Technical HW 4 Solutions

Problem 1:

- Input the values in a csv record that is specified on the command line.
- Print the list of students in each course separately, sorted by score, as well as their overall averages.
- Print the total averages for each course.

Solution:

- One tricky part here is that you have the "operator<" method defined, but for StudentRecord. The storage is, however, shared_ptr<StudentRecord>. So, you need a function that compares their dereferenced values like:

```
bool comparescore( StudentRecordPtr const & r1, StudentRecordPtr
const & r2 ){
 return *r1 < *r2;
}
```

- Then you sort them like:

```
std::sort( vals->begin(), vals->end(), comparescore );
```

- You can then add the scores to a vector and average them:

```
for ( auto member : *vals ) {
  std::cout << *member << std::endl;
  scores.push_back( member->score() );
}
double average = std::accumulate( scores.begin(), scores.end(), 0.);
if ( scores.size() > 0 )
  average /= scores.size();
```

Problem 2: *Repeat problem 1, but in python.*

Solution:

The python interface is much easier for humans so the top-level looks much simpler. The classes themselves are nearly direct one-to-one correspondence. However, due to the fact that there is no need to store pointers, the sorting is a bit simpler. You need to just tell python what to sort by:

```
    for key,vals in records.items():
        vals = sorted( vals, key=lambda x: x.score())
```

Problem 3 PHY 505 ONLY:

The Diffusion equation is given by:

$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2} + \nu \frac{\partial^2 u}{\partial y^2}$$

In discretized form, the $n+1$st time step is related to the nth time step by:

$$u_{i,j}^{n+1} = u_{i,j}^{n} + \frac{\nu \Delta t}{\Delta x^2}(u_{i+1,j}^{n} - 2u_{i,j}^{n} + u_{i-1,j}^{n}) + \frac{\nu \Delta t}{\Delta y^2}(u_{i,j+1}^{n} - 2u_{i,j}^{n} + u_{i,j-1}^{n})$$

If we assume $\Delta t = 1, \Delta x = 1, \Delta y = 1, \nu = 1$, this becomes

$$u_{i,j}^{n+1} = u_{i,j}^{n} + (u_{i+1,j}^{n} + u_{i,j+1}^{n} - 4u_{i,j}^{n} + u_{i-1,j}^{n} + u_{i,j-1}^{n})$$

Starting with the jupyter notebook "Problem3.ipynb", implement one step of the diffusion equation for the interior of the square, with the values of the edges set to zero. Do this all WITHOUT using for loops! Hint: It should be a single line of numpy code.

Solution:

There should only one line to your solution. We are setting the interior of the array, so the indices will NOT be 0—>N, but 1—>N-1 on all sides. We can use the syntax "[1,-1]" to accomplish that. The rest of the values are just offset by a single value up, down, left, and right! So the equation is:

```
u[1:-1, 1:-1] = un[1:-1,1:-1] + un[1:-1, 2:] - 4 * un[1:-1, 1:-1] +
un[1:-1, 0:-2] + un[2:,1: -1] + un[0:-2, 1:-1]
```