Help

```
#include <iostream>
#include <iostream>
#include <cstdlib>
#include <cmath>

#include "
href../../../../common/math/mcam/src/DupireModel_h_src.pdfModel.hpp"
#include "pnl/pnl_matrix.h"
#include "
href../../../../common/math/jlparser/include/jlparser/parser_h_src.pdfjlparser/p
#include "
href../../../../common/math/mcam/src/BlackScholesModel_h_src.pdfBlackScholesMode
#include "
href../../../../common/math/mcam/src/DupireModel_h_src.pdfDupireModel.hpp"
#include "
href../../../../common/math/mcam/src/HestonModel_h_src.pdfHestonModel.hpp"

using namespace std;

mcam::Model::Model()
{
    dates = 1;
    subticks = 1;
}

mcam::Model::Model(const Param &P)
{
    dates = 1;
    subticks = 1;
    P.extract("maturity", T);
    P.extract("dates", dates);
    P.extract("sub ticks", subticks);
    P.extract("model size", size);
    P.extract("spot", spot, size);
    P.extract("interest rate", r);
    P.extract("dividend rate", dividend, size);
    if (dividend == NULL)
    {
        dividend = pnl_vect_create_from_zero(size);
```

```
    }


    subdates = subticks * dates;
    dt = T / subdates;
    sqrt_dt = sqrt(T / subdates);
}

mcam::Model *mcam::instantiate_model(const Param &P)
{
    mcam::Model *mod = NULL;
    string Modelype;
    P.extract("model type", Modelype);

    if (Modelype == "bs")
        mod = new mcam::BlackScholesModel(P);
    else if (Modelype == "dupire")
        mod = new mcam::DupireModel(P);
    else if (Modelype == "heston")
        mod = new mcam::HestonModel(P);
    else
        printf("No valid model found. Aborting...\ n");

    return mod;
}

mcam::Model::~Model()
{
    pnl_vect_free(&spot);
    pnl_vect_free(&dividend);
}


void mcam::Model::print() const
{
    cout << " interest rate : " << r << endl;
    cout << " dividend rate : ";
    pnl_vect_print_asrow(dividend);
    cout << " Number of exercise dates : " << dates << endl;
    cout << " Number of sub ticks : " << subticks << endl;
    cout << " Spot : ";
```

```
    pnl_vect_print_asrow(spot);
}

/**
 * Compute the discount factor
 *
 * @param n is the tick number on the finer grid
 *
 * @return
 */
double mcam::Model::discount(int n) const
{
    return exp(- r * n * dt);
}

PnlMat* mcam::Model::getPath() const
{
    return this->path_m();
}

PnlMat* mcam::Model::getRegressor() const
{
    return this->path_m();
}
```