Help

```
#include <stdlib.h>
#include  "
href../../../mod/bs1d/bs1d_std/bs1d_std_h_src.pdfbs1d_std.h"
#include "
href../../../common/error_msg_h_src.pdferror_msg.h"

static int LnThirdMoment(int am, double s, NumFunc_1 *p, double t, double r, dou
{
  double  h, mu, u, d, scan, proba, lowerstock, iv, stock;
  double  *P;
  int     i, j, npoints = 2 * N + 1;

  /*Price array*/
  P = malloc(npoints * sizeof(double));
  if (P == NULL)
    return MEMORY_ALLOCATION_FAILURE;

  /*Up and Down factors*/
  h = t / (double)N;
  mu = (r - divid) - .5 * sigma * sigma;
  u = exp(sigma * sqrt(3.*h));
  d = 1. / u;
  scan = u;
  u *= exp(mu * h);
  d *= exp(mu * h);

  /*Discounted Probability*/
  proba = exp(-r * h) / 6.;

  /*Terminal values*/
  lowerstock = s;
  for (i = 0; i < N; i++)
    lowerstock *= d;

  stock = lowerstock;
  for (i = 0; i < npoints; i++)
    {
      iv = (p->Compute)(p->Par, stock);
      P[i] = iv;
```

```
    stock *= scan;
  }


/*Backward Resolution*/
for (i = N; i > 1; i--)
  {
    npoints -= 2;
    lowerstock /= d;
    stock = lowerstock;
    for (j = 0; j < npoints; j++)
      {
        P[j] = proba * (P[j] + 4.*P[j + 1] + P[j + 2]);
        if (am)
          {
            iv = (p->Compute)(p->Par, stock);
            P[j] = MAX(iv, P[j]);
          }
        stock *= scan;
      }

  }
lowerstock /= d;
stock = lowerstock;

/*Delta*/
*ptdelta = (P[2] - P[0]) / (stock * u - stock * d);


/*First time step*/
P[0] = proba * (P[0] + 4.*P[1] + P[2]);
if (am)
  {
    iv = (p->Compute)(p->Par, stock);
    P[0] = MAX(iv, P[0]);
  }

/*Price*/
*ptprice = P[0];

/*Memory desallocation*/
```

```
  free(P);

  return OK;
}

int CALC(TR_LnThirdMoment)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;
  double r, divid;

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

  return  LnThirdMoment(ptOpt->EuOrAm.Val.V_BOOL, ptMod->S0.Val.V_PDOUBLE,
                        ptOpt->PayOff.Val.V_NUMFUNC_1, ptOpt->Maturity.Val.V_DAT
                        r, divid, ptMod->Sigma.Val.V_PDOUBLE, Met->Par[0].Val.V_
}

static int CHK_OPT(TR_LnThirdMoment)(void *Opt, void *Mod)
{
  return  OK;
}

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;

      Met->Par[0].Val.V_INT2 = 100;


    }

  return OK;
}

PricingMethod MET(TR_LnThirdMoment) =
{
  "TR_LnThirdMoment",
```

```
  {{"StepNumber", INT2, {100}, ALLOW}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(TR_LnThirdMoment),
  {{"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID} , {" ", PR
  CHK_OPT(TR_LnThirdMoment),
  CHK_tree,
  MET(Init)
};
```