```
   Help
#include "mer1d_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)
          {

    Fprintf(TOSCREENANDFILE,"Limit Down greater than spot!
    {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)
          {
    Fprintf(TOSCREENANDFILE,"Limit Up lower than spot!{n")
    ;
    status+=1;
          };
```

```
    }
  return status;
}


extern PricingMethod MET(FD_ImpExpDownOut);
extern PricingMethod MET(FD_ImpExpUpOut);

PricingMethod *MOD_OPT(methods)[]={

  &MET(FD_ImpExpDownOut),
  &MET(FD_ImpExpUpOut),
  NULL
};
DynamicTest* MOD_OPT(tests)[]={
  NULL
};

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

References