```
/**************************************************************************
 * AUTHOR       : JOSHUA SALZEDO, Chris Burrell
 * LAB #8       : Array Train - Set # 1
 * CLASS        : CS1A
 * SECTION      : MW: 8AM
 * DUE DATE     : 04/23/2018
 *************************************************************************/
#include <iostream>
#include <iomanip>
#include "MyHeader.h"
#include "functions.h"

/**************************************************************************
 *  Array Train - Set # 1
 * ------------------------------------------------------------------------
 * This program outputs min, max, avg, and sum of two parallel arrays
 *  and outputs selected elements from user input
 * ------------------------------------------------------------------------
 * INPUT:
 *  Age to saerch by
 *  name so earch by
 *
 * OUTPUT:
 *  Average age , minimum age, maximum age, sum of ages,
 *  namees and ages from user input
 *************************************************************************/

int main()
{
/**************************************************************************
 * CONSTANTS
 * ------------------------------------------------------------------------
 * OUTPUT - USED FOR CLASS HEADING
 * ------------------------------------------------------------------------
 * PROGRAMMER       :   Programmer's Name
 * CLASS            :   students course
 * SECTION          :   Class Days and Times
 * LAB_NUM          :   Lab Number
 * LAB_NAME         :   Title of the lab
 * ------------------------------------------------------------------------
 * PROCESSING - USED FOR PROGRAM EXECUTION
 * ------------------------------------------------------------------------
 * NAMES            :   hardcoded input names (parallel array)
 * AGES             :   hardcoded input ages (parallel array)
 * PROMPT_FIND_INT  :   Prompt for an age input
 * PROMPT_NAME      :   Prompt for a name input
 * ***********************************************************************/
    const char PROGRAMMER[] = "Joshua Salzedo, Chris Burrell";
    const char CLASS[] = "CS1A";
    const char SECTION[] = "MW: 8:00a - 12:00p";
    const int LAB_NUM = 25;
    const char LAB_NAME[] = "Array Train # 1";

    const int AR_LEN = 11;

    const string NAMES[] = {
            "zac",
            "Kasra",
            "Bas",
            "Sara",
            "Nick",
            "Delvin",
            "Justin",
            "Abe",
            "Jeremy",
            "Farah",
            "Maryan",
    };
    const int AGES[] = {
            22,
            75,
            19,
            21,
            18,
            12,
            19,
            5,
            62,
            21,
            21
    };
    const string PROMPT_FIND_INT = "Please enter an age you want to look "
                                   "for: ";
    const string PROMPT_NAME = "enter a name to find: ";


    double averageAges;             // CALC & OUT  - average value of AGES
    //                                  array
    int firstInstanceOfAgeIndex;    // CALC & OUT  - index for First position
    //                                  for input int
    int totalOccurrences;           // CALC & OUT  - total occurences of
    //                                  an input int
    int nameIndex;                  // CALC & OUT  - index for found name
    int sumAges;                    // CALC & OUT  - sum of AGES array
    int youngestPersonIndex;        // CALC & OUT  -index of the smallest age
    int oldestPersonIndex;          // CALC & OUT  - index of the biggest age
    int searchInt;                  // IN & OUT    - user provided input
    string searchString;            // IN & OUT    - user provided input
    string formattedString;         // CALC & OUT  - Formatted output strings
    string oldestPerson;            // CALC & OUT  - name of oldest person
    string youngestPerson;          // CALC & OUT  - name of youngest person

    // OUTPUT - class header
    DisplayHeader(PROGRAMMER, CLASS, SECTION, LAB_NUM, LAB_NAME);

    // compute the average of the AGES array
    averageAges = AverageIntArray(AGES, AR_LEN);

    // set up formatting
    cout << fixed << setprecision(2);
    cout << "Average value of ages array is : " << averageAges;
    // clean up after ourselves
    cout << scientific << setprecision(6) << endl << endl;

    // find an int to search for
    searchInt = SaneInputCinInt(0, 100, PROMPT_FIND_INT);

    // find the first instance of the searchInt
    firstInstanceOfAgeIndex = FindFirstInstance(AGES, AR_LEN, searchInt);

    // check if an instance was actually found
    if (firstInstanceOfAgeIndex != AR_LEN)
```

```cpp
    {
        formattedString = "you found ";
        formattedString += NAMES[firstInstanceOfAgeIndex];
        formattedString += " who is ";
        formattedString += to_string(AGES[firstInstanceOfAgeIndex]);
        formattedString += " years old at index # ";
        formattedString += to_string(firstInstanceOfAgeIndex) += '\n';

    }
    else
    {
        formattedString = "Name not found on the list.\n";
    }
    cout << formattedString << endl;

    // find total occurences of the age
    totalOccurrences = FindOccurrencesInt(AGES, AR_LEN, searchInt);
    // reinit
    formattedString = "";

    // switch over # occurences, for word choice and verb tense.
    switch (totalOccurrences)
    {
        case 0:
            formattedString = "There is noone with that age.\n";
            break;

        case 1:
            formattedString = "there is a grand total of one person with "
                              "that specific age.";
            break;

        default:
            formattedString = "There are a total of ";
            formattedString += to_string(totalOccurrences);
            formattedString += " people with that age.";
            // do something
            break;
    }

    cout << formattedString << endl;


    cout << PROMPT_NAME;
    getline(cin, searchString);

    nameIndex = FindString(NAMES, AR_LEN, searchString);
    formattedString = "";

    if (nameIndex == AR_LEN)
    {
        formattedString = "Im sorry, \"";
        formattedString += searchString += "\" was not found in my records.";
    }
    else
    {
        formattedString += NAMES[nameIndex];
        formattedString += "is";
        formattedString += to_string(AGES[nameIndex]) += " years old and "
                                                         "exists at index #";
        formattedString += to_string(nameIndex);
    }

    cout << formattedString << endl << endl;

    // fetch the index of the oldest person
    oldestPersonIndex = FindLargestInt(AGES, AR_LEN);
    // and assign their name to a value we can more easily use
    oldestPerson = NAMES[oldestPersonIndex];

    // fetch the index of the youngest person
    youngestPersonIndex = FindSmallestInt(AGES, AR_LEN);
    youngestPerson = NAMES[youngestPersonIndex];

    cout << "The oldest person is " << oldestPerson << " who is "
         << AGES[oldestPersonIndex] << " years old which exists at index #"
         << oldestPersonIndex << endl;

    cout << "The youngest person is " << youngestPerson << " who is "
         << AGES[youngestPersonIndex] << " years old which exists at index #"
         << oldestPersonIndex << endl;

    sumAges = SumIntArray(AGES, AR_LEN);
    cout << "Overall, the total combined age is " << sumAges << endl;

    cout << left << left;
    cout << "=======================" << endl;
    cout << "==-- Begin Part 3&4 --==" << endl;
    cout << "=======================" << endl;

    cout << " ----------= Test #1=----------" << endl;

    cout << fixed << setprecision(2);
    // compute average
    averageAges = AverageIntArray(AGES, AR_LEN);

    cout << "The average of the array is " << averageAges << endl;
    // restore defaults
    cout << scientific << setprecision(6);

    cout << " ----------= Test #2=----------" << endl;
    // loop over caluclating first instance of the array
    for (int i = 0; i < 4; ++i)
    {
        int myInt;
        int result;

        myInt = SaneInputCinInt(0, 100, PROMPT_FIND_INT);
        result = FindFirstInstance(AGES, AR_LEN, myInt);

        if (result == AR_LEN)
        {
            cout << myInt << " does not exist within the array." << endl;
        }
        else
        {
            cout << NAMES[result] << " is " << AGES[result] << " years old"
                 << "(index # " << result << ')' << endl;

        }
    }

    cout << endl;

    cout << " ----------= Test #3=----------" << endl;
    for (int i = 0; i < 4; ++i)
    {
        int instances;
        int input;

        input = SaneInputCinInt(0, 100, PROMPT_FIND_INT);
        instances = FindOccurrencesInt(AGES, AR_LEN, input);

        cout << input << " Occured " << instances << " time(s)." << endl;
    }

    cout << endl;
    cout << " ----------= Test #4=----------" << endl;
    // test for name search
    for (int i = 0; i < 4; ++i)
    {
        int resultIndex;
        string searchName;

        cout << PROMPT_NAME;
        // fetch a name
        getline(cin, searchName);

        // get the result
        resultIndex = FindString(NAMES, AR_LEN, searchName);

        if (resultIndex != AR_LEN)
        {
            // array hit
            cout << searchName << " is " << AGES[resultIndex] << " years old"
                 << "( index # " << resultIndex << ')';
        }
        else
        {
            // array miss
            cout << searchName << " does not exist within the array.";
        }
        cout << endl;

    }
    cout << " ----------= Test #5=----------" << endl << endl;
    {
        int youngestIndex;

        youngestIndex = FindSmallestInt(AGES, AR_LEN);

        cout << "The youngest person is " << NAMES[youngestIndex]
             << " who is ";
        cout << AGES[youngestIndex] << " years old (Index # "
             << youngestIndex << ')';
    }
    cout << endl;
    cout << " ----------= Test #6=----------" << endl << endl;
    {
        int oldestIndex;
        oldestIndex = FindLargestInt(AGES, AR_LEN);

        cout << "The oldest person is " << NAMES[oldestIndex]
```

```
             << " who is ";
        cout << AGES[oldestIndex] << " years old (Index # "
             << oldestIndex << ')';
    }
    cout << endl << " ----------= Test #7=----------" << endl << endl;
    {
        int mSumAges;

        mSumAges = SumIntArray(AGES, AR_LEN);
        cout << "Sum of all elements of the array is: " << mSumAges;
    }
    cout << endl;
    cout << "Done.";

    return 0;
}
```