map.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include "string.h"
#include "map.h"
```

Functions

static int	**compareCities** (void *s1, void *s2)
static int	**compareCitiesDistance** (void *s1, void *s2)
static void	**displayCity** (void *s)
static int	**compareNeighbours** (void *s1, void *s2)
static void	**displayNeighbour** (void *s)
static int	**get_heuristic_cost** (**City** *startCity, **City** *goalCity)
static void	**displaySearchResults** (**City** *currCity)
static **City** *	**getLowestfValueCity** (**List** *openList)
List *	**parseMapFile** (char *filename)
status	**findPath** (**List** *cityList, char *srcCity, char *destCity)
City *	**findCity** (**List** *cityList, char *name)
City *	**createCity** ()
Neighbour *	**createNeighbour** ()
void	**destroyCities** (**List** *listCities)
void	**destroyCity** (**City** **city)
void	**destroyNeighbour** (**Neighbour** **neighbour)

Function Documentation

static int compareCities (void * s1,
void * s2
) static

static int compareCitiesDistance (void * s1,
void * s2
) static

```
static int compareNeighbours ( void *  s1,
                               void *  s2
                             )                                    static
```

City* createCity ( )

Empty **City** creation by dynamic memory allocation.

Returns

a new (empty) city if memory allocation OK

0 otherwise

Neighbour* createNeighbour ( )

Empty **Neighbour** creation by dynamic memory allocation.

Returns

a new (empty) **Neighbour** if memory allocation OK

0 otherwise

void destroyCities (List *)

destroy the list of Cities by deallocation of the the used memory.

Parameters

void destroyCity (City **)

destroy the **City** by deallocation of the the used memory.

Parameters

void destroyNeighbour (Neighbour **)

destroy the City by deallocation of the the used memory.

Parameters

static void displayCity (void * s) static

static void displayNeighbour (void * s) static

static void displaySearchResults (City * currCity) static

City* findCity (List * ,
char *
)

Find City details from list bu using comparison function provided during list creation.

Parameters

status findPath (List * ,
char * ,
char *
)

Finds shortest path between two cities using A Star algorithm and display using function provided during list creation .

Parameters

if successor city is on the OPEN list but the existing

if successor city is on the CLOSED list but the existing

static int get_heuristic_cost (City * startCity,
City * goalCity
) static

static City* getLowestfValueCity (List * openList) static

List* parseMapFile (char *)

Function loads the Map details about cities and connected neighboring cities to a **List** .

Parameters