PA1.1. PA1 (2021W2) - A 2D doubly-linked list

Due: Feb.07, 22:00

Goals and Overview

In this PA (Programming Assignment) you will:

- learn about the course programming environment
- learn about vectors
- learn about pointers
- learn about linked lists
- learn about management of dynamic memory

Credit for this Assignment

There are 26 points worth of tests on this assignment. 6 of those points are for optional work (marked "OPTIONAL, FOR BONUS" below). You can earn full credit on the *assignment* (but not this problem) by passing any 20 test points, and once you reach that threshold your overall PA1 grade will be 100%.

To earn the bonus marks, you must pass at least 23 (1 bonus point) or all 26 (2 bonus points) of the test points. Your PA1 grade will still show as 100%. What are the bonus points worth? Some tiny boost to your course grade that is still-to-be-determined but definitely not worth the very large amount of work required. **Don't attempt the bonus problems until you've finished the rest of the assignment!** And then don't attempt the bonus problems unless you really want to do it just for fun and learning! Note: we reserve the right to require manual inspection of code before assigning bonus points.

Important note on the tests: Until you make progress on failing tests, you may not see all 26 test points reported. (That will not keep you from earning 100% once you pass 20 points of tests, however.)

The Assignment

Problem Specification

A Linked List is a dynamic linear structure designed to hold any type of data. In this exercise, we develop and use a 2-dimensional doubly-linked list to manipulate pixels from an image. (Technically, a 2-D linked list is not a linked *list* at all. This is more like a linked representation of a 2-D array, grid, or matrix. Structures like this are commonly used, though in use they typically store arrays of pointers to the start of each row/column. The gaps or holes we allow in our grid are especially useful for representing enormous matrices where much of the matrix may be simply zeroes.)

We define a class called `ImgList`, which is a 2-dimensional doubly-linked list constructed out of `ImgNode` objects.

An `ImgNode` consists of member attributes that describe:

- its colour
- four `ImgNode` pointers linking to its neighbours to the north/east/south/west, if available
- four integers describing how many pixels spaces to fill during reconstruction of the overall image data, if any neighbours have been deleted from the list

The `ImgList` provides entry into the linked structure at two locations (`northwest` and `southeast`) and an example 8x8 structure with its (enlarged) image blueprint is shown below:

Programming Assignment 1

Assessment overview

Total points: 20/20

Score: 100%

Question

Value: 26

History: 21

Awarded points: 21/26

You are viewing the question instance of a different user and so are not authorized to report an error.

Previous question

Next question

Attached files

No attached files

You are viewing the question instance of a different user and so are not authorized to add or delete files.

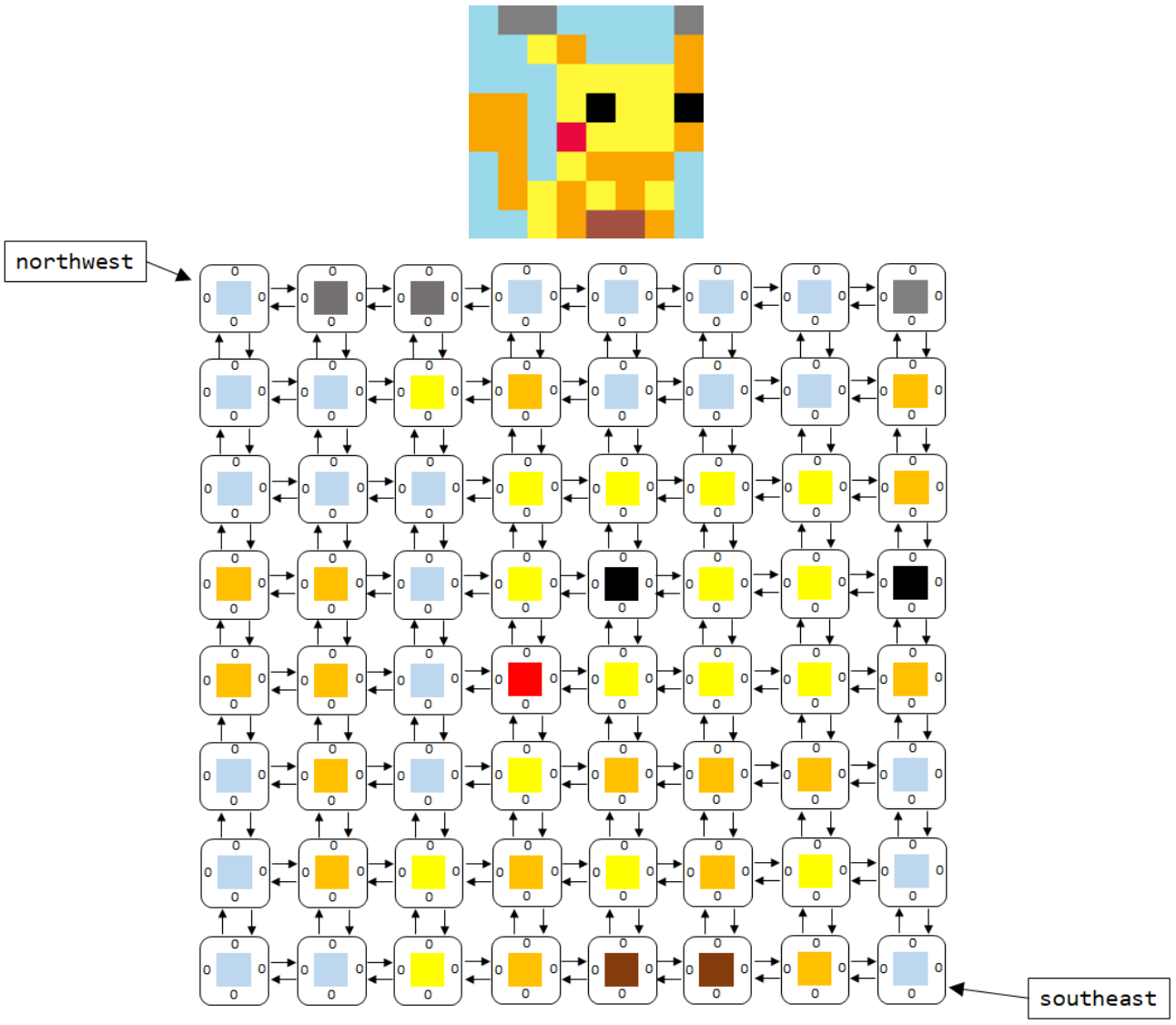
Staff information

Staff user:

Yutong Li
ytongli@ubc.ca

Student:

Xiaoyang Zhang
azxyxza@ubc.ca

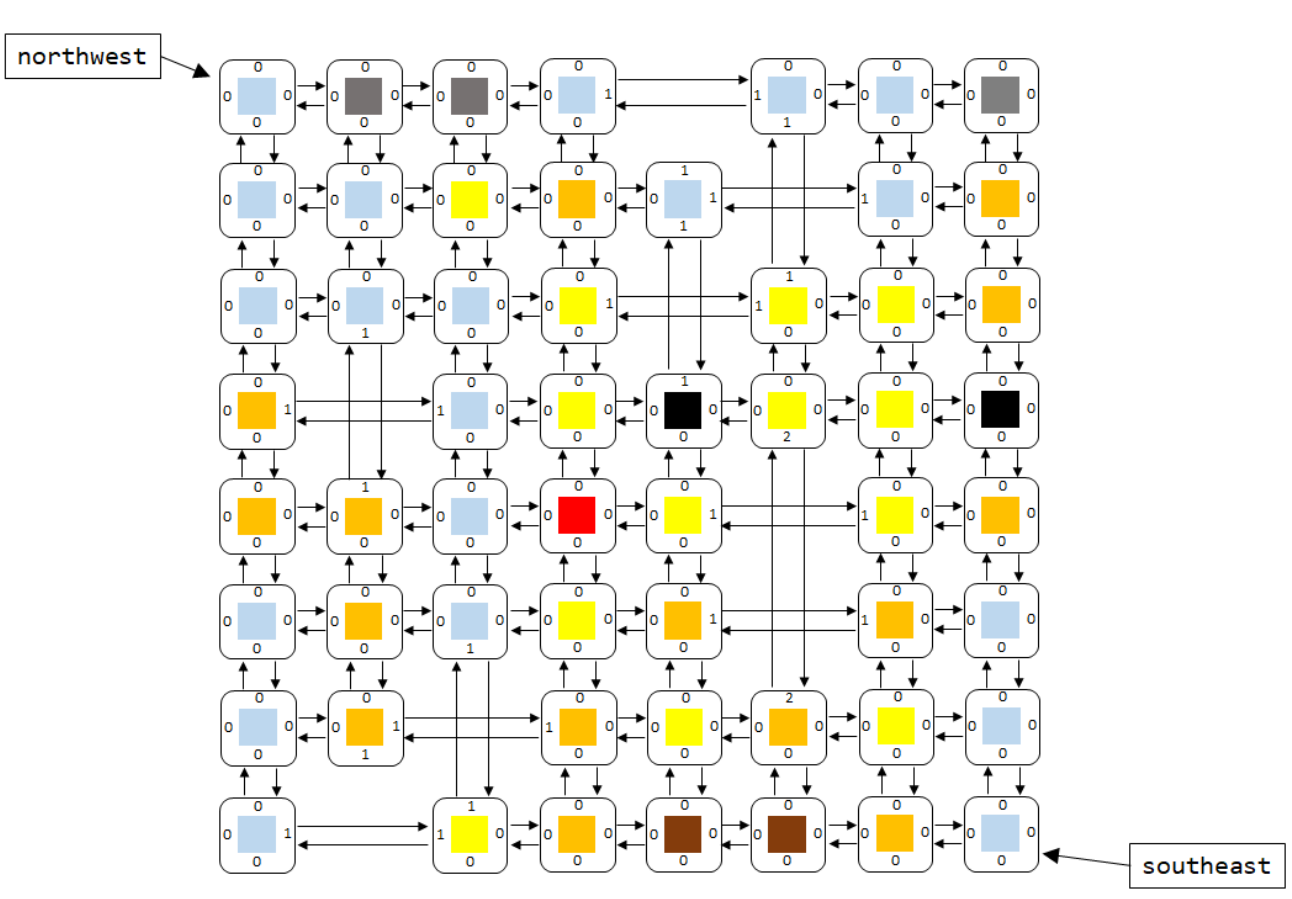


Any node that has fewer than four outgoing pointers will have a value of `NULL` for each pointer not illustrated.

In this assignment, you will construct an `ImgList` object using `HSLAPixel` data from an input `PNG` image, implement a function which removes a node from each row of nodes, and reconstruct and render a `PNG` image from (possibly reduced) `ImgList` content.

We have provided a starting point for achieving this functionality. It is your task to complete and expand on our implementation.

The most notable feature of this structure is the ability to "carve" away one or more nodes from each row in the `ImgList`, thereby reducing the amount of memory needed to store the `ImgList` object. When removing nodes, the "skip" attributes of neighbouring nodes must be updated to reflect the number of nodes that previously existed in the gap which gets created/enlarged. An example list with one node removed from each row is illustrated below:



A subsequent round of removal (for a total of 2 pixels removed from each row) is illustrated below. Pay special attention to the "skip" values of each node that is adjacent to a gap:

Question:

QID: pas/pa1_imglist
Title:
PA1 (2021W2) - A 2D doubly-linked list
Started at:
2022-01-24 09:22:42 (PST)
Duration:
03:41:53.167921

Show/Hide answer

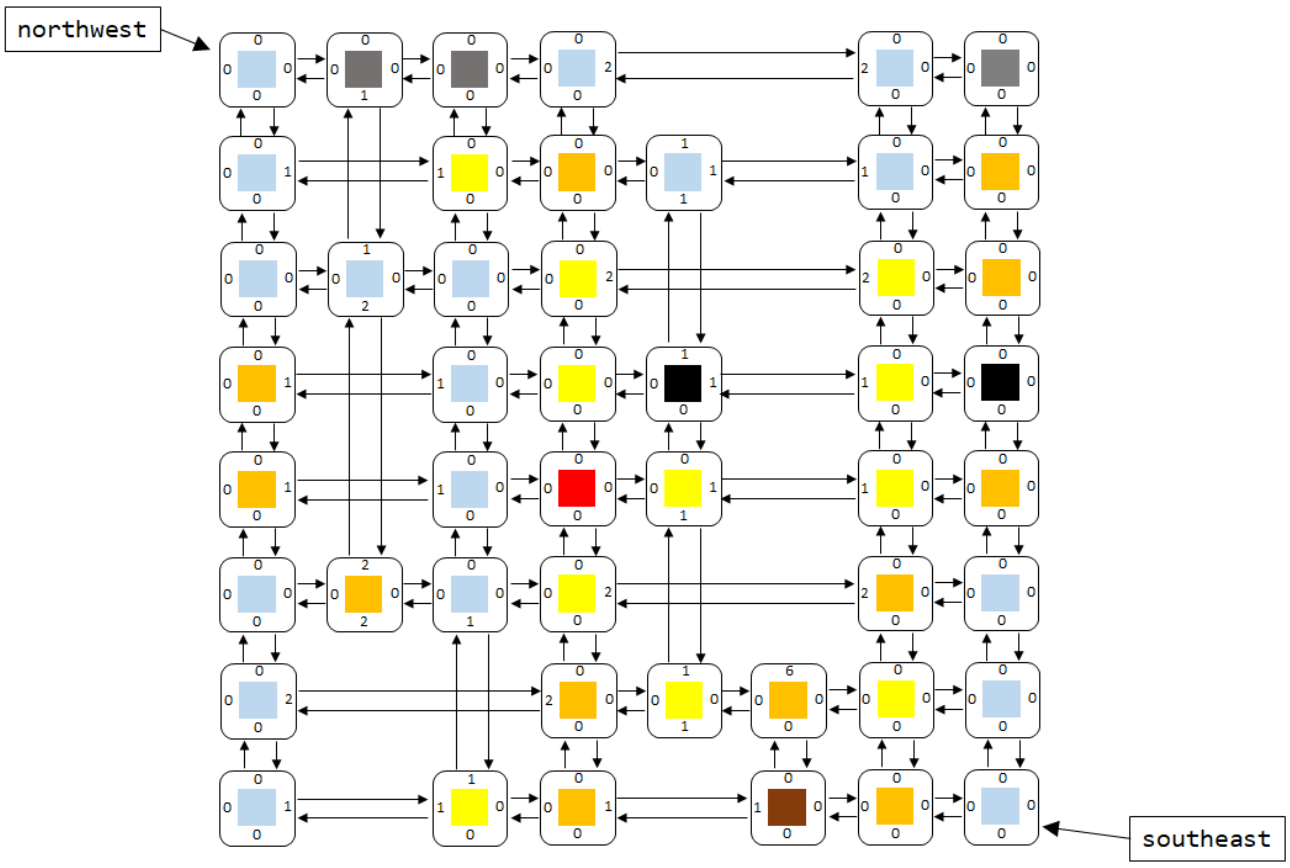
Assessment Instance:

AID: pas/pa1
Started at:
2022-01-24 09:22:39 (PST)
Duration:
03:41:53.167921
View log

Manual Grading:

Status: Graded
Grade

This box is not visible to students.



Note that the specific nodes selected for removal in the figures above may not accurately match the selection criteria which you are asked to implement; they are shown only to illustrate how the pointer links and skip values should be updated.

Specifications for each function you write are contained in the given code. The list of functions here should serve as a checklist for completing the exercise.

In `imglist.cpp`

- `ImgList()`: default constructor creates an empty list.
- `ImgList(PNG& img)`: constructs a new list using pixel data from `img`.
- `void Clear()`: Helper function for destructor and assignment operator.
- `void Copy(const ImgList & otherlist)`: (OPTIONAL, FOR BONUS) Helper function for copy constructor and assignment operator. BEWARE: Copying the north/south links correctly is challenging, but you *can* do it in $O(\text{number of remaining pixels} + \text{original width})$ time (using $O(\text{original width})$ additional space) or $O(\text{number of remaining pixels}^2)$ time.
- `unsigned int GetDimensionX() const`: retrieves the number of nodes in each row of the list; the horizontal dimension of the list.
- `unsigned int GetDimensionY() const`: retrieves the number of nodes in the tallest column of the list; the vertical dimension of the list. This will always correspond to the original height of the `PNG` image used to construct the list.
- `unsigned int GetDimensionFullX() const`: retrieves the original pixel width of the `PNG` image used to construct the list. Computed using skip values.
- `ImgNode* SelectNode(ImgNode* rowstart, int selectionmode)`: From the row beginning with `rowstart`, returns a pointer to the node which best satisfies the selection criteria determined by `selectionmode`. The selected node MUST have non-NULL left and right neighbours.
- `void Carve(int selectionmode)`: Removes a node from every row in the list. Hint: use the `SelectNode` function.
- `void Carve(unsigned int rounds, int selectionmode)`: Removes `rounds` number of nodes from each row of the list.
- `PNG Render(bool fillgaps, int fillmode) const`: Draws the entire `ImgList`'s pixel data into an appropriately-sized PNG and returns it. `fillgaps` and `fillmode` are used to determine whether a (possibly carved) list is rendered to its current dimensions, or to its pre-carving dimensions, and how any gaps due to carving are filled. (`fillmode == 2` uses a linear interpolation to fill the gaps and will be OPTIONAL, FOR BONUS. See the documentation in `imglist.cpp` for details. BEWARE: Getting all the details just right on this can be challenging!)

Implementation Constraints and Advice

We will be grading your work on functionality, efficiency, and memory use. All `ImgList` functionality, aside from the constructor(s) and the copy functions, can be achieved by manipulating node pointers, rather than by allocating new ones and/or making copies. If you

are tempted to use the `new` function when you are manipulating the `ImgList`, ask yourself if you can achieve your goal by reassigning pointers, instead.

Getting the Given Code

Download the source files here pa1-2021w2-20220205-2259.zip and follow the procedures you learned in lab to move them to your home directory on the remote linux machines or to set up a project in a local development environment.

Handing in your code

Pair work is encouraged, although both partners must submit code separately. Before you hand in this assignment, create a file called `partners.txt` that contains only the CWLs of people in your collaboration partnership (if one exists), one per line. If you worked alone, include only your own CWL in this file. We will be automatically processing this information, so do not include anything else in the file. Failure to cite collaborators violates our academic integrity policy.

Submit your files for PA1 grading here! These are the only files used for grading.

Drop files here or click to upload.

Only the files listed below will be accepted—others will be ignored.

The combined size limit of all uploaded files is 5MB.

Files

imglist.h uploaded	Download Show preview
imglist.cpp uploaded	Download Show preview
partners.txt uploaded	Download Show preview

All other files will not be used for grading.

You are viewing the question instance of a different user and so are not authorized to save answers, to submit answers for grading, or to try a new variant of this same question.

Submitted answer 20 **partially correct: 80%**

Submitted at 2022-02-03 15:12:58 (PST)

hide

Files

imglist.cpp uploaded	Download Show preview
imglist.h uploaded	Download Show preview
partners.txt uploaded	Download Show preview

Score: 21/25 (80.77%)

Output

```
Test group imglist output:
stdout: clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/PNG.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/HSLAPixel.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/lodepng/lodepng.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
testimglist.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
imglist.cpp
clang++ PNG.o HSLAPixel.o lodepng.o testimglist.o imglist.o -stdlib=libc++
-std=c++1y -lc++abi -lpthread -lm -o testimglist

stderr: imglist.cpp:514:35: warning: unused parameter 'otherlist' [-
Wunused-parameter]
void ImgList::Copy(const ImgList &otherlist)
                                  ^
1 warning generated.

Test group memtest output:
stdout: clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/PNG.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/HSLAPixel.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/lodepng/lodepng.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
testimglist.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
imglist.cpp
clang++ PNG.o HSLAPixel.o lodepng.o testimglist.o imglist.o -stdlib=libc++
-std=c++1y -lc++abi -lpthread -lm -o required

stderr: imglist.cpp:514:35: warning: unused parameter 'otherlist' [-
Wunused-parameter]
void ImgList::Copy(const ImgList &otherlist)
                                  ^
1 warning generated.

Test group OPTIONAL_memtest_linear_interpolation output:
stdout: clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/PNG.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/HSLAPixel.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/lodepng/lodepng.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
testimglist.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
imglist.cpp
clang++ PNG.o HSLAPixel.o lodepng.o testimglist.o imglist.o -stdlib=libc++
-std=c++1y -lc++abi -lpthread -lm -o OPTIONAL_linear_interpolation

stderr: imglist.cpp:514:35: warning: unused parameter 'otherlist' [-
Wunused-parameter]
void ImgList::Copy(const ImgList &otherlist)
                                  ^
1 warning generated.

Test group OPTIONAL_memtest_copy output:
stdout: clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/PNG.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
```

```
cs221util/HSLAPixel.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
cs221util/lodepng/lodepng.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
testimglist.cpp
clang++ -stdlib=libc++ -std=c++1y -c -g -O0 -Wall -Wextra -pedantic
imglist.cpp
clang++ PNG.o HSLAPixel.o lodepng.o testimglist.o imglist.o -stdlib=libc++
-std=c++1y -lc++abi -lpthread -lm -o OPTIONAL_copy

stderr: imglist.cpp:514:35: warning: unused parameter 'otherlist' [-
Wunused-parameter]
void ImgList::Copy(const ImgList &otherlist)
                                  ^

1 warning generated.
```

Test Results

- ✔ **[1/1]** imglist::default constructor
- ✔ **[3/3]** imglist::png constructor small
- ✔ **[1/1]** imglist::GetDimensionXY
- ✔ **[1/1]** imglist::GetDimensionFullX
- ✔ **[2/2]** imglist::Render_nocarve_nofill
- ✔ **[2/2]** imglist::Carve_1px_sel0
- ✔ **[2/2]** imglist::Carve_2px_sel0
- ✔ **[3/3]** imglist::Carve_15px_sel1
- ✔ **[2/2]** imglist::Render_fill0
- ✔ **[2/2]** imglist::Render_fill1
- ✘ **[0/2]** OPTIONAL imglist::Render_fill2
- ✘ **[0/1]** OPTIONAL imglist::Copy_constructor
- ✔ **[1/1]** [memory] required
- ✔ **[1/1]** [memory] OPTIONAL_linear_interpolation
- ✘ **[0/1]** [Memory test] OPTIONAL_copy

Submitted answer 19 **partially correct: 73%**
Submitted at 2022-02-03 15:06:27 (PST)
show

Submitted answer 18 **partially correct: 65%**
Submitted at 2022-02-03 00:08:13 (PST)
show

Show/hide older submissions