Help

```
#include <string>
#include "
href../../../../common/math/mcam/src/Martingale_h_src.pdfMartingale.hpp"
#include "
href../../../../common/math/mcam/src/chaos_h_src.pdfchaos.hpp"
#include "
href../../../../common/math/mcam/src/wiener_h_src.pdfwiener.hpp"


mcam::Martingale *mcam::instantiate_martingale(mcam::Model *mod, const Param &P,
{
    mcam::Martingale *O = NULL;
    std::string mart_type;
    P.extract("martingale type", mart_type);

    if (mart_type == "trigo")
        O = new mcam::Trigo(P, mod);
    else if (mart_type == "pol")
        O = new mcam::Pol(P, mod);
    else if (mart_type == "chaos")
        O = new mcam::Chaos(P, mod, prune);
    else
        printf("No valid martingale type found. Aborting...\ n");

    return O;
}

mcam::Martingale::Martingale()
    : size(0), subdates(0), nbFreedom(0)
{
    label = "";
}

mcam::Martingale::~Martingale() { }

void mcam::Martingale::print() const
{
    std::cout << std::endl;
    std::cout << "*******************************" << std::endl;
```

```
    std::cout << "**** Martingale Characteristics ****" << std::endl;
    std::cout << " Martingale type " << label << std::endl;
    std::cout << " Finer grid dates " << subdates << std::endl;
}

/**
 * Run a Monte Carlo algorithm to compute the first two moments of the
 * martingale
 *
 * @param rng a random number generator
 * @param alpha the optimal solution of the minimization problem
 * @param t index of the time step
 * @param iterationsMC number of Monte Carlo samples to use
 * @param[out] one
 * @param[out] two
 */
void mcam::Martingale::martingale_moments(mcam::PnlRng_Workspace &rng, const Pnl
{
    PnlMat *DeltaB;
    DeltaB = pnl_mat_create(subdates, size);
    one = two = 0.;
    /*
     * Monte Carlo loop
     */
    for (int l = 0; l < iterationsMC; l++)
    {
        double tmp;
        pnl_mat_rng_normal(DeltaB, subdates, size, rng());
        computePath(DeltaB, alpha);
        tmp = at(t);
        one += tmp;
        two += tmp * tmp;
    }
    one /= iterationsMC;
    two = two / iterationsMC - one * one;
    pnl_mat_free(&DeltaB);
}
```