```
   Help
#include "bs1d_lim.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;
    };
  if ((ptOpt->DownOrUp).Val.V_BOOL==DOWN)
    {

      if ( ((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((pt
    Opt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE)>ptMod->
    S0.Val.V_PDOUBLE && (ptOpt->Parisian).Val.V_BOOL==WRONG)
        {

      Fprintf(TOSCREENANDFILE,"Limit Down greater than spo
    t!{n");
      status+=1;
        };


    }
  if ((ptOpt->DownOrUp).Val.V_BOOL==UP)
    {
      if ( ((ptOpt->Limit.Val.V_NUMFUNC_1)->Compute)((pt
    Opt->Limit.Val.V_NUMFUNC_1)->Par,ptMod->T.Val.V_DATE)<ptMod->
    S0.Val.V_PDOUBLE && (ptOpt->Parisian).Val.V_BOOL==WRONG)
        {
      Fprintf(TOSCREENANDFILE,"Limit Up lower than spot!{n"
    );
      status+=1;
        };
    }
  return status;
}

extern PricingMethod MET(TR_Ritchken_UpOut);
extern PricingMethod MET(TR_Ritchken_UpIn);
extern PricingMethod MET(CF_CallDownOut);
extern PricingMethod MET(CF_CallUpIn);
extern PricingMethod MET(CF_CallUpOut);
extern PricingMethod MET(CF_PutDownIn);
extern PricingMethod MET(CF_PutDownOut);
extern PricingMethod MET(CF_PutUpIn);
extern PricingMethod MET(CF_PutUpOut);
extern PricingMethod MET(FD_Psor_DownOut);
extern PricingMethod MET(FD_Psor_UpOut);
extern PricingMethod MET(FD_Psor_DownIn);
extern PricingMethod MET(FD_Psor_UpIn);
extern PricingMethod MET(FD_Cryer_DownOut);
extern PricingMethod MET(FD_Cryer_UpOut);
extern PricingMethod MET(FD_Cryer_DownIn);
extern PricingMethod MET(FD_Cryer_UpIn);
extern PricingMethod MET(FD_Gauss_DownIn);
extern PricingMethod MET(FD_Gauss_DownOut);
extern PricingMethod MET(FD_Gauss_UpIn);
extern PricingMethod MET(FD_Fem_Out);
extern PricingMethod MET(FD_Gauss_UpOut);
extern PricingMethod MET(TR_Ritchken_DownOut);
extern PricingMethod MET(TR_Ritchken_DownIn);
extern PricingMethod MET(TR_DermanKani);
extern PricingMethod MET(TR_RogersStapleton_DownOut);
extern PricingMethod MET(TR_RogersStapleton_UpOut);
extern PricingMethod MET(CF_CallDownIn);
extern PricingMethod MET(MC_OutBaldi);
extern PricingMethod MET(MC_InBaldi);
extern PricingMethod MET(MC_ParisianOut);
extern PricingMethod MET(MC_ParisianIn);
extern PricingMethod MET(AP_LaplaceParisian);

PricingMethod *MOD_OPT(methods)[]={
  &MET(CF_CallDownIn),
  &MET(CF_CallDownOut),
  &MET(CF_CallUpIn),
  &MET(CF_CallUpOut),
  &MET(CF_PutDownIn),
  &MET(CF_PutDownOut),
  &MET(CF_PutUpIn),
  &MET(CF_PutUpOut),
  &MET(FD_Psor_DownOut),
  &MET(FD_Psor_UpOut),
  &MET(FD_Psor_DownIn),
  &MET(FD_Psor_UpIn),
  &MET(FD_Cryer_DownOut),
  &MET(FD_Cryer_DownIn),
  &MET(FD_Cryer_UpOut),
  &MET(FD_Cryer_UpIn),
  &MET(FD_Gauss_DownIn),
  &MET(FD_Gauss_DownOut),
  &MET(FD_Gauss_UpIn),
  &MET(FD_Gauss_UpOut),
  &MET(FD_Fem_Out),
  &MET(TR_Ritchken_UpOut),
  &MET(TR_Ritchken_UpIn),
  &MET(TR_Ritchken_DownOut),
  &MET(TR_Ritchken_DownIn),
  &MET(TR_DermanKani),
  &MET(TR_RogersStapleton_DownOut),
  &MET(TR_RogersStapleton_UpOut),
  &MET(MC_OutBaldi),
  &MET(MC_InBaldi),
  &MET(MC_ParisianOut),
  &MET(MC_ParisianIn),
  &MET(AP_LaplaceParisian),
  NULL
};

extern DynamicTest MOD_OPT(test);
DynamicTest* MOD_OPT(tests)[]={
  &MOD_OPT(test),
  NULL
};

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

References