```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

#ifndef functions_h
#define functions_h

#include "common.h"
#include<stdlib.h>
#include<iostream>
#include <cmath>
#include <vector>

using std::cout;
using std::endl;

//psi=P[ZT>=a,maxZs>=b s dans[0 T]] ou Z levy poisson compo
    se long double exponentielle
long double psiB(long double *x,const long double &T);
//fonction psi=P[ZT>=a] ou Z levy poisson compose long
    double exponentielle
long double psiVN(long double *,int nb=20);

//deriviï£¡de psi=P[ZT>=a] % ïï£¡a
long double dpsiVN(long double *,int nb=20);
std::vector<long double> psiVNb(long double *,int nb=20);
//prix du put loockback floating strike
long double PLB(long double *,const long double &);
//prix du call loockback floating strike
long double CLB(long double *,const long double &);
//delta du put loockback floating strike
long double dPLB(long double *,const long double &);
//delta du call loockback floating strike
long double dCLB(long double *,const long double &);

long double psiM(long double *,const long double &);
long double rebateproba(long double *,const long double &,
    const long double &);
long double psiMA(long double *,const long double &);
```

```
long double psiMB(long double *,const long double &);
class function
{
 public:
 long double *param;
 function();
 function(long double*);
 virtual ~function(){};
 virtual const long double f(const long double &)const{ret
    urn 0;};
 virtual const long double Df(const long double &)const{ret
    urn 0;};
};
inline function::function()
{
 param=NULL;
};
inline function::function(long double* v)
{
 param=v;
};
//algorithme d'inversion de la transformÃŕÂ£Â¡ de Laplace
class inverselaplace
{
 public:
 function *LF;
 inverselaplace(){};
 inverselaplace(function &F){LF=&F;};
 const long double InvLF(const long double& T)const;
};
class solver
{
 public:
 function *F;
 solver(){};
 solver(function &G){F=&G;};
 virtual ~ solver(){};
 virtual std::vector<long double> racine()const{return std:
    :vector<long double>();};
};
```

```
class newton : public solver
{
 public:
 long double start;
 long double accuracy;
 newton(function &G,long double x0, long double eps):
    solver(G){accuracy=eps; start=x0;};
 virtual std::vector<long double> racine()const;
};
class secante : public solver
{
 public:
 long double start0;
 long double start1;
 long double accuracy;
 secante(function &G,long double x0, long double x1, long
    double eps):solver(G){accuracy=eps; start0=x0;start1=x1;};
 virtual std::vector<long double> racine()const;
};
class dichotomie : public solver
{
 public:
 long double a;
 long double b;
 long double accuracy;
 dichotomie(function &G,long double x0, long double x1, lon
    g double eps):solver(G){accuracy=eps; a=x0;b=x1;};
 virtual std::vector<long double> racine()const;
};
//fonction exposant caracteristique d'uN levy poisson compo
    si£¡double exponenetielle G(x)-alpha
class KCE : public function
{
 public:
 KCE(long double* param):function(param){};
 virtual const long double f(const long double &x)const;
 virtual const long double Df(const long double &x)const;
};
class LPLB : public function
{
 public:
```

```
 LPLB(long double *param):function(param){};
 //Transformï£¡ de Laplace du prix du put loockback floatin
    g strike
 virtual const long double f(const long double& alpha) cons
    t;
};
class DLPLB : public function
{
 public:
 DLPLB(long double *param):function(param){};
 //Transformï£¡ de Laplace de la derivï£¡ % ï£¡S0 du prix
    du put loockback floating strike
 virtual const long double f(const long double& alpha) cons
    t;
};

class LCLB : public function
{
 public:
 LCLB(long double *param):function(param){};
 //Transformï£¡ de Laplace du prix du call loockback floa
    ting strike
 virtual const long double f(const long double& alpha) cons
    t;
};
class DLCLB : public function
{
 public:
 DLCLB(long double *param):function(param){};
 //Transformï£¡ de Laplace de la derivï£¡ % ï£¡S0 du prix
    du call loockback floating strike
 virtual const long double f(const long double& alpha) cons
    t;
};

class LPsiM : public function
{
 public:
 LPsiM(long double *param):function(param){};
 //transformï£¡ de Laplace de psi=P[maxZs>=b s dans[0 T]]
    ou Z levy poisson compose double exponentielle
```

```
 virtual const long double f(const long double& alpha)cons
    t;
};
class LdPsiM : public function
{
 public:
 LdPsiM(long double *param):function(param){};
 virtual const long double f(const long double& alpha)cons
    t;
};
class LPsiMA : public function
{
 public:
 LPsiMA(long double *param):function(param){};
 virtual const long double f(const long double& alpha)cons
    t;
};
class LPsiMB : public function
{
 public:
 LPsiMB(long double *param):function(param){};
 virtual const long double f(const long double& alpha)cons
    t;
};
class amer_eq : public function
{
 public:
 amer_eq(long double *param):function(param){};
 //fonction dont la racine correspond au prix critique du
    put americain a horizon fini
 virtual const long double f(const long double& v) const;
};

#endif

#endif //PremiaCurrentVersion
```

# References