```
   Help
#include  "dup1d_std.h"

int MOD_OPT(ChkMix)(Option *Opt,Model *Mod)
{
  TYPEOPT* ptOpt=( TYPEOPT*)(Opt->TypeOpt);
  TYPEMOD* ptMod=( TYPEMOD*)(Mod->TypeModel);
  int status=OK;

  if ((ptOpt->Maturity.Val.V_DATE)<=(ptMod->T.Val.V_DATE))
    {
      Fprintf(TOSCREENANDFILE,"Current date greater than
    maturity!{n");
      status+=1;
    };

  return status;
}



extern PricingMethod MET(FD_Implicit);
extern PricingMethod MET(FD_Adaptive);
extern PricingMethod MET(MC_Dupire);

PricingMethod* MOD_OPT(methods)[]={

  &MET(FD_Implicit),
  &MET(FD_Adaptive),
  &MET(MC_Dupire),

  NULL
};

DynamicTest* MOD_OPT(tests)[]={
  NULL
};

Pricing MOD_OPT(pricing)={
  ID_MOD_OPT,
  MOD_OPT(methods),
  MOD_OPT(tests),
  MOD_OPT(ChkMix)
};



/* utilities shared */

/* Local Volatility Examples Sigma(t,x) */

double MOD_OPT(lib_volatility)(double t, double x,int sigma
    _type)
{
  double val;

  if(sigma_type==0)
    {
      val=15./x;
    }
  else /*if(sigma_type==1)*/
    {
      val=0.01+0.1*exp(-x/100)+0.01*t;
    }

  if (val>=1.)  val=1.;
  if (val<=0.01) val=0.01;

  return val;
}

/* First Order Derivatives Sigma(t,x) for Adaptive Method*/

double MOD_OPT(lib_volatility_x)(double t, double x,int si
    gma_type)
{
  double val;
  if(sigma_type==0)
    {
      val=-15./SQR(x);
    }
  else /*if(sigma_type==1)*/
```

```
    {
      val=-0.1/100.*exp(-x/100.);
    }
  return val;
}

/* Second Order Derivatives Sigma(t,x) for Adaptive Method
    */

double MOD_OPT(lib_volatility_xx)(double t, double x,int si
    gma_type)
{
  double val;
  if(sigma_type==0)
    {
      val=30./CUB(x);
    }
  else /*if(sigma_type==1)*/
    {
      val=0.1/(100.*100.)*exp(-x/100.);
    }
  return val;
}
```

References