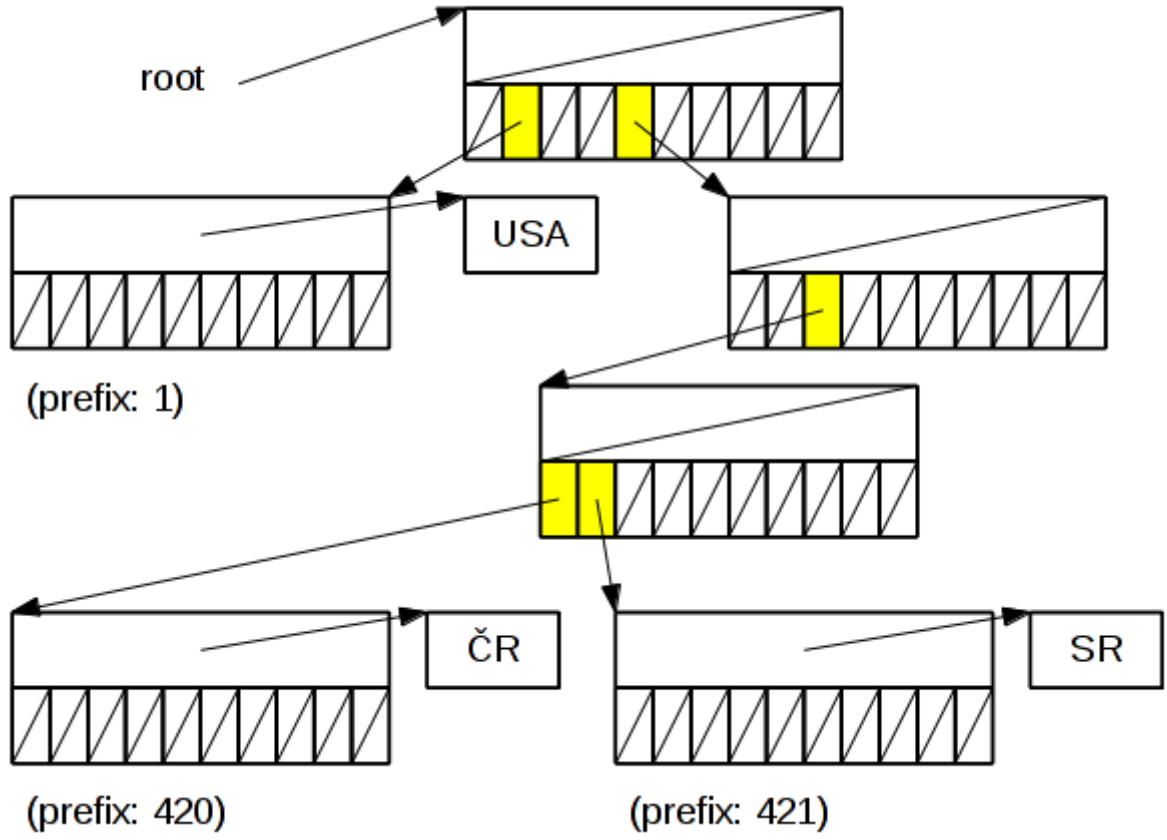
Telefonní čísla

Termín odevzdání:	**16.12.2012 23:59:59**
Hodnocení:	**5.5000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	2 / 10 Volné pokusy + 20 Penalizované pokusy (-2 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je vytvořit sadu funkcí (ne celý program, sadu funkcí), která bude simulovat práci s prefixovým stromem (trie).

Předpokládáme, že chceme naceňovat telefonní hovory. Cena závisí na cíli volání. Známe databázi předčíslí (prefixů) a jim odpovídající pojmenování. Například prefix 420 znamená ČR, 421 SR, 1 USA, ... Prefixy mají různou délku, navíc mohou být dále vnitřně strukturované. Například prefix 420 znamená ČR, prefix 420800 budou zelené linky v ČR, 42073 bude T-Mobile ČR, ...

Pro reprezentaci takovéto hierarchické struktury se hodí strom organizovaný jako trie. Strom bude 10-ární, pro každou cifru zápisu tel. čísla bude mít jednu úroveň, pořadí synovských uzlů bude odpovídat cifrám 0 až 9. Strukturu zachycuje obrázek.



Vaším Úkolem je realizovat sadu funkcí, které budou umět trie rozšiřovat, promazávat a vyhledávat v něm. Požadované funkce mají následující rozhraní:

```
#ifndef __PROGTEST__
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define TELCO_NUMBERS 10
typedef struct TNode
 {
   char          * m_Dest;
   struct TNode  * m_Child[TELCO_NUMBERS];
 } TNODE;
```

```
#endif /* __PROGTEST__ */

void delTree ( TNODE * root )
 {
   /* todo */
 }
int addDest ( TNODE ** root, const char * prefix, const char * dest )
 {
   /* todo */
 }
int delDest ( TNODE ** root, const char * prefix )
 {
   /* todo */
 }
const char * search ( TNODE * root, const char * number )
 {
   /* todo */
 }
#ifndef __PROGTEST__
int main ( int argc, char * argv [] )
 {
   /* tests */
   return 0;
 }
#endif /* __PROGTEST__ */
```

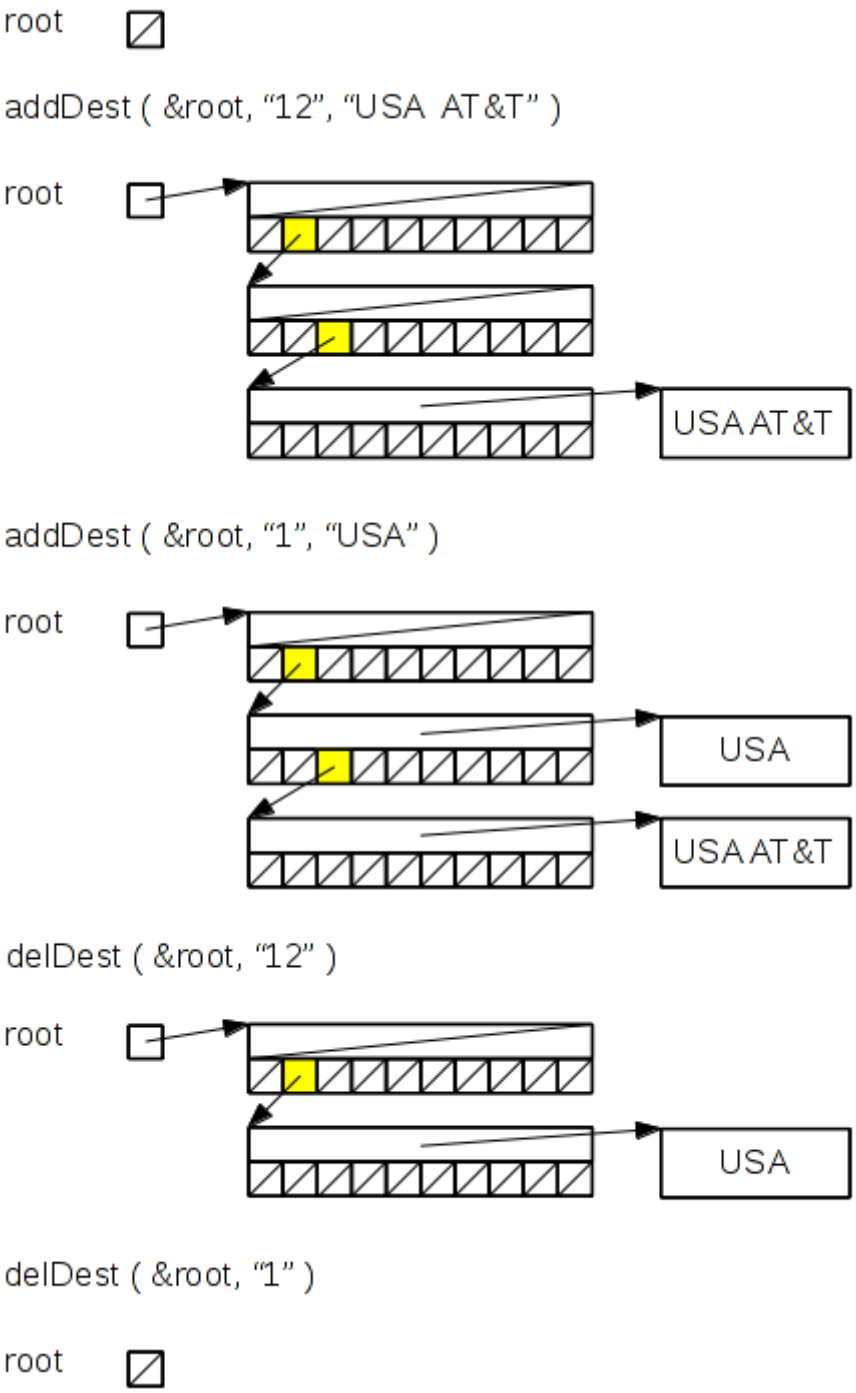
TNODE
: je struktura reprezentující jeden uzel stromu. Má složky `m_Dest`, která je ukazatelem na řetězec popisujícím název cíle (nebo NULL) a pole 10 odkazů na potomky pro pokračování stromu.

addDest
: je funkce, které do stávajícího trie přidá požadovaný prefix a jemu odpovídající cíl volání. Pokud daný prefix ve stromě již existuje, funkce pouze změní jemu asociovaný cíl. Pokud prefix neexistuje, funkce do stromu doplní odpovídající uzly. Funkce vrací úspěch (návratová hodnota 1) nebo neúspěch (návratová hodnota 0). Neúspěchem volání skončí pokud zadaný prefix tel. čísla obsahuje jiné znaky než cifry 0 až 9, v takovém případě funkce ponechá strom beze změn.

delDest
: funkce odstraní ze stromu zadaný prefix a jemu asociovaný cíl volání. Pokud odkazovaný prefix ve stromu neexistuje nebo pokud zadaný prefix obsahuje nečíselné hodnoty, funkce vrací neúspěch (0) a strom ponechá beze změn. Pokud funkce úspěšně odstraní zadaný cíl volání, vrací hodnotu 1. Funkce z mazaného prefixu odstraní řetězec jména cíle (nahradí jej hodnotou NULL) a dále vymaže i všechny nepotřebné uzly, které ve stromu nemusí být (nemají žádnou funkci). V extrémním případě funkce může vymazat i celý strom - pokud odstraňuje poslední cíl volání ve stromu obsažený.

search
: funkce vyhledá ve stromu jméno cíle, které odpovídá zadanému číslu. Návratovou hodnotou je nalezený řetězec - název cíle, případně hodnota NULL. NULL je vrácen pro dotazované číslo, které obsahovalo nečíselné znaky nebo pro číslo, pro které není definovaný cíl. Pozor: strom je prefixový, nemusí obsahovat celé zadávané číslo. Pokud by strom obsahoval jediný cíl - prefix 1 (USA), tedy strom bude tvořen dvojicí uzlů, bude volání search vracet pro číslo 123456 hodnotu "USA", ale např. pro číslo 23456 hodnotu NULL.

delTree
: funkce uvolní veškeré paměťové bloky, které byly použité pro zadaný strom.



Odevzdávejte zdrojový soubor s implementací požadovaných funkcí. Odevzdávaný soubor musí obsahovat implementaci všech požadovaných funkcí s rozhraním (parametry), které odpovídají předpisu nahoře. Dále ve zdrojovém souboru musí být další Vaše pomocné funkce, které z požadovaných funkcí voláte. V odevzdávaném zdrojovém souboru by naopak neměly být zbytečnosti (pozůstatky vývoje a ladění), vkládání hlavičkových souborů ani funkce `main` - toto je již obsaženo v testovacím prostředí. Pro usnadnění vývoje a odevzdávání (abyste nemuseli před každým odevzdáním ručně odstraňovat funkci `main` a vkládání hlaviček) použijte šablonu výše. Všimněte si, že funkce `main` a vkládání hlavičkových souborů je v bloku podmíněného překladu, tedy jsou testovacím prostředím přeskočeny.

Vaše funkce je testovaná v omezeném prostředí. Omezena je doba běhu i dostupná paměť. Konkrétní omezení je zřejmé z výpisu testování referenčního řešení. V této jednoduché úloze by se ale ani paměťové ani časové omezení nemělo uplatnit.

Úloha vyžaduje pečlivou práci s ukazateli a paměťovou alokací. Úloha má za úkol procvičit práci se spojovými seznamy. Testovací prostředí volá implementované funkce a zároveň kontroluje strukturu Vašeho stromu, zda odpovídá očekávání (neobsahuje zbytečné uzly, ...). Chyba je hlášena jednak pro nesprávný výsledek funkce a dále i pro nesoulad tvaru Vašeho a referenčního stromu.

Dodržte přesně rozhraní funkcí a použitých struktur. Pokud rozhraní nedodržíte, program nepůjde v testovacím prostředí zkompilovat. Zachovejte bloky podmíněného překladu jak jsou v šabloně - svévolné odstranění bloků podmíněného překladu nejspíše způsobí odmítnutí Vašeho programu.

Ukázka použití funkcí:

```
TNODE       * root;
char          tmpStr[100];
const char  * dst;
int           res;

root = NULL;
res = addDest ( &root, "420", "Czech republic" ); /* res = 1 */
res = addDest ( &root, "421", "Slovak republic" ); /* res = 1 */
res = addDest ( &root, "1", "USA" ); /* res = 1 */
res = addDest ( &root, "420606", "CZ - O2 mobil" ); /* res = 1 */
res = addDest ( &root, "420606123456", "CZ - Vodafone" ); /* res = 1 */
dst = search ( root, "420606334455" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "420603212223" ); /* dst = "Czech republic" */
dst = search ( root, "420606123456" ); /* dst = "CZ - Vodafone" */
dst = search ( root, "42060612345" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "37123456" ); /* dst = NULL */
dst = search ( root, "1998877665544332211" ); /* dst = "USA" */
delTree ( root );

root = NULL;
res = addDest ( &root, "420", "Czech republic" ); /* res = 1 */
res = addDest ( &root, "421", "Slovak republic" ); /* res = 1 */
res = addDest ( &root, "1", "USA" ); /* res = 1 */
res = addDest ( &root, "420606", "CZ - O2 mobil" ); /* res = 1 */
res = addDest ( &root, "420606123456", "CZ - Vodafone" ); /* res = 1 */
dst = search ( root, "420222333444" ); /* dst = "Czech republic" */
dst = search ( root, "420606112233" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "420606123456" ); /* dst = "CZ - Vodafone" */
res = delDest ( &root, "420" ); /* res = 1 */
dst = search ( root, "420222333444" ); /* dst = NULL */
dst = search ( root, "420606112233" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "420606123456" ); /* dst = "CZ - Vodafone" */
res = addDest ( &root, "42", "Euro telco company" ); /* res = 1 */
dst = search ( root, "420222333444" ); /* dst = "Euro telco company" */
dst = search ( root, "420606112233" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "420606123456" ); /* dst = "CZ - Vodafone" */
delTree ( root );

root = NULL;
strncpy ( tmpStr, "Czech republic", sizeof ( tmpStr ) );
res = addDest ( &root, "420", tmpStr ); /* res = 1 */
strncpy ( tmpStr, "Slovak republic", sizeof ( tmpStr ) );
res = addDest ( &root, "421", tmpStr ); /* res = 1 */
strncpy ( tmpStr, "USA", sizeof ( tmpStr ) );
res = addDest ( &root, "1", tmpStr ); /* res = 1 */
strncpy ( tmpStr, "CZ - O2 mobil", sizeof ( tmpStr ) );
res = addDest ( &root, "420606", tmpStr ); /* res = 1 */
strncpy ( tmpStr, "CZ - Vodafone", sizeof ( tmpStr ) );
res = addDest ( &root, "420606123456", tmpStr ); /* res = 1 */
dst = search ( root, "420606334455" ); /* dst = "CZ - O2 mobil" */
dst = search ( root, "420603212223" ); /* dst = "Czech republic" */
dst = search ( root, "420606123456" ); /* dst = "CZ - Vodafone" */
dst = search ( root, "37123456" ); /* dst = NULL */
dst = search ( root, "1998877665544332211" ); /* dst = "USA" */
delTree ( root );

root = NULL;
res = addDest ( &root, "420", "Czech republic" ); /* res = 1 */
res = addDest ( &root, "1", "USA" ); /* res = 1 */
dst = search ( root, "420606334455" ); /* dst = "Czech republic" */
dst = search ( root, "12345" ); /* dst = "USA" */
res = delDest ( &root, "1" ); /* res = 1 */
dst = search ( root, "12345" ); /* dst = NULL */
res = delDest ( &root, "420" ); /* res = 1 */
dst = search ( root, "420606334455" ); /* dst = NULL */
res = delDest ( &root, "420" ); /* res = 0 */
res = addDest ( &root, "420A", "???" ); /* res = 0 */
delTree ( root );
```

Referenční řešení

2	**01.12.2012 15:23:34**	**Download**

Stav odevzdání:	Ohodnoceno
Hodnocení:	5.5000

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 3.000 s)
    - Využití paměti: 13264 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty (add, test)': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.152 s (limit: 3.000 s)
    - Využití paměti: 13400 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty (add, del, test)': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.387 s (limit: 2.848 s)
    - Využití paměti: 13400 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.097 s (limit: 2.000 s)
    - Využití paměti: 20416 KiB (limit: 29176 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Všechny paměťové bloky byly uvolněné - ok.
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * (5.00 + 0.50) = 5.50

SW metriky:

	Celkem	Průměr	Maximum	Jméno funkce
Funkce:	**10**	--	--	--
Řádek kódu:	**132**	**13.20 ± 11.13**	**41**	`recDelNode(TNODE *,int,const char *)`
Cyklomatická složitost:	**57**	**5.70 ± 5.00**	**17**	`recDelNode(TNODE *,int,const char *)`

1	**01.12.2012 15:22:16**	**Download**

Stav odevzdání:	Ohodnoceno
Hodnocení:	5.5000

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 3.000 s)
    - Využití paměti: 13264 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty (add, test)': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.151 s (limit: 3.000 s)
    - Využití paměti: 13396 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty (add, del, test)': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.373 s (limit: 2.849 s)
    - Využití paměti: 13396 KiB (limit: 21363 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi hodnotami + test prace s pameti': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.090 s (limit: 2.000 s)
    - Využití paměti: 20416 KiB (limit: 29176 KiB)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Všechny paměťové bloky byly uvolněné - ok.
  - Celkové hodnocení: 100.00 % (= 1.00 * 1.00 * 1.00 * 1.00)
- Celkové procentní hodnocení: 100.00 %
- Bonus za včasné odevzdání: 0.50
- Celkem bodů: 1.00 * (5.00 + 0.50) = 5.50

		Celkem	Průměr	Maximum	Jméno funkce
SW metriky:	Funkce:	**10**	--	--	--
	Řádek kódu:	**248**	**24.80 ± 33.14**	**119**	`main(int,char **)`
	Cyklomatická složitost:	**57**	**5.70 ± 5.00**	**17**	`recDelNode(TNODE *,int,const char *)`