Create a container class **people** that holds an array of 20 **person** objects. Write a program that uses the container class **people** to store the contents of an input. This class should use dynamic memory allocation and clean up memory when it is destroyed. Test your program with the code and data files below.

<https://markbowman.org/231/Program02.zip>

You may not change the Main.cpp file.

The **people** class should have the following data element:

* Map - a pointer to an array of **person** objects.
* Len - the number of objects in use.

The **people** class should have the following functions:

* Null constructor - create storage for array using **new**.
* Destructor - delete array storage using **[ ]** **delete**.
* Insert - add a person to the array, in ascending order.
* Display - put the entire array to an output stream, and show the number of **person** items, and the average age.
* Find - given an id, locate a matching person in the list. Return true if found, false if not.
* Remove - given an id, remove a matching person from the list.

You will need to update your **person** class to add the following:

* ID - a private data element
* **==** compare the **person** to an integer, using the **id** data.

When inserting an item to the array, don’t just append it to the end, and then sort the list. This works, but is very inefficient. Instead, determine where the item should go, and shift the rest of the array to make room.

What to Hand In

Hand in a copy of the code, and runs to demonstrate that your program and functions work with both data files. Sample output and formatting is shown on the next page.

**Sample Runs:**

**Family.txt:**

Text

Description automatically generated

**Friends.txt:**

Text

Description automatically generated

**Final Code:**

Main.cpp (not-edited):

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Program 02

\* Written by Mark M Bowman

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

#include "People.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Main

\* Test function - DO NOT CHANGE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void main()

{ int i;

int list[4] = { 4732,1132,7437,6706 };

string fname;

person temp;

people a;

fstream infile;

// Get file name

cout << "Enter data file name: ";

cin >> fname;

cout << endl;

// Read from file

infile.open(fname,ios::in);

while (temp.get(infile))

{

a.insert(temp);

}

infile.close();

// Display list

cout << "Initial List" << endl << endl;

a.display(cout);

// Test find

for(i=0;i<4;i++)

{

cout << "Find " << list[i] << " ";

if (a.find(list[i]))

{

cout << "Found";

}

else

{

cout << "Not Found";

}

cout << endl;

};

cout << endl;

// Test remove

for (i = 0; i < 4; i++)

{

a.remove(list[i]);

}

// Display updated list

cout << "Updated List" << endl << endl;

a.display(cout);

}

Person.h (Person class):

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Person Class Declaration

\* Created by Safford, Twymun

\* Original Date: 08-Sep-2021

\* Update Date: 03-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//include guards - person.h included multiple times within the file

#ifndef PERSON\_H

#define PERSON\_H

#include <iostream>

#include <string>

using namespace std;

// change the name of class to person

class person

{

public:

person();

bool get(istream& in);

void put(ostream& out);

bool operator<(const person& t);

bool operator>(const person& t);

bool operator==(int ID); // compare person's ID

int getAge(); // function to return the age

private:

string firstName, lastName;

int age;

int ID; // add ID

void capitalize();

};

#endif

Note: For the above, I got my code to compile but only by using the include guards.

Person.cpp (Person class):

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Person Class Definitions

\* Created by Safford, Twymun

\* Date: 08-Sep-2021

\* Update Date: 03-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "Person.h"

#include <iostream>

#include <iomanip>

#include <string>

using namespace std;

#include <cctype>

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Null constructor

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

person::person() :firstName(""), lastName(""), age()

{

//null constructor - empty

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* get() - Read first name, last name,

\* and age from I/O stream (updated to

\* use Boolean versus the void used

\* in Program 01)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// function to read first name, last name, and age from an input stream

//void person::get(istream &in)

bool person::get(istream& in)

{

if (in.eof())

{

return false;

}

in >> ID >> firstName >> lastName >> age;

capitalize(); // helper function to capitalize the first and last name

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Capitalize - function to capitalize

\* first and last names.

\* Updated 04-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// helper function to capitalize the first and last names

void person::capitalize()

{

//unsigned integer size\_t

size\_t i;

//temp - used to temporarily hold first/last name before capitalizing

string temp = "";

// capitalize the first name i.e first letter will be capital and rest small

for (i = 0; i < firstName.length(); i++)

{

if (i == 0)

temp += toupper(firstName[i]);

else

temp += tolower(firstName[i]);

}

//update their first name

firstName = temp;

// capitalize the last name i.e first letter will be capital and rest small

temp = "";

for (i = 0; i < lastName.length(); i++)

{

if (i == 0)

temp += toupper(lastName[i]);

else

temp += tolower(lastName[i]);

}

lastName = temp; // update lastName

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* put() - writes last name, first name,

\* and age to an output stream (to screen

\* in this case)

\* \*Updated 04-Oct-2021 for spacing

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// function to write last name, first name, and age to an output stream

void person::put(ostream& out)

{

std::cout << std::left << std::setw(10) << ID << std::left << std::setw(12)

<< lastName << " " << std::left << std::setw(12) << firstName << std::left

<< std::setw(12) << age << endl;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Comparison Operators : Implementation/Solution

\* Safford, Twymun - 08-Sep-2021

\*

\* >, <, and == - to compare the person

\* to another person: uses last name, then

\* first name, and then age - returns

\* boolean

\*

\* i.e.:

\*

\* Mark Bowman 20

\* Frank Bowman 40

\* Frank Bowman 16

\*

\* would be:

\*

\* Bowman Frank 16

\* Bowman Frank 40

\* Bowman Mark 20

\*

\* Updated for Program 2: 04-Oct-2021

\* to compare IDs now

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Operator : <

\* Safford, Twymun - 08-Sep-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// function to compare this trial with t and return true if this trial < p else return false

bool person::operator<(const person& p)

{

// trials are compared using last name, first name and then age

return (lastName < p.lastName) || (lastName == p.lastName && firstName < p.firstName)

|| (lastName == p.lastName && firstName == p.firstName && age < p.age);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Operator : >

\* Safford, Twymun - 08-Sep-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// function to compare this trial with t and return true if this trial > p else return false

bool person:: operator>(const person& p)

{

return (lastName > p.lastName) || (lastName == p.lastName && firstName > p.firstName)

|| (lastName == p.lastName && firstName == p.firstName && age > p.age);

}

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* Operator : ==

// \* Safford, Twymun - 08-Sep-2021

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//

//// function to compare this trial with t and return true if this trial = p else return false

//bool person:: operator==(const person& p)

//{

// return (lastName == p.lastName && firstName == p.firstName && age == p.age);

//}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Implementation - Operator: == (updated)

\* the individual with the int ID

\* Implemented: 04-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool person:: operator==(int ID)

{

return (this->ID == ID);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function to return the age of

the individual

\*Implemented: 04-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int person::getAge()

{

return age;

}

Container Class Starts Here:

People.h (People class):

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* People Class Declaration

\* for container creation

\* Created by Safford, Twymun

\* Date: 03-Oct-2021

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//include guards

#ifndef PEOPLE\_H

#define PEOPLE\_H

#include "Person.h"

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

class people

{

private:

person\* Map;

int Len;

public:

people();

~people();

void insert(person p);

void display(ostream& out);

bool find(int ID);

void remove(int ID);

};

#endif

People.cpp:

#include "People.h"

#include "Person.h"

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

// default constructor

people::people()

{

// create Map to be an array of size 20

//inside of the class header 'People.h'

Map = new person[20];

//inside of the class header 'People.h'

Len = 0; // set Len to 0 since it is empty

}

// destructor

people::~people()

{

delete[] Map;

}

// function to insert p in sorted order

void people::insert(person p)

{

int i, j;

if (Len < 20) // check array has space

{

// loop to find the index of insertion

for (i = 0; i < Len; i++)

{

if (Map[i] > p) // ith element > p

{

// loop to shift the elements right by 1 position from index i to end

for (j = Len; j > i; j--)

{

Map[j] = Map[j - 1];

}

// insert p at index i

Map[i] = p;

Len++; // increment number of person - up to max size of 20

return;

//break;

}

}

//// insert p at index i

Map[i] = p;

Len++; // increment number of person - up to max size of 20

}

}

// function to display the details of each person and display average age

void people::display(ostream& out)

{

int totalAge = 0, avgAge = 0;

std::cout << std::left << std::setw(10) << "ID" << std::left << std::setw(12) << "Last" << " "

<< std::left << std::setw(12) << "First" << std::left << std::setw(12) << "Age" << endl;

std::cout << std::left << std::setw(10) << "------" << std::left << std::setw(12) << "-------"

<< " " << std::left << std::setw(12)

<< "------" << std::left << std::setw(12) << "------" << endl;

// loop to display the details of each person and calculate total age of all persons

int i;

for (i = 0; i < Len; i++)

{

Map[i].put(out);

totalAge += Map[i].getAge();

}

std::cout << std::left << std::setw(10) << "------" << std::left << std::setw(12) << "-------"

<< " " << std::left << std::setw(12)

<< "------" << std::left << std::setw(12) << "------" << endl;

// calculate average age

if (Len > 0)

avgAge = totalAge / Len;

// display total number of persons and average age

std::cout << std::left << std::setw(10) << "" << std::left << std::setw(3) << Len

<< std::left << std::setw(9) << "Persons" << std::left << std::setw(12) << "Average"

<< std::left << std::setw(12) << avgAge << endl;

}

// function to return true if person with ID exists else return false

bool people::find(int ID)

{

// loop over array

int i;

for (i = 0; i < Len; i++)

{

if (Map[i] == ID) // person found

return true;

}

return false; // person not found

}

// function to remove the person with ID

void people::remove(int ID)

{

int i;

// loop over the array

for (i = 0; i < Len; i++)

{

if (Map[i] == ID) // ID found

{

// loop to shift the elements 1 position to left from index i+1 to end

for (int j = i; j < Len - 1; j++)

Map[j] = Map[j + 1];

Len--; // decrement the number of person

break;

}

}

}

Enter data file name: ***Family.txt***

Initial List

ID Last First Age

---- -------------- -------- ---

4732 Bowman David 45

9440 Bowman Frank 37

2635 Bowman John 30

7166 Bowman Mark 13

1132 Bowman Mark 42

3333 Bowman Richard 47

2487 Christensen Ann 70

9636 Cox Susan 36

7390 Gueller Kathleen 34

6706 Morales Carlos 68

---- -------------- -------- ---

10 Persons Average 42

Find 4732 Found

Find 1132 Found

Find 7437 Not Found

Find 6706 Found

Updated List

ID Last First Age

---- -------------- -------- ---

9440 Bowman Frank 37

2635 Bowman John 30

7166 Bowman Mark 13

3333 Bowman Richard 47

2487 Christensen Ann 70

9636 Cox Susan 36

7390 Gueller Kathleen 34

---- -------------- -------- ---

7 Persons Average 38

Enter data file name: ***Friends.txt***

Initial List

ID Last First Age

---- -------------- -------- ---

1132 Bowman Mark 56

6798 Gleason Kent 62

1163 Hall Josh 32

8546 Haney Robert 33

8610 Smith Gary 28

1147 Taylor Brian 36

---- -------------- -------- ---

6 Persons Average 41

Find 4732 Not Found

Find 1132 Found

Find 7437 Not Found

Find 6706 Not Found

Updated List

ID Last First Age

---- -------------- -------- ---

6798 Gleason Kent 62

1163 Hall Josh 32

8546 Haney Robert 33

8610 Smith Gary 28

1147 Taylor Brian 36

---- -------------- -------- ---

5 Persons Average 38