CIS 410/510: Project #8
Due February 22, 2022
(which means submitted by 6am on February 23, 2022)
Worth 8 points toward your grade

Assignment:
1) Download skeleton code proj8.cxx
2) Download data file noise.vtk
3) Compile and run (using your CMakeLists.txt from previous projects) and
   make sure it runs correctly

You will do a series of seven tasks to learn VTK. Each time you complete a task, you
should make a copy of your source. You will be handing in your source code for
each task. They should be named proj8A.cxx, proj8B.cxx, proj8C.cxx, ... proj8G.cxx.
What to upload? A tarball with 8 files: proj8A.cxx-proj8G.cxx + one screenshot
showing proj8F.cxx working.

What are the seven tasks?:

A) Make the visualization window be 768x768 when the program is first invoked.
B) Add a contour filter to the pipeline, with isovalues at 2.4 & 4.
C) Remove the contour filter and instead slice by Z=0
D) Modify the vtkLookupTable so that it smoothly interpolates from blue at the
minimum to red at the maximum, with purple in between.
E) Modify your program to have both a contour and a slice filter. You will need to
have two networks.
F) Add two renderers. Place the slice in the left renderer and the contour in the
right.
G) Modify your program to iterate over isovalues. Start with 1.0 and add 0.02 all the
way up to 6 (a total of 250 isovalues). Render each new value. This will make an
animation.

Hints:

First, the execution model appears to be a little buggy, meaning that it doesn't
always automatically execute the filters. I found I needed to add many "Update()"
calls to make modules update. My advice is to call "Update()" whenever you feel you
have a filter that is ready to execute. (This is different than how I lectured.)

=== 8C ===
There is no module for slicing. Instead, you use the module "vtkCutter" and then set
up a vtkPlane and assign it as the vtkCutter's "CutFunction".

=== 8D ===

The vtkLookupTable has 256 entries. You want to set the color for each of these 256 entries, manually interpolating from blue to red.

=== 8E ===
The output from the slice and the output from the cutter will each need to have its own mapper and actor (so two mappers and two actors).

=== 8F ===
You will need two vtkRenderers and to add them to the vtkRenderWindow. Also, you'll need a SetViewport command.
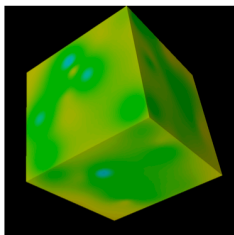
=== 8G ===
Never call vtkRenderWindowInteractor::Start().
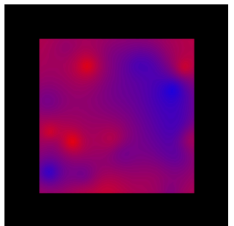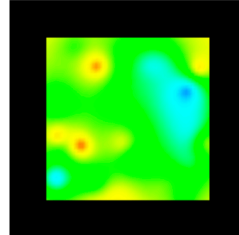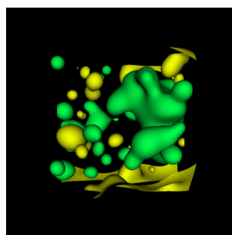Your program should just end after the 250 iterations.
I found that the camera wasn't setting itself correctly. Each time I updated the isovalue, I did:
```
cf->Update();
ren2->GetActiveCamera()->ShallowCopy(ren->GetActiveCamera());
```
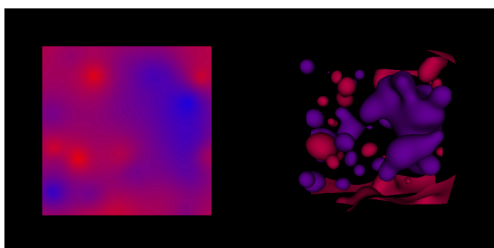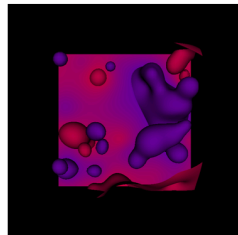
Note: this screenshot refers to the parts as 1A, 1B, etc., instead of 8A, 8B, etc.

Result from 1A, 1B, 1C
(1A after rotation)

Result from 1D, 1E

Result from 1F

Result from 1G: image on right has single isovalue that animates.

VTK bug: for some operating systems, your 8G code will not show up. It should be that "vtkRenderWindow::Render()" causes the window to appear on the screen. But the bug is that the window will not appear until "vtkRenderWindowInteractor::Start()" is called. Of course, I am telling you not to call Start in 8G.

Here is a work-around to the bug. The idea is that you go ahead and use vtkRenderWindowInteractor::Start(), but that you add a callback. The callback does the part of the isosurface animation. With this mode, you start the program and you see the slice on the left. If you press 'a' at the keyboard, then my changes below will call your callback, which can do the animation. There is also some yuckiness with global variables.

First, here are the global variables I needed.

```
vtkRenderWindow *renwin = NULL;
vtkContourFilter *cf = NULL;
vtkRenderer *ren2 = NULL;
vtkRenderer *ren = NULL;
```

(your mileage may vary)

Be *very* careful as you set them up to not do "shadowing." For example, if your main code has
```
 vtkContourFilter *cf = vtkContourFilter::New();
```
then you would need to change it to:
```
 cf = vtkContourFilter::New();
```
The reason you need the change is to make sure you don't have two variables called "cf" – one in main and one a global variable. Further, if your global variable does "shadow" the variable in main (i.e., has the same name), then the global variable will never be set and you will be working with a NULL pointer.

Second, here is a function you need for the animation:

```
void Animate(vtkObject*caller, unsigned long eid, void* clientdata, void *calldata)
{
   vtkRenderWindowInteractor *iren =
       static_cast<vtkRenderWindowInteractor*>(caller);
   if (iren->GetKeySym()[0] != 'a')
        return;
   // code to animate goes here...
}
```

Third, I added these lines of code to main:

```
vtkRenderWindowInteractor *iren = vtkRenderWindowInteractor::New();
iren->SetRenderWindow(renwin);
renwin->Render();

// Here's the new code
vtkSmartPointer<vtkCallbackCommand> keypressCallback =
    vtkSmartPointer<vtkCallbackCommand>::New();
keypressCallback->SetCallback(Animate);
iren->AddObserver(vtkCommand::KeyPressEvent, keypressCallback);

iren->Start();
```