CST 8219 – F18 - Assignment #2

RasterGraphic in C++ using Container Classes and Overloaded Operators

Due Time: 23.59, Sat 10 November 2017 **Earnings:** 9% of your final grade

NOTE: Plan to finish a few days early to avoid last minute hardware/software or other unexpected holdups, for which no allowance is given.
NOTE: The code in this assignment must be your own work. It must not be code taken from another student or written for you by someone else, even if you give a reference to the person you got it from (attribution); if it is not entirely your own work it will be treated as plagiarism and given a fail mark, or less.

Purpose: This is a development of assignment 1 with the addition of a two ready-made container class templates and overloaded operators. It works in a very similar way. It is a console application that holds the GraphicElements of an RasterGraphic application (there is no actual graphics in the assignment) using a `forward_list` class template of unspecified length in dynamic memory. The `forward_list` is a container class that supports fast insertion and removal of elements and is implemented as a singly-linked list just as you have used in assignments 0 and 1.
In addition a GraphicElement now has set of Image objects associated with it. You can think of them as the separate graphical components of a GraphicElement that combine the make the complete image (– they might be green/blue-screen overlays as in chroma keying). A Image object holds the pixel location that it is to be overlayed in the GraphicElement and also the duration for which it should be Imaged. The set of Image objects is held in a `vector` , another common container class template, more flexible but not (for some applications) as fast the forward list which is designed to be minimal. In addition the `string` class is used to hold strings. A `string` is essentially a vector of `char`s.
Part of the code is shown on the next page; it is also on Brightspace in a text file that you can copy and paste. You **MUST** use this code **without modification (not a single character changed): no code added or removed, no new global variables or functions, no new classes, no macros, no defines and no statics**. Your task is to implement, using C++, only the RasterGraphic, GraphicElement and Image class member functions and the global insertion operators and not add any new ones. Everything you write and submit is in the files: RasterGraphic.cpp, GraphicElement.cpp and Image.cpp.

The RasterGraphic is a series of GraphicElements held in a `forward_list`. Each GraphicElement holds its list of Image objects in a `vector`. An Image object contains its Image time which is set by the user - therefore the fixed GraphicElement Image time of 1 second that was used in previous assignments is gone and is replaced by the individual Image times that the user specifies. You can:

- Add a new GraphicElement to the RasterGraphic at a position in the forward list selected by the user
- Delete the first GraphicElement in the RasterGraphic
- Run the RasterGraphic to show the list of Image details of each GraphicElement one after another at the Image intervals specified by the user when the Image was entered – note that the output counts up the seconds
- concatenate two GraphicElements to make a single GraphicElement with combined Images that is automatically added to the start of the `forward_list`
- Quit

An example of the output of the running application is given at the end. Yours must work identically and produce identical output.
Note the following:

- dynamic memory management is done with new and delete
- there is no unused dynamic memory at any time
- input/output is done with cin and cout
- string objects are used in the RasterGraphic and GraphicElement classes to hold names but a c-style NULL-terminated char array is still used in the Image class
- Release of dynamically allocated memory is done in destructors so there is no resource leak (or you lose 30%).

See the Marking Sheet for how you can lose marks, but you will lose at least 60% if:
1. you change the supplied code in any way at all (not a single character) - no code added or removed, no macros, no defines, no statics and no additional classes, global functions or variables,
2. it fails to build in Visual Studio 2015,
3. It crashes in normal operation,
4. it doesn't produce the example output.

Part of the code is shown on the next page. You MUST use this code **without modification.** Your task is to add the implementation of the class member functions.

What to Submit : Use Brightspace to submit this assignment as a plain zip file (**not** RAR or 7-Zip or 9 Zip) containing only RasterGraphic.cpp, GraphicElement.cpp and Image.cpp. The name of the zipped folder **must** contain your name as a prefix so that I can identify it, for example using my name the file would be tyleraAss2CST8219.zip. It is also vital that you include the Cover Information (as specified in the Submission Standard) as a file header in your source file so the file can be identified as yours. Use comment lines in the file to include the header.

Before you submit the code,

- check that it builds and executes in Visual Studio 2015 as you expect - if it doesn't build for me, for whatever reason, you get a deduction of at least 60%.
- make sure you have submitted the correct file – if I cannot build it because the file is wrong or missing from the zip, even if it's an honest mistake, you get 0.

There is a late penalty of 25% per day. Don't send me files as an email attachment – they will get 0.

Supplied code (also in a text file you can copy and paste on BlackBoard). Don't change it.

```
// Image.h
#pragma once

class Image
{
          int pixel_x;
          int pixel_y;
          int duration;
          char* name;
public:
          Image(int x, int y, int duration, char* name);
          Image(const Image&);
          ~Image();
          friend ostream& operator<<(ostream&, Image&);
};
```

```
// GraphicElement.h
#pragma once

class GraphicElement
{
          string fileName;
          vector<Image> Images;
public:
          GraphicElement::GraphicElement(string s, vector<Image> d) :fileName(s), Images(d){}
          GraphicElement operator+(GraphicElement&);
          friend ostream& operator<<(ostream&, GraphicElement&);
};
```

```
//RasterGraphic.h
#pragma once

class RasterGraphic
{
          string name;
          forward_list<GraphicElement> GraphicElements;
public:
          RasterGraphic(string s): name(s){}
          void InsertGraphicElement();
          void DeleteGraphicElement();
          void Concatenate() // inline
          {
                    cout << "Concatenating two GraphicElements" << endl;
                    int index1 = -1, index2 = -1;
                    RasterGraphic& A = *this;
                    int count = distance(GraphicElements.begin(),GraphicElements.end());
                    do {
                              cout << "Please enter valid indexes of the two GraphicElements to concatenate (0 to " << count -
1 << ")" << endl;
                              cin >> index1 >> index2;
                    } while ((index1<0 || index1>count - 1) || (index2<0 || index2>count - 1));
                    A += A[index1] + A[index2];
          }
          GraphicElement& operator[](unsigned int);
          void operator+=(GraphicElement&);
          friend ostream& operator<<(ostream& , RasterGraphic&);
};
```

```
// ass2.cpp
#define _CRT_SECURE_NO_WARNINGS
#define _CRTDBG_MAP_ALLOC     // need this to get the line identification
//_CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF|_CRTDBG_LEAK_CHECK_DF); // in main, after local declarations
//NB must be in debug build
#include <crtdbg.h>
#include <iostream>
#include <string>
#include <vector>
#include <forward_list>
using namespace std;
#include "Image.h"
#include "GraphicElement.h"
#include "RasterGraphic.h"
bool running = true;

int main(void)
{
          char selection;
          bool running = true;
          RasterGraphic A("A");
          _CrtSetDbgFlag(_CRTDBG_ALLOC_MEM_DF | _CRTDBG_LEAK_CHECK_DF);
          while (running)
          {
                    cout<<    "MENU\n 1. Insert a GraphicElement\n 2. Delete the first GraphicElements\n 3. Concatenate two
GraphicElements\n 4. Run the RasterGraphic\n 5. Quit\n"<<endl;
                    cin >> selection;
                    switch (selection)
                    {
                    case '1':
                              A.InsertGraphicElement();
                              break;
                    case '2':
                              A.DeleteGraphicElement();
                              break;
                    case '3':
                              A.Concatenate();
                              break;
                    case '4':
                              cout << A << endl;
                              break;
                    case '5':
                              running = false;
                              break;
                    default:
                              break;
                    }
          }
          return 0;
}
```

Example Output

```
MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

1
Insert a GraphicElement in the RasterGraphic
Please enter the GraphicElement filename: Graphic_Element_1
Entering the GraphicElement Images (the sets of dimensions and durations)
Please enter the number of Images: 2
Please enter pixel x for Image #0 pixel_x:0
Please enter pixel y for Image #0 pixel_y:0
Please enter the duration sec for this Image: 2
Please enter the name for this Image: Image_1
Please enter pixel x for Image #1 pixel_x:16
Please enter pixel y for Image #1 pixel_y:32
Please enter the duration sec for this Image: 1
Please enter the name for this Image: Image_2
This is the first GraphicElement in the list

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

1
Insert a GraphicElement in the RasterGraphic
Please enter the GraphicElement filename: Graphic_Element_2
```

```
Entering the GraphicElement Images (the sets of dimensions and durations)
Please enter the number of Images: 3
Please enter pixel x for Image #0 pixel_x:1
Please enter pixel y for Image #0 pixel_y:1
Please enter the duration sec for this Image: 1
Please enter the name for this Image: Image_3
Please enter pixel x for Image #1 pixel_x:64
Please enter pixel y for Image #1 pixel_y:32
Please enter the duration sec for this Image: 4
Please enter the name for this Image: Image_4
Please enter pixel x for Image #2 pixel_x:128
Please enter pixel y for Image #2 pixel_y:256
Please enter the duration sec for this Image: 6
Please enter the name for this Image: Image_5

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

4
RasterGraphic A
Run the RasterGraphic
GraphicElement #0:
        fileName = Graphic_Element_1
        Image #0:       name = Image_1; pixel_x = 0; pixel_y = 0; duration =  2
        Counting the seconds for this Image: 1, 2,
        Image #1:       name = Image_2; pixel_x = 16; pixel_y = 32; duration =  1
        Counting the seconds for this Image: 1,
GraphicElement #1:
        fileName = Graphic_Element_2
        Image #0:       name = Image_3; pixel_x = 1; pixel_y = 1; duration =  1
        Counting the seconds for this Image: 1,
        Image #1:       name = Image_4; pixel_x = 64; pixel_y = 32; duration =  4
        Counting the seconds for this Image: 1, 2, 3, 4,
        Image #2:       name = Image_5; pixel_x = 128; pixel_y = 256; duration =  6
        Counting the seconds for this Image: 1, 2, 3, 4, 5, 6,

Output finished

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

3
Concatenating two GraphicElements
Please enter valid indexes of the two GraphicElements to concatenate (0 to 1)
1
0

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

4
RasterGraphic A
Run the RasterGraphic
GraphicElement #0:
        fileName = Graphic_Element_2_Graphic_Element_1
        Image #0:       name = Image_3; pixel_x = 1; pixel_y = 1; duration =  1
        Counting the seconds for this Image: 1,
        Image #1:       name = Image_4; pixel_x = 64; pixel_y = 32; duration =  4
        Counting the seconds for this Image: 1, 2, 3, 4,
        Image #2:       name = Image_5; pixel_x = 128; pixel_y = 256; duration =  6
        Counting the seconds for this Image: 1, 2, 3, 4, 5, 6,
        Image #3:       name = Image_1; pixel_x = 0; pixel_y = 0; duration =  2
        Counting the seconds for this Image: 1, 2,
        Image #4:       name = Image_2; pixel_x = 16; pixel_y = 32; duration =  1
        Counting the seconds for this Image: 1,
GraphicElement #1:
        fileName = Graphic_Element_1
        Image #0:       name = Image_1; pixel_x = 0; pixel_y = 0; duration =  2
        Counting the seconds for this Image: 1, 2,
        Image #1:       name = Image_2; pixel_x = 16; pixel_y = 32; duration =  1
        Counting the seconds for this Image: 1,
GraphicElement #2:
        fileName = Graphic_Element_2
        Image #0:       name = Image_3; pixel_x = 1; pixel_y = 1; duration =  1
        Counting the seconds for this Image: 1,
        Image #1:       name = Image_4; pixel_x = 64; pixel_y = 32; duration =  4
        Counting the seconds for this Image: 1, 2, 3, 4,
        Image #2:       name = Image_5; pixel_x = 128; pixel_y = 256; duration =  6
        Counting the seconds for this Image: 1, 2, 3, 4, 5, 6,

Output finished

MENU
```

```
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

2
Delete the first GraphicElement from the RasterGraphic
GraphicElement deleted

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

4
RasterGraphic A
Run the RasterGraphic
GraphicElement #0:
        fileName = Graphic_Element_1
        Image #0:       name = Image_1; pixel_x = 0; pixel_y = 0; duration =  2
        Counting the seconds for this Image: 1, 2,
        Image #1:       name = Image_2; pixel_x = 16; pixel_y = 32; duration =  1
        Counting the seconds for this Image: 1,
GraphicElement #1:
        fileName = Graphic_Element_2
        Image #0:       name = Image_3; pixel_x = 1; pixel_y = 1; duration =  1
        Counting the seconds for this Image: 1,
        Image #1:       name = Image_4; pixel_x = 64; pixel_y = 32; duration =  4
        Counting the seconds for this Image: 1, 2, 3, 4,
        Image #2:       name = Image_5; pixel_x = 128; pixel_y = 256; duration =  6
        Counting the seconds for this Image: 1, 2, 3, 4, 5, 6,

Output finished

MENU
 1. Insert a GraphicElement
 2. Delete the first GraphicElements
 3. Concatenate two GraphicElements
 4. Run the RasterGraphic
 5. Quit

5
Press any key to continue . . .
```