```
   Help
#ifndef _PREMIA_LIST_H
#define _PREMIA_LIST_H


#ifdef __cplusplus
extern "C" {
#endif /* __cplusplus */

#include <stdlib.h>
#include <stdio.h>
#include "pnl/pnl_vector.h"

typedef struct PremiaContains
{
  int index;
  double value;
} PremiaContains;


extern PremiaContains *premia_contains_create(const int ind
    , double Val);
extern PremiaContains *premia_contains_clone(int ind,
    double Val);
extern void premia_contains_fprint(FILE *fic, Premia    Contains *C);
extern void  premia_contains_add(PremiaContains *C, const
    PremiaContains *C2);
extern int  premia_contains_less(const PremiaContains *C1,
    const PremiaContains *C2);
extern int  premia_contains_equal(const PremiaContains *C1,
     const PremiaContains *C2);
extern PremiaContains *premia_contains_copy(const Premia    Contains *C2);
extern void premia_contains_free(PremiaContains **C);

typedef struct _PremiaNode PremiaNode;

struct _PremiaNode
{
  PremiaNode  *previous;
  PremiaNode  *next;
  PremiaContains *obj;
```

```
};

typedef struct PremiaSortList
{
  int size; /*!< size of the List */
  PremiaNode *first;
  PremiaNode *last;
  PremiaNode *current;
} PremiaSortList;

extern PremiaSortList *premia_sort_list_create();
extern void premia_sort_list_free(PremiaSortList **List);
extern int premia_sort_list_find(PremiaSortList *List, Prem
    iaNode **current, int Key, double Val);
extern int premia_sort_list_find_dicho(PremiaSortList *Lis
    t, PremiaNode **current, int Key, double Val);
extern void premia_sort_list_add(PremiaSortList *List, cons
    t PremiaContains *Val);
extern void premia_sort_list_add_dicho(PremiaSortList *Lis
    t, const PremiaContains *Val);
extern void premia_sort_list_print(const PremiaSortList *
    List);



typedef struct PremiaSparsePoint
{
  PnlVectInt *index;
  int value;
} PremiaSparsePoint;

extern PremiaSparsePoint *premia_sparse_point_create(const
    PnlVectInt *ind, int Val);
extern PremiaSparsePoint *premia_sparse_point_clone(PnlVec
    tInt *ind, int val);
extern void premia_sparse_point_fprint(FILE *fic, Premia
    SparsePoint *C);
extern void  premia_sparse_point_add(PremiaSparsePoint *C,
    const PremiaSparsePoint *C2);
extern int  premia_sparse_point_less(const PremiaSparsePoi
    nt *C1, const PremiaSparsePoint *C2);
```

```
extern int  premia_sparse_point_equal(const PremiaSparsePoi
    nt *C1, const PremiaSparsePoint *C2);
extern PremiaSparsePoint *premia_sparse_point_copy(const
    PremiaSparsePoint *C2);
extern void premia_sparse_point_free(PremiaSparsePoint **C)
    ;


typedef struct _PremiaNodeSparsePoint PremiaNodeSparsePoint
    ;

struct _PremiaNodeSparsePoint
{
  PremiaNodeSparsePoint  *previous;
  PremiaNodeSparsePoint  *next;
  PremiaSparsePoint *obj;
};

extern  void premia_node_sparse_point_free(PremiaNodeSpars
    ePoint **N);

typedef struct PremiaSortListSparsePoint
{
  int size; //!< size of the List
  PremiaNodeSparsePoint *first;
  PremiaNodeSparsePoint *last;
  PremiaNodeSparsePoint *current;
} PremiaSortListSparsePoint;

extern PremiaSortListSparsePoint *premia_sort_list_    sparse_point_create();
extern void premia_sort_list_sparse_point_free(PremiaSortL
    istSparsePoint **List);
extern int premia_sort_list_sparse_point_find(PremiaSortLis
    tSparsePoint *List, PremiaNodeSparsePoint **current, PnlVec
    tInt *Key, int Val);
extern int premia_sort_list_sparse_point_find_dicho(Premia
    SortListSparsePoint *List, PremiaNodeSparsePoint **current,
    PnlVectInt *Key, int Val);
extern void premia_sort_list_sparse_point_add(PremiaSortLis
    tSparsePoint *List, const PremiaSparsePoint *Val);
extern void premia_sort_list_sparse_point_add_dicho(Premia
```

```
    SortListSparsePoint *List, const PremiaSparsePoint *Val);
extern void premia_sort_list_sparse_point_print(const Prem
    iaSortListSparsePoint *List);

#ifdef __cplusplus
}
#endif /* __cplusplus */


#endif /* _PREMIA_LIST_H */
```

References