Help

```
#include  "
href../../../common/math/ImportanceSampling_jl/src/wrapper_h_src.pdfmath/Importa
using namespace std;

extern "C" {
#include  "
href../../../mod/bns/bns_std/bns_std_h_src.pdfbns_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2015+2) //The "#els
  static int CHK_OPT(MC_Lelong_IS)(void *Opt, void *Mod)
  {
    return NONACTIVE;
  }
  int CALC(MC_Lelong_IS)(void *Opt, void *Mod, PricingMethod *Met)
  {
    return AVAILABLE_IN_FULL_PREMIA;
  }
#else

  int CALC(MC_Lelong_IS)(void *Opt, void *Mod, PricingMethod *Met)
  {
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    NumFunc_1  *p = ptOpt->PayOff.Val.V_NUMFUNC_1;
    double option_type = 0.;
    double r, divid;

    if ((p->Compute) == &Call)
      option_type = 1;
    else if ((p->Compute) == &Put)
      option_type = -1;

    r = log(1. + ptMod->R.Val.V_DOUBLE / 100.);
    divid = log(1. + ptMod->Divid.Val.V_DOUBLE / 100.);


    Param P;
    bool poisson = false, poisson_only = false;
    PnlRng *rng = pnl_rng_create(Met->Par[3].Val.V_ENUM.value);
```

```
pnl_rng_sseed(rng, 0);
std::vector<double> payoff_coeffs(1, option_type);
std::vector<double> kappa(1, ptMod->Lambda.Val.V_PDOUBLE);
std::vector<double> leverage(1, ptMod->Rho.Val.V_PDOUBLE);
std::vector<double> jump_intensity(1, ptMod->Alpha.Val.V_PDOUBLE);
std::vector<double> jump_size(1, ptMod->Beta.Val.V_PDOUBLE);
std::vector<double> spot(1, ptMod->S0.Val.V_PDOUBLE);
std::vector<double> volatility(1, ptMod->Sigma0.Val.V_PDOUBLE);
std::vector<double> dividend_rate(1, divid);


P.insert("model type", T_STRING, string("bns"));
P.insert("option size", T_INT, 1);
P.insert("poisson size", T_INT, 1);
P.insert("jump intensity", T_VECTOR, jump_intensity);
P.insert("kappa", T_VECTOR, kappa);
P.insert("jump size parameter", T_VECTOR, jump_size);
P.insert("leverage parameter", T_VECTOR, leverage);
P.insert("strike", T_DOUBLE, option_type * p->Par[0].Val.V_DOUBLE);
P.insert("spot", T_VECTOR, spot);
P.insert("maturity", T_DOUBLE, ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_D
P.insert("volatility", T_VECTOR, volatility);
P.insert("interest rate", T_DOUBLE, r);
P.insert("dividend rate", T_VECTOR, dividend_rate);
P.insert("correlation", T_DOUBLE, 0.);
P.insert("option type", T_STRING, string("basket"));
P.insert("step size", T_DOUBLE, 0.5);
P.insert("timestep number", T_INT, Met->Par[0].Val.V_PINT);
P.insert("sample number", T_LONG, size_t(Met->Par[1].Val.V_PINT));
P.insert("payoff coefficients", T_VECTOR, payoff_coeffs);

if (Met->Par[2].Val.V_ENUM.value == 1)
  {
    poisson_only = false;
    poisson = false;
  }
else if (Met->Par[2].Val.V_ENUM.value == 2)
  {
    poisson_only = true;
    poisson = true;
  }
```

```
    else if (Met->Par[2].Val.V_ENUM.value == 3)
      {
        poisson_only = false;
        poisson = true;
      }
    MonteCarloWrapper(rng, P, false, false, true, false, false, poisson_only, po

    // Do not delete the vectors inserted in the Param P because they are handle
    pnl_rng_free(&rng);


    return OK;
  }


  static int CHK_OPT(MC_Lelong_IS)(void *Opt, void *Mod)
  {
    if ((strcmp(((Option *)Opt)->Name, "CallEuro") == 0) || (strcmp(((Option *)O
      return OK;

    return  WRONG;
  }

#endif //PremiaCurrentVersion

  static int MET(Init)(PricingMethod *Met, Option *Opt)
  {
    if (Met->init == 0)
      {
Met->HelpFilenameHint = "MC_BNS_IS";
        Met->init = 1;
        Met->Par[0].Val.V_PINT = 50;
        Met->Par[1].Val.V_PINT = 30000;
        Met->Par[2].Val.V_ENUM.members = &PremiaEnumISType;
        Met->Par[2].Val.V_ENUM.value = 1;
        Met->Par[3].Val.V_ENUM.members = &PremiaEnumMCRNGs;
        Met->Par[3].Val.V_ENUM.value = 0;
      }

    return OK;
  }
```

```
  PricingMethod MET(MC_Lelong_IS) =
  {
    "MC_BNS_Lelong_IS",
    {
      {"Nb.of TimeSteps", PINT, {50}, ALLOW, SETABLE},
      {"Nb.of Samples", PINT, {20000}, ALLOW, SETABLE},
      {"Type of IS", ENUM, {0}, ALLOW, SETABLE},
      {"Random Generator", ENUM, {0}, ALLOW, SETABLE},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CALC(MC_Lelong_IS),
    {
      {"Price", DOUBLE, {100}, FORBID},
      {"Std_dev", DOUBLE, {100}, FORBID},
      {" ", PREMIA_NULLTYPE, {0}, FORBID}
    },
    CHK_OPT(MC_Lelong_IS),
    CHK_ok,
    MET(Init)
  };

}
```