```cpp
/*
 * ForceUtilities.cpp
 *
 *  Created on: Oct 19, 2013
 *      Author: arjunnar
 */

#include "PhysicsUtilities.h"
#include <vector>

using namespace std;

Vector3f PhysicsUtilities::getSpringForce(Vector3f xi, Vector3f xj, float
springConstant, float springRestLength)
{
    Vector3f d = xi - xj;

    float partialCalc = -1.0 * springConstant * (d.abs() - springRestLength);
    Vector3f force = ( partialCalc/d.abs() ) * d;
    return force;
}

Vector3f PhysicsUtilities::getGravityForce(float mass, float gravityConstant)
{
    return Vector3f(0.0, -1.0 * mass * gravityConstant, 0.0);
}

Vector3f PhysicsUtilities::getDragForce(float dragConstant, Vector3f velocityVector)
{
    return -1.0 * dragConstant * velocityVector;
}

Vector3f PhysicsUtilities::getPressureForce(float mj,
                                            float pi,
                                            float pj,
                                            float densityj,
                                            Vector3f gradSpikyKernel)
{
    // Eq(10)
    float avgPressure = (pi + pj) / 2.0;
    return (-1.0 * mj * avgPressure / densityj) * gradSpikyKernel;
}

Vector3f PhysicsUtilities::getPositionOfParticle(vector<Vector3f> &state, int
particleNum)
{
    return state[2 * particleNum];
}

Vector3f PhysicsUtilities::getVelocityOfParticle(vector<Vector3f> &state, int
particleNum)
{
    return state[2 * particleNum + 1];
}

void PhysicsUtilities::setPositionOfParticle(vector<Vector3f> &state, int particleNum,
Vector3f position)
{
    state[2 * particleNum] = position;
}

vector<Vector3f> PhysicsUtilities::getParticlePositions(vector<Vector3f> &state)
{
    vector<Vector3f> particlePositions;
    for (int i = 0; i < state.size(); i += 2)
    {
        particlePositions.push_back(state[i]);
    }
```

```
    return particlePositions;
}

void PhysicsUtilities::setVelocityOfParticle(vector<Vector3f> &state, int particleNum,
Vector3f velocity)
{
    state[2 * particleNum + 1] = velocity;
}

float PhysicsUtilities::getPressureAtLocation(float densityAtLoc, float restDensity,
float gasConstant)
{
    return gasConstant * fmax( pow(densityAtLoc/restDensity, 3.0) - 1, 0.0 );
    // Eq(12)
    //return gasConstant * (densityAtLoc - restDensity);
}

Vector3f PhysicsUtilities::getViscosityForce(float mj,
                                             float viscosityConstant,
                                             float densityj,
                                             float viscosityKernelLaplacian,
                                             Vector3f vj,
                                             Vector3f vi)
{
    // Eq(14)
    float constant = (viscosityConstant * mj) / densityj;
    Vector3f velocityDiff = vj - vi;
    return constant * viscosityKernelLaplacian * velocityDiff;
}
```