PA2.1. PA2 Fancy Fills

2022/5/1 14:05

# Due: Mar 7, 22:00 Goals and Overview

In this PA (Programming Assignment) you will:

- learn about Stacks and Queues
- learn about applications of Stacks and Queues called depth-first-search (DFS) and breadth-first-search (BFS)
- · explore flood fill algorithms.
- become more skilled at C++ constructs, including the use of basic templates and inheritance

# Part 1: The Stack, Queue, and PriorityNeighbours Classes

**Template Compilation** Note that the stack.h and queue.h files we've given you include the stack.cpp and queue.cpp files respectively at the end, so you should NOT use a #include "stack.h" (or #include "queue.h") statement at the top of the stack.cpp (or queue.cpp) file. We have done it this way so that the template types are instantiated appropriately.

### The Stack Class

You will write a class named Stack that works just like the stack you heard about in lecture, with the addition of the Peek() function. It is declared in the given file stack.h. You will implement it in stack.cpp.

Your Stack class must implement all of the methods mentioned in the given code. Please read the documentation in the header file to see what limitations we have placed on your Stack class and what the running times of each function should be.

# The Queue Class

You will write a class named Queue that works just like the queue you heard about in lecture, with the addition of the Peek() function. It is declared in the given file queue.h. You will implement it in queue.cpp.

Your Queue class must implement all of the methods mentioned in the given code. Please read the documentation in the header file to see what limitations we have placed on your Queue class and what the running times of each function should be.

# The PriorityNeighbours Class

You will write a class named PriorityNeighbours that is designed specifically to be used in this PA. It is used to store a collection of (x,y) image coordinates along with some color information. Its behaviour when an item is removed, is to select the item from among its contents that best satisfies a priority condition. Please carefully read the documentation in priority. In for a detailed explanation of the priority condition.

# **Testing**

We have provided test files testStackQueue.cpp and testPriority which include a few (albeit trivial) test cases that your code should pass if it is correct. To compile this executable, type:

make testStackQueue

Programming
Assignment 2

Assessment
overview

Total 27/27
points:

Score: 100%

Value: 27

History: 27

Awarded points: 27/27

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

make testPriority

or

These test cases are deliberately insufficient. We encourage you to augment this file with additional test cases, using the provided ones as examples.

# **Grading Information for Part 1**

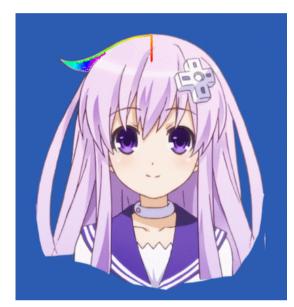
The following files are used to grade the stack and queue classes:

- stack.cpp
- queue.cpp
- priority.h
- priority.cpp

All other files (including any testing files you create) will not be used for grading.

# Part 2: Flood Fill





For this part of the assignment, you will be writing a number of functions that execute a "flood fill" on a PNG image. To flood fill a region of an image, you specify a point on the image and a fill color (or pattern) and all points similar in color and adjacent to the chosen point are changed to the fill color (or pattern). We will implement two different fill algorithms and three different fill patterns. The two fill algorithms are animated in the pictures above with a rainbow color fill pattern that we implemented.

### **Functors**

Your first task in this part of the project is to implement the mechanism by which colors are selected for the fill pattern. To do this, you will create function objects (also known as "functors") which will be passed into the fill algorithm and applied to pixels in a PNG. Specifically you will be making three new functors, each of which will be derived classes of the abstract base class called colorPicker, which is provided and described in the given code file colorPicker. h. We have also given you an example functor so you can see how things are supposed to work. Our functor is called rainbowColorPicker (its behavior is a little non-deterministic--er-buggy, but it's cool to look at!). The colorPicker base class has one purely virtual function that you must implement in your derived classes, but you may add other functions if you wish. You should expect your derived class constructors to initialize the functor with situation-dependent variables, and it is up to you do decide what those variables should be.

The purpose of any colorPicker is to take an image coordinate as its (x,y) parameters to operator(), and return the color it would suggest the client recolor that location in the source PNG to be. So in the examples above, the colorPicker is returning rainbow colors, based on the number of iterations in the fill algorithm.

Your task is to implement the ImageTileColorPicker, NegativeColorPicker, and CensorColorPicker function objects. Refer to the given code for implementation details (particularly solidColorPicker.com and rainbowColorPicker.com as they

#### Staff user:

Yutong Li

ytongli@ubc.ca

#### Group:

XforAce

(azxyxza@ubc.ca, xiaorang@ubc.ca)

#### Question:

QID:

pas/pa2 fills 2021W2

Title:

PA2 Fancy Fills

Started at:

2022-02-15 08:45:11 (PST)

Duration:

04:47:22.584261

Show/Hide answer

# Assessment Instance:

AID:

pas/pa2

Started at:

2022-02-15 08:45:08 (PST)

Duration:

04:47:22.584261

are already completed): they **must** overwrite the abstract **operator()**!

The Fill Algorithms

dotallo (partiodiarry politaco torri teller reppiaria

You will be implementing two different basic kinds of flood fills: depth-first-search (DFS) and breadth-first-search (BFS), each with three different patterns. The first pattern will be a simple replacement of pixels from a different image into the destination image (ImageTileColorPicker). The second will be filling in the area with negated hue and saturation values of the original color (NegativeColorPicker), and the third produces a blocky mosaic censor using averaged colors, covering a circular region of specified size and location (CensorColorPicker). The animations above are breadth-first-search and depthfirst-search, respectively, both with a rainbow fill. We have written the rainbow fill functions as an example for the others.

You must implement **all** of the functions in the filler namespace. Refer to the documentation in the header file for implementation details.

# **Testing**

Note that you must have ImageMagick installed if you're developing on your personal machine. You can test for this requirement by running

convert

in your terminal. Windows users should visit <u>ImageMagick</u> to download the latest binary. OSX users can use homebrew or MacPorts to install it if missing. Linux users likely already have it installed. If not, then you can install it via your system's package manager.

We have provided a file testFills.cpp which includes several test cases that your code should pass if it is correct. To compile this executable, type:

make testFills

This will create executable testFills.

One important thing to note about this PA is that the .gif compression may take some time. In general, on the remote machines via ssh, the solution code runs in somewhere from 5-20 seconds when all the tests are enabled and completed and the server is under a light load. It is advisable to comment out the tests for things you have already completed, so that you don't have to wait as long (and use the CPU up) for things you already did. That said, the main point is that if your code doesn't finish immediately don't fret. Also note that as more traffic is happening on the server things will slow down. This may be an incentive to not wait until the last minute and have to try to run your code when everyone else is too!

(These slow tests may feel unrealistic. The opposite is true: trading off which expensive tests to run and how often is an absolutely critical workflow choice in substantial projects in industry!)

The testFills program will generate images and animated gifs in the images/ folder, which can be diffed with the solution images in soln\_images/. For animated gifs, the frames making up the animation are placed in frames/, and can be diffed with their corresponding expected frames in soln\_frames/.

You can diff each frame, which will probably be useful for debugging:

diff frames/bfssolid00.png soln\_frames/bfssolid00.png

These test cases are deliberately insufficient. We encourage you to augment this file with additional test cases, using the provided ones as examples.

View log Manual **Grading:** Status: Graded <u>Grade</u> This box is not visible to

students.

# Handing in Your Code

Pair work is highly encouraged for this assignment. Each group member will be able to make joint submissions and have access to submissions made by other members.

# **Grading Information**

The following files are used to grade PA 2:

- stack.cpp
- queue.cpp
- priority.h
- priority.cpp
- imageTileColorPicker.h
- imageTileColorPicker.cpp
- negativeColorPicker.h
- negativeColorPicker.cpp
- censorColorPicker.h
- censorColorPicker.cpp
- filler.h
- filler.cpp

All other files will not be used for grading.

# Implementation Constraints and Advice

For Part 1, we *strongly* encourage you to sketch out small examples to help you understand how the Queue class should work.

For Part 2, you are also welcome to add your own helper functions. You'll be submitting both the .h and .cpp files for that part of the assignment. (The same is *not* true for the Stack and Queue of Part 1.)

# Getting the Given Code

Download the source files here <u>pa2-20220222-2317.zip</u> and follow the procedures you learned in lab to move them to your home directory on the remote linux machines.

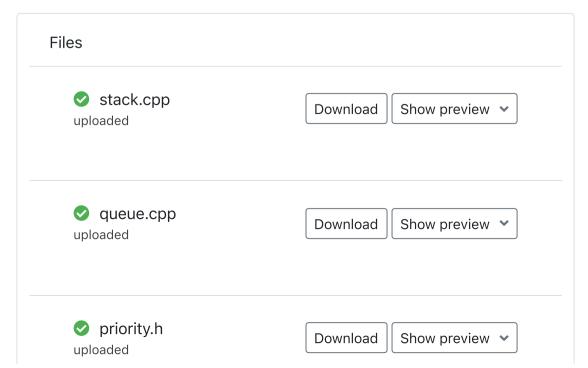
## **Submission**

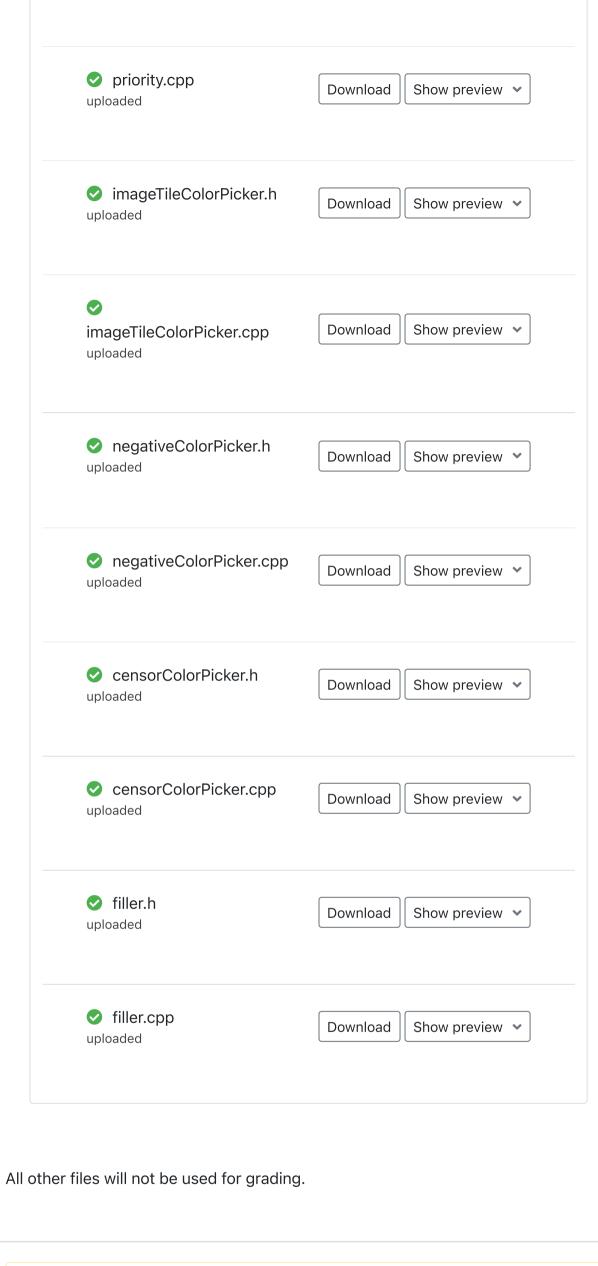
Submit your files for PA2 grading here! These are the only files used for grading.

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.

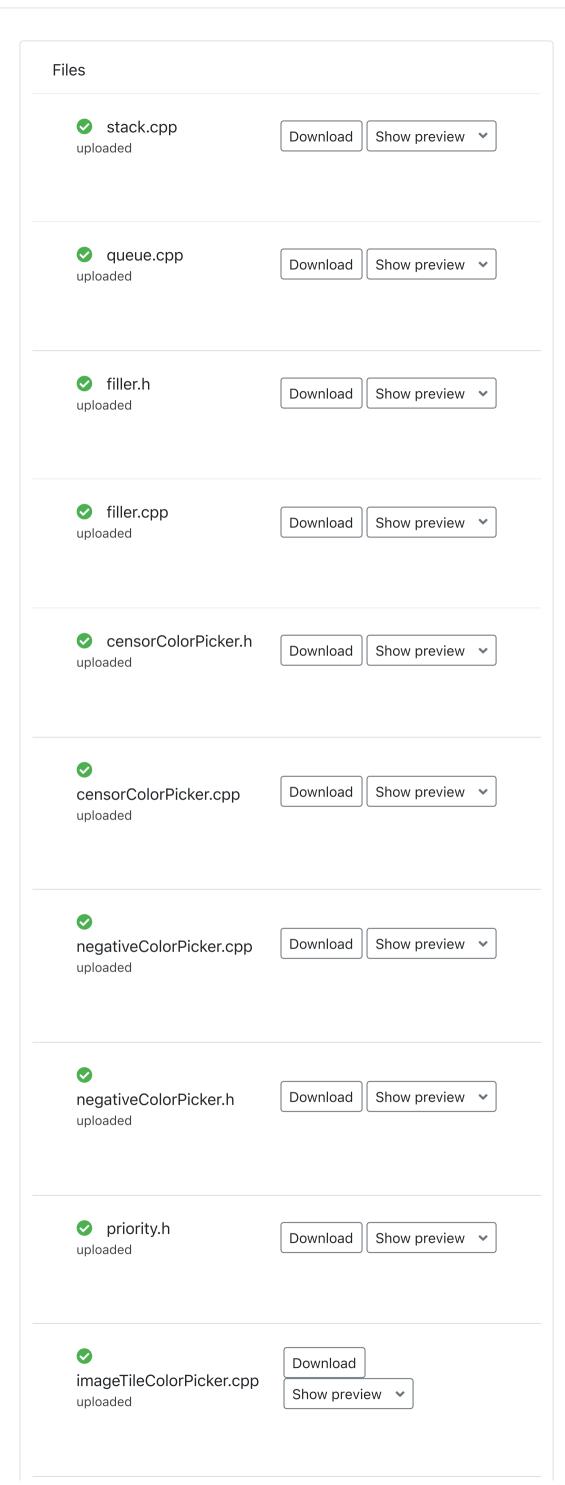




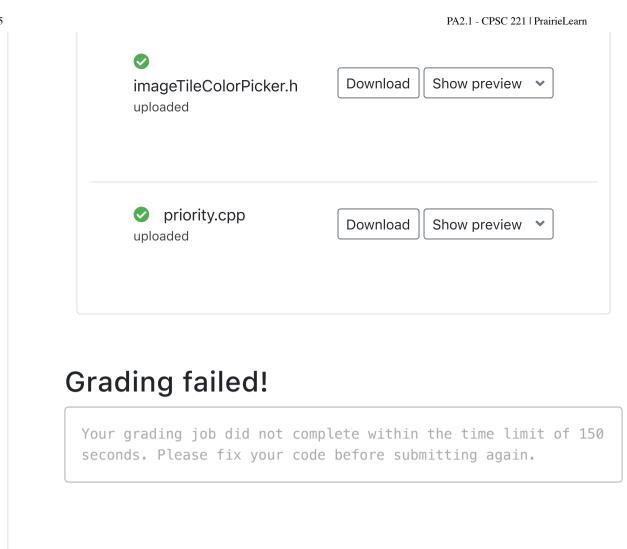
You are viewing the question instance of a different user and so are not authorized to save answers, to submit answers for grading, or to try a new variant of this same question.

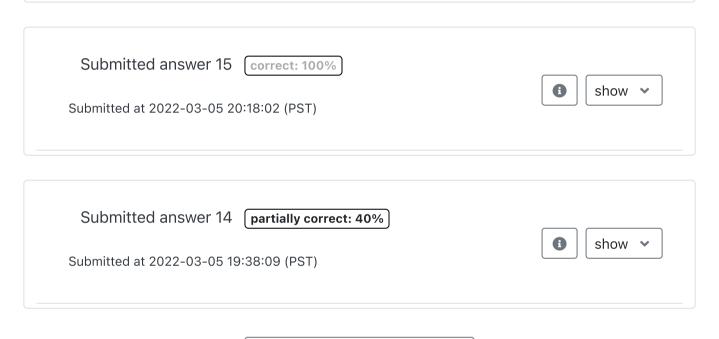
Submitted answer 16 incorrect: 0%

Submitted at 2022-03-06 22:57:33 (PST)



2022/5/1 14:05





Show/hide older submissions 💌