Help
```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

#include"lmm_header.h"
#include "pnl/pnl_vector.h"

int mallocLibor(Libor **ptLib, int numOfMat, double tenor
    Val,double l0)
{
    int i;

    Libor *pt;
    pt=(Libor*)malloc(sizeof(Libor));

    pt->numberOfMaturities=numOfMat;
    pt->tenor=tenorVal;

    pt->libor = pnl_vect_create(numOfMat);
    pt->maturity = pnl_vect_create(numOfMat);

    for (i=0;i<numOfMat;i++)
    {
        LET(pt->maturity, i) = i*tenorVal;
        LET(pt->libor, i) = l0;
    }

    *ptLib = pt;

    return(EXIT_SUCCESS);
}

void freeLibor(Libor **ptLib)
{
    pnl_vect_free(&((*ptLib)->libor));
    pnl_vect_free(&((*ptLib)->maturity));
    free(*ptLib);
    ptLib=NULL;
}
```

```
int initLibor(Libor *ptLib,double l0)
{
    int i;

    for (i=0; i<ptLib->numberOfMaturities; i++)
    {
        LET(ptLib->libor, i)=l0;
    }
    return(1);
};

void Libor_To_ZeroCoupon(Libor* ptLib, PnlVect* zc) //
    Compute P(0, Ti) i=0:N
{
    int i, N;

    N = ptLib->numberOfMaturities;

    pnl_vect_resize(zc, N+1);

    LET(zc, 0) = 1;

    for (i=0; i<N; i++)
    {
        LET(zc, i+1) = GET(zc, i)/(1.+(ptLib->tenor)*GET(pt
    Lib->libor,i));
    }
}

double Sum_ZC(Libor* ptLib, int i_first, int i_last) //
    Compute "sum P(0, T_i)" for "i" from "i_first" to "i_last".
{
    int i;
    double zc, sum_zc;

    zc = 1.;
    for (i=0; i<i_first; i++)
    {
        zc /= (1.+(ptLib->tenor)*GET(ptLib->libor,i));
    }
    sum_zc = zc; // P(0, T(i_first))
    for (i=i_first; i<i_last; i++)
    {
        zc /= (1.+(ptLib->tenor)*GET(ptLib->libor,i)); //
    P(0, T(i+1))
        sum_zc += zc;
    }

    return sum_zc;
}


int readLiborFromFile(Libor **ptLib, char *fileName)
{
    /*lie les donnees des libors initiaux et leurs maturité
    s, les donnees lues prevalent sur celles donnees dans 'ini
    tLiborList()'*/
    int i,n;
    char ligne[20];
    char* pligne;
    double t, l, Tprev, delta, deltaprev;
    FILE *datas;
    double *L;
    double *T;

    datas=fopen(fileName, "r");

    if (datas==NULL)
    {
        printf("Le FICHIER N'A PU ETRE OUVERT. VERIFIER LE
    CHEMIN{n");
        exit(1);
    }

    n=0;
    Tprev=0;
    deltaprev=0;

    pligne=ligne;
    T=(double *)malloc(100*sizeof(double));
```

```
L=(double *)malloc(100*sizeof(double));
/* printf("OUVERTURE{n");*/


while (1)
{
    pligne=fgets(ligne, sizeof(ligne), datas);
    if (pligne==NULL)
        break;
    else
    {
        sscanf(ligne, "%lf t=%lf", &t, &l);

        T[n]=t;
        L[n]=t;

        delta=t-Tprev;
        Tprev=t;
        if (delta!=deltaprev && n>0)
        {
            printf("WARNING, NO CONSTANT TENOR IN
LIBOR LIST!{n");
        }
        deltaprev=delta;

        n++;
    }
}
fclose(datas);


(*ptLib)->maturity=pnl_vect_create(n);
(*ptLib)->libor =  pnl_vect_create(n);

for (i=0;i<n;i++)
{
    LET((*ptLib)->maturity,i)=T[i];
    LET((*ptLib)->libor,i)=L[i];
}
(*ptLib)->numberOfMaturities=n;
(*ptLib)->tenor=T[2]-T[1];
```

```
    free(T);
    free(L);

    return(1);
}


int putLiborToZero(Libor *ptLib, int index)
{
    LET(ptLib->libor,index)=0.0;
    return (1);
};


int copyLibor(Libor *ptLibSrc , Libor *ptLibDest )
{
    int i ;

    ptLibDest->numberOfMaturities=ptLibSrc->numberOfMaturit
    ies;
    ptLibDest->tenor=ptLibSrc->tenor;

    for (i=0; i<ptLibSrc->numberOfMaturities; i++)
    {
        LET(ptLibDest->libor,i)=GET(ptLibSrc->libor,i);
        LET(ptLibDest->maturity,i)=GET(ptLibSrc->maturity,
    i);
    }
    return (1);
};


int printLibor(Libor *ptLib)
{
    int i;
    for (i=0;i<ptLib->numberOfMaturities;i++)
    {
        printf("%lf %lf {n",GET(ptLib->maturity,i), GET(pt
    Lib->libor,i));
    }
    printf("{n");
    return (1);
}


double computeSwapRate(Libor* ptLib, int o, int s,int m )
{
    // compute   (B(T_o,T_s)-B(T_o,T_m))/( sum_{i=s+1}^{m}
    {tau B(T_o,T_i) )=S(T_o,T_s,T_m) the forward swap rate
    int k,l;
    double val=1.;
    double sum=0.0;
    double vald;

    for (k=s+1;k<=m;k++)
    {
        val=1.;
        for (l=o ; l<k ; l++)
        {
            val*=1./(1.+ ptLib->tenor*GET(ptLib->libor,l));
        }
        sum+=ptLib->tenor* val;
    }

    val=1.;
    for (k=o;k<m;k++)
    {
        val*=1./(1.+ ptLib->tenor*GET(ptLib->libor,k));
    }

    if (o!=s)
    {
        vald=1.;
        for (k=o;k<=(s-1);k++)
        {
            vald*=1./(1.+ ptLib->tenor*GET(ptLib->libor,k))
    ;
        }
        return((vald-val)/sum);
    }
    else
    {
        return((1.-val)/sum);
    }
}

double computeSwapPrice(Libor* ptLib, Swaption* ptSwp,int
    o, int s,int m )
{
    // compute   B(T_o,T_s)-B(T_o,T_m)-K* sum_{i=s+1}^{m} {
    tau B(T_o,T_i)
    //price at time T_o of a swap on T_s,....T_m
    int k,l;
    double val=1.;
    double sum=0.0;
    double vald;
    double price;

    //sum_{k=s+1}^{m} {tau B(T_o,T_k)
    for (k=s+1;k<=m;k++)
    {
        val=1.;
        //B(T_o,T_k)
        for (l=o ; l<k ; l++)
        {
            val*=1./(1.+ ptLib->tenor*GET(ptLib->libor,l));
        }
        sum+=ptLib->tenor* val;
    }

    val=1.;
    //B(T_o,T_m)
    for (k=o;k<m;k++)
    {
        val*=1./(1.+ ptLib->tenor*GET(ptLib->libor,k));
    }

    if (o!=s)
    {
        vald=1.;
        //B(T_o,T_s)
        for (k=o;k<s;k++)
        {
            vald*=1./(1.+ ptLib->tenor*GET(ptLib->libor,k))
    ;
        }
        price=vald-val-ptSwp->strike*sum;
    }
    else //B(T_o,T_s)=1
    {
        price=1.-val-ptSwp->strike*sum;
    }
    return(price);
}

double computeZeroCouponSum(Libor* ptLib, int o,int s,int
    m )
{
    // compute   sum_{i=s}^{m} {tau B(T_o,T_i)
    int j,k;
    double val=1.;
    double sum=0.0;

    for (j=s;j<=m;j++)
    {
        val=1.;
        for (k=o;k<j;k++)
        {
            val*=1./(1.+ ptLib->tenor*GET(ptLib->libor,k));
        }
        sum+=ptLib->tenor* val;
    }

    return( sum );
}


double computeZeroCoupon(Libor* ptLib, int o,int s)
{
    // compute B(T_o,T_s)
    if (o==s)
    {
        return(1);
    }
    else
    {
        return(computeZeroCouponSum( ptLib, o, s, s )/pt
    Lib->tenor);
    }
}


#endif //PremiaCurrentVersion
```

References