Help

```
#include  "bs1d_std.h"

static int Euler(int am,double s,NumFunc_1 *p,
    double t,double r,double divid,double sigma,int N,
    double *ptprice,double *ptdelta)
    {
    double  h,mu,u,d,scan,proba,lowerstock,iv,sto
    ck;
    double  *P;
    int     i,j;

    /*Price array*/
    P=(double *)malloc((N+1)*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(h));d=1./u;
    u*=exp(mu*h);
    d*=exp(mu*h);
    scan=u/d;

    /*Discounted Probability*/
    proba=.5*exp(-r*h);

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

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

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

        }

    lowerstock/=d;
    stock=lowerstock;

    /*Delta*/
    *ptdelta=(P[1]-P[0])/(stock*u-stock*d);

    /*First time step*/
    P[0]=proba*(P[0]+P[1]);
    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 CHK_OPT(TR_Euler)(void *Opt, void *Mod)
```

```
    {
    Option* ptOpt=(Option*)Opt;
    TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

    return  OK;
    }

int CALC(TR_Euler)(void *Opt,void *Mod,Pricing
    Method *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 Euler(ptOpt->EuOrAm.Val.V_BOOL,ptMod->
    S0.Val.V_PDOUBLE,ptOpt->PayOff.Val.V_NUMFUNC_1,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.
    V_DATE,r,divid,ptMod->Sigma.Val.V_PDOUBLE,
        Met->Par[0].Val.V_INT,&(Met->Res[0].Val.
    V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
    }

static int MET(Init)(PricingMethod *Met)
    {
    static int first=1;

    if (first)
        {
        Met->Par[0].Val.V_INT2=100;

        first=0;
        }

    return OK;
    }

PricingMethod MET(TR_Euler)=
    {
```

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

References