaration for the following public member function to your class definition:

`void set(int year, double balance)`: if the object has a valid SIN and has room to store tax return data, this function stores `year` and `balance` in the object. If not, this function does nothing.

Modify the definitions of the following public member functions:

`void set(const char* familyName, const char* givenName, int sin)`: check if the data received is valid: that is, if `sin` is valid and if the family and given names are not empty. If the data received is valid, this function stores the family and given names on the account along with the SIN *and initializes the number of years for which data is stored to 0*. If any part of the data received is invalid, this function stores values that identify an empty state. For this submission, a SIN is considered valid if it is a 9-digit number between the predefined limits and has digits that satisfy the following rule:

A Canadian Social Insurance Number (SIN) has nine digits. The rightmost digit is a check digit that validates the other digits. For an integer to be a valid SIN, it must contain nine digits and the weighted sum of all digits including the check digit must be divisible by 10.

To obtain the weighted sum, take the digit to the left of the check digit and then every second digit leftwards (as shown below). Add each of these digits to itself. Then, add each digit of each sum to form the weighted sum of the even positioned digits. Add each of the remaining SIN digits (except the check digit) to form the sum of the odd positioned digits. Add the two sums and subtract the next highest number ending in zero from their total. If this number is the check digit, the whole number is a valid SIN; otherwise, the whole number is not a valid SIN.

For example:

```
SIN  193 456 787
               |
check digit is 7
add first set of alternates to themselves
   9  4  6  8
   9  4  6  8
  18  8 12 16
add the digits of each sum 1+8+8+1+2+1+6 = 27
add the other alternates   1+3+5+7       = 16
total                                    = 43
Next highest integer multiple of 10      = 50
Difference                               =  7
 Matches the check digit, therefore this number is a valid SIN
```

`void display() const`: if the object is not empty, this function inserts into the standard output stream the contents of the object in the following format:

- balance owing is > $2

```
Family Name: FAMILY_NAME<ENDL>
Given Name : GIVEN_NAME<ENDL>
CRA Account: SIN<ENDL>
Year (YEAR) balance owing: AMOUNT<ENDL>
```

- refund due is > $2

```
Family Name: FAMILY_NAME<ENDL>
Given Name : GIVEN_NAME<ENDL>
CRA Account: SIN<ENDL>
Year (YEAR) refund due: AMOUNT<ENDL>
```

- all other cases

```
Family Name: FAMILY_NAME<ENDL>
Given Name : GIVEN_NAME<ENDL>
CRA Account: SIN<ENDL>
Year (YEAR) No balance owing or refund due! <ENDL>
```

If the object is empty, your function inserts the following message:

```
Account object is empty!<ENDL>
```

Make sure that the printed amount has exactly two digits.

The main program that uses your upgraded `CRA_Account` module contains the following code:

```
/*********************************************************
// Workshop 3: Classes and Privacy
// w3_at_home.cpp
// Version 2.0
// 2017/09/09
// Chris Szalwinski
// Description
// This workshop demonstrates classes in C++ and
// how member variables are defined privately, but
// accessed through public member functions
//
// Revision History
/////////////////////////////////////////////////////////
// Name                    Date                    Reason
//
/////////////////////////////////////////////////////////
*********************************************************/

#include <iostream>
#include "CRA_Account.h"
using namespace std;
using namespace sict;

int main() {
    CRA_Account myCRA;
    int sin;
    bool quit = false;
    char familyName[max_name_length + 1];
    char givenName[max_name_length + 1];

    cout << "Canada Revenue Agency Account App" << endl
         << "=================================" << endl << endl;

    do {
        cout << "Please enter your family name: ";
        cin.getline(familyName, max_name_length);
        cout << "Please enter your given name: ";
        cin.getline(givenName, max_name_length);
        cout << "Please enter your social insurance number (0 to quit): ";
        cin >> sin;
        cin.ignore();
        if (sin != 0) {
            myCRA.set(familyName, givenName, sin);
            if (myCRA.isEmpty()) {
                cout << "Invalid input! Try again." << endl;
            }
            else {
                int year;
                double balance;
                do {
                    cout << "Please enter the year of your tax return (0 to quit): ";
                    cin >> year;
                    cin.ignore();
                    if (year != 0) {
                        cout << "Please enter the balance owing on your tax return (0
to quit): ";
```

```
                        cin >> balance;
                        cin.ignore();
                        myCRA.set(year, balance);
                    }
                } while (year != 0);
                quit = true;
            }
        }
        else {
            quit = true;
        }
    } while (!quit);
    cout << "----------------------------------------" << endl;
    cout << "Testing the display function" << endl;
    cout << "----------------------------------------" << endl;
    myCRA.display();
    cout << "----------------------------------------" << endl;

    return 0;
}
```

Output Sample:

```
Canada Revenue Agency Account App
=================================

Please enter your family name: Doe
Please enter your given name:
Please enter your social insurance number (0 to quit): 193456787
Invalid input! Try again.
Please enter your family name: Doe
Please enter your given name: Jane
Please enter your social insurance number (0 to quit): 123456789
Invalid input! Try again.
Please enter your family name: Doe
Please enter your given name: Jane
Please enter your social insurance number (0 to quit): 193456787
Please enter the year of your tax return (0 to quit): 2014
Please enter the balance owing on your tax return (0 to quit): 34.56
Please enter the year of your tax return (0 to quit): 2015
Please enter the balance owing on your tax return (0 to quit): -1234
Please enter the year of your tax return (0 to quit): 2016
Please enter the balance owing on your tax return (0 to quit): 1.23
Please enter the year of your tax return (0 to quit): 2013
Please enter the balance owing on your tax return (0 to quit): -0.12
Please enter the year of your tax return (0 to quit): 0
----------------------------------------
Testing the display function
----------------------------------------
Family Name: Doe
Given Name : Jane
CRA Account: 193456787
Year (2014) balance owing: 34.56
Year (2015) refund due: 1234.00
```

```
Year (2016) No balance owing or refund due!
Year (2013) No balance owing or refund due!
---------------------------------------
```

AT-HOME REFLECTION (40%)

Create a file named `reflect.txt` that contains answers to the following questions:

1) What is the size of each C-style character string that stores a family or given name for the account? Why?
2) Your class declares two member functions named `set(...)`. In C, this would generate an error. Explain why C++ allows this.
3) What have you learned during this workshop?

QUIZ REFLECTION

Add a section to `reflect.txt` called "**Quiz X Reflection:**" (replace the '**X**' with the number of the last quiz).

Identify all of the questions from the quiz that you did not answer correctly. Under each question, specify the correct answer. If you missed the last quiz, enter all of the questions and the correct answer for each.

AT-HOME SUBMISSION

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

If not on matrix already, upload `reflect.txt`, `CRA_Account.h`, `CRA_Account.cpp` 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: (use your professor's Seneca `userid` to replace `profname.proflastname`)

~profname.proflastname/submit 244_w3_home<ENTER>

and follow the instructions.

IMPORTANT: Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.