Help

```
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2007+2) //The "#els
#else

#ifndef HKTREE_H
#define HKTREE_H

#include <stdio.h>
#include <stdlib.h>
#include <
href../../../common/math/cdo/cdo_math_h_src.pdfmath.h>

struct Tree
{
  double *t;          /*Time step grid, from t[0] to t[Ngrid] */
  double Tf;          /*Final time of the tree, dt=Tf/Ngrid */
  int Ngrid;          /*Number of time steps in the Tree */
  double **Payoffunc; /*Vector Payoff for the tree (see the function initPayoff1

  double P_T;         /*The value of the Z-C bond P(0,T)*/
  double **pLRij;     /*The value of the short rates in the tree*/
  double **pLQij;     /*The value of the Options or other things (depend on Payo
  double **pLPDo;     /*Transition proba. in the trinomial tree for the lower po
  double **pLPMi;     /*Transition proba. in the trinomial tree for the middle p
  double **pLPUp;     /*Transition proba. in the trinomial tree for the upper po
  int   **pLRef;      /*Reference index for the midle point of the next time ste
  int    *TSize;      /*Size of the scale rate at given time step.*/
};



void SetHKtree(struct Tree *Meth, double a0, double sigma0);
// constructs a tree for the HK-process (x_t) given by: dx_t = sigma0*exp(a0*t)


void SetTimegrid(struct Tree *Meth, double Tf, int Ngrid);
// Allocate the uniform time grid t[i]=i*Tf/Ngrid for i=0,...,Ngrid


int indiceTime(struct Tree *Meth, double s);
```

```
// Return the i such that Meth->t[i] is closest possible to s


void Computepayoff1(struct Tree *Meth, double s);
// initialization: Meth->pLQij[n] = Meth->Payoffunc[n], where n=indiceTime(Meth,
// then AMERICAN backward iteration of Meth->pLQij[i] from i=n-1 to i=0
// here AMERICAN means: taking the max with Meth->Payoffunc[i]
// IMPORTANT: Meth->plQij simulates hence the DISCOUNTED value of an american op
// whose DISCOUNTED payoff is given by Meth->Payoffunc and whose maturity is s !


void initPayoff1(struct Tree *Meth, double T0);
// Allocates Payoffunc[0...n], where n=indiceTime(Meth, T0); sets all the Payoff
// and all the Payoffunc[i][j] = 0, where i<n (T0 must be <= Tf, the final time


void DeletePayoff1(struct Tree *Meth, double T0);
// Deletes Payoffunc[0...n], where n=indiceTime(Meth, T[0])


int AddTime(struct Tree *Meth, double T);
// adds (if necessary) T in Meth->t and returns the i such that Meth->t[i]=T

double OPTION(struct Tree *Meth);
// returns Meth->pLQij[0][1]

double OPTIONr(struct Tree *Meth, double r, double s);

int DeleteTree(struct Tree *Meth);
// Delete all the allocated memory of the tree

#endif
#endif //PremiaCurrentVersion
```