Part B – Foundations

Dynamic Memory

Workshop 2

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:

- to allocate and deallocate dynamic memory for an array of elements;
- to overload a global function;
- to explain the difference between static and dynamic memory;
- to describe to your instructor what you have learned in completing this workshop.

Submission Policy

The *"in-lab"* portion 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"* portion 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 your next scheduled workshop**.

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

You are responsible to regularly backup your work.

In-Lab (50%):

Design and code a class named `Kingdom` in the namespace `westeros`. The class should have two public members:

`m_name`: a statically allocated array of characters of size 32 (including `'\0'`) that holds the name of the kingdom;

`m_population`: an integer that stores the number of people living in the kingdom.

Add to the `westeros` namespace, a function a function called `display(...)` that returns nothing, receives as a parameter a **reference** to an object of type `Kingdom` and prints to the screen the parameter in the following format:

```
KINGDOM_NAME, population POPULATION<ENDL>
```

Put the class **definition** and the `westeros::display(...)` **declaration** in a header named `kingdom.h`. Put the implementation of `westeros::display(...)` in a file named `kingdom.cpp`.

Complete the implementation of the `w2_in_lab.cpp` main module shown below (see the parts marked with `TODO`). Below the source code is the expected output from your program (with red color is what you should type as input for your program). The output of your program should match **exactly** the sample output shown below.

```
#include <iostream>
#include "kingdom.h"

using namespace std;
using namespace westeros;

int main(void)
{
    int count = 0; // the number of kingdoms in the array

    // TODO: declare the pKingdoms pointer here (don't forget to initialize it to null)

    cout << "==========" << endl
        << "Input data" << endl
        << "==========" << endl
        << "Enter the number of kingdoms: ";
    cin >> count;
    cin.ignore();

    // TODO: allocate dynamic memory here for the pKingdoms pointer

    for (int i = 0; i < count; ++i)
    {
        // TODO: add code to accept user input for the pKingdoms array
    }
    cout << "==========" << endl << endl;
```

```
    // testing that "display(...)" works
    cout << "------------------------------" << endl
        << "The first kingdom of Westeros" << endl
        << "------------------------------" << endl;
    display(pKingdoms[0]);
    cout << "------------------------------" << endl << endl;

    // TODO: deallocate the dynamic memory here

    return 0;
}
```

Output Sample:

```
==========
Input data
==========
Enter the number of kingdoms: 2
Enter the name for kingdom #1: The_Vale
Enter the number people living in The_Vale: 234567
Enter the name for kingdom #2: The_Reach
Enter the number people living in The_Reach: 567890
==========

------------------------------
The first kingdom of Westeros
------------------------------
The_Vale, population 234567
------------------------------
```

IN-LAB SUBMISSION:

To submit the *in-lab* section, demonstrate execution of your program with the exact output as example above. Upload `kingdom.h`, `kingdom.cpp` and `w2_in_lab.cpp` to your matrix account. Compile and run your code and make sure everything works properly. To submit, run the following script from your account (and follow the instructions):

~profname.proflastname/submit 200_w2_lab <ENTER>

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 (40%):

Overload the function `westeros::display(...)` by adding 3 more methods:

Overload #1: a function that returns nothing and receives two parameters: the *first* one is an array of `Kingdom`s, and the *second* one is an integer representing the number of elements in the array. This function should calculate the total number of people living in Westeros and print the array to the screen in the following format:

```
------------------------------<ENDL>
Kingdoms of Westeros<ENDL>
------------------------------<ENDL>
1. KINGDOM_NAME, population POPULATION<ENDL>
2. KINGDOM_NAME, population POPULATION<ENDL>
3. KINGDOM_NAME, population POPULATION<ENDL>
------------------------------<ENDL>
Total population of Westeros: TOTAL_POPULATION<ENDL>
------------------------------<ENDL>
```

Overload #2: a function that returns nothing and receives **three** parameters: the *first* one is an array of `Kingdom`s, the *second* one is the number of elements in the array, and the *third* one is an integer representing the minimum number of people that a kingdom should have in order to be printed on the screen. This function should print to the screen only the kingdoms that have a population bigger or equal to the value specified in the third parameter, in the following format:

```
------------------------------<ENDL>
Kingdoms of Westeros with more than MIN_POPULATION people<ENDL>
------------------------------<ENDL>
KINGDOM_NAME, population POPULATION<ENDL>
KINGDOM_NAME, population POPULATION<ENDL>
------------------------------<ENDL>
```

Overload #3: a function that returns nothing and receives **three** parameters: the *first* one is an array of `Kingdom`s, the *second* one is the number of elements in the array, and the *third* one is the name of a kingdom. If in the array exists a Kingdom with the name specified in the third parameter, this function should print it to the screen in the following format:

```
------------------------------<ENDL>
Searching for kingdom KINGDOM_NAME in Westeros<ENDL>
------------------------------<ENDL>
KINGDOM_NAME, population POPULATION<ENDL>
------------------------------<ENDL>
```

If in the array there is no kingdom with the specified name, this function should print to the screen a message in the following format:

```
------------------------------<ENDL>
Searching for kingdom KINGDOM_NAME in Westeros<ENDL>
------------------------------<ENDL>
KINGDOM_NAME is not part of Westeros.<ENDL>
------------------------------<ENDL>
```

NOTE: all overloads must be part of the `westeros` namespace, have a declaration in `kingdom.h` and an implementation in `kingdom.cpp`.

NOTE: all overloads must call the `westeros::display(...)` function you created for the *in-lab* part in order to display the name and the population of the kingdom.

Complete the implementation of the `w2_at_home.cpp` main module shown below (see the parts marked with **TODO**, reuse the parts that you have done for the *in-lab* part). Below the source code is the expected output from your program (with red color is what you should type as input for your program). The output of your program should match **exactly** the sample output shown below.

```
#include <iostream>
#include "kingdom.h"

using namespace std;
using namespace westeros;

int main(void)
{
    int count = 0; // the number of kingdoms in the array

    // TODO: declare the pKingdoms pointer here (don't forget to initialize it)

    cout << "==========" << endl
        << "Input data" << endl
        << "==========" << endl
        << "Enter the number of kingdoms: ";
    cin >> count;
    cin.ignore();

    // TODO: allocate dynamic memory here for the pKingdoms pointer

    for (int i = 0; i < count; ++i)
    {
        // TODO: add code to accept user input for the pKingdoms array
    }
    cout << "==========" << endl << endl;


    // testing that "display(...)" works
```

```
    cout << "------------------------------" << endl
        << "The first kingdom of Westeros" << endl
        << "------------------------------" << endl;
    display(pKingdoms[0]);
    cout << "------------------------------" << endl << endl;


    // testing that the first overload of "display(...)" works
    display(pKingdoms, count);
    cout << endl;


    // testing that the second overload of "display(...)" works
    display(pKingdoms, count, 345678);
    cout << endl;


    // testing that the third overload of "display(...)" works
    display(pKingdoms, count, "Mordor");
    cout << endl;

    display(pKingdoms, count, "The_Vale");
    cout << endl;

    // TODO: deallocate the dynamic memory here

    return 0;
}
```

Output Sample:

```
==========
Input data
==========
Enter the number of kingdoms: 5
Enter the name for kingdom #1: The_Riverlands
Enter the number people living in The_Riverlands: 123456
Enter the name for kingdom #2: The_Vale
Enter the number people living in The_Vale: 234567
Enter the name for kingdom #3: The_Westernlands
Enter the number people living in The_Westernlands: 345678
Enter the name for kingdom #4: The_Stormlands
Enter the number people living in The_Stormlands: 456789
Enter the name for kingdom #5: The_Reach
Enter the number people living in The_Reach: 567890
==========

------------------------------
The first kingdom of Westeros
------------------------------
The_Riverlands, population 123456
------------------------------

------------------------------
Kingdoms of Westeros
```

```
------------------------------
1. The_Riverlands, population 123456
2. The_Vale, population 234567
3. The_Westernlands, population 345678
4. The_Stormlands, population 456789
5. The_Reach, population 567890
------------------------------
Total population of Westeros: 1728380
------------------------------

------------------------------
Kingdoms of Westeros with more than 345678 people
------------------------------
The_Westernlands, population 345678
The_Stormlands, population 456789
The_Reach, population 567890
------------------------------

------------------------------
Searching for kingdom Mordor in Westeros
------------------------------
Mordor is not part of Westeros.
------------------------------


------------------------------
Searching for kingdom The_Vale in Westeros
------------------------------
The_Vale, population 234567
------------------------------
```

AT-HOME REFLECTION (10%):

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

1) What happens to dynamic memory if it is not deallocated?
2) What is the difference between dynamic and static memory? Give examples from you code where have you used dynamic memory and where have you used static memory.
3) Explain what have you learned in this workshop

AT-HOME SUBMISSION:

To submit the *at-home* section, demonstrate execution of your program with the exact output as example above. Upload `reflect.txt`, `kingdom.h`, `kingdom.cpp` and `w2_at_home.cpp` to your matrix account. Compile and run your code and make sure everything works properly. To submit, run the following script from your account (and follow the instructions):

```
~profname.proflastname/submit 200_w2_home<ENTER>
```

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.