Termín odevzdání:	**01.01.2024 11:59:59**
Pozdní odevzdání s penalizací:	**01.01.2024 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**3.9600**
Max. hodnocení:	**3.0000** (bez bonusů)
Odevzdaná řešení:	2 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	0 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Úkolem je realizovat sadu funkcí pro práci se spojovými seznamy. Spojový seznam reprezentuje seznam zločinců v evidenci policie. Zločinec je reprezentován jménem a odkazy na ostatní zločince, na které má kontakt. S takovým seznamem chceme provádět tyto operace: přidání zločince, přidání kontaktu, kopírování seznamu a mazání seznamu.

`TCRIMINAL`
: tato datová struktura je deklarovaná v testovacím prostředí. Vaše implementace bude s touto strukturou pracovat, ale nesmí jí nijak měnit. Struktura reprezentuje jednoho zločince. Zločinci jsou uloženi v podobě jednosměrně zřetězeného spojového seznamu. Struktura má následující složky:

  - `m_Next` - odkaz na dalšího zločince ve spojovém seznamu. Poslední záznam v seznamu má odkaz `m_Next` nastaven na hodnotu `nullptr`.
  - `m_Name` - ASCIIZ (nulou ukončený) řetězec udávající jméno zločince.
  - `m_Contacts` - pole s odkazy na další zločince, na které má tento zločinec kontakt. V poli se může vyskytovat ten samý odkaz vícekrát (evidence nemá čištěná data) a ve výjimečných případech může být v poli i odkaz na sebe sama.
  - `m_Cnt` - počet vyplněných odkazů v poli `m_Contacts`.
  - `m_Capacity` - prostor alokovaný polem `m_Contacts`.

`createRecord ( name, next )`
: funkce vytvoří nový prvek ve spojovém seznamu a umístí jej na první pozici. Parametrem je `name` - jméno nového zločince a `next` - odkaz na dosavadní počátek spojového seznamu zločinců. Návratovou hodnotou funkce je ukazatel na první prvek nového spojového seznamu zločinců. Funkce je zodpovědná za alokaci struktury a za vyplnění jejích složek. Nově přidávaný zločinec bude mít prázdné pole kontaktů (nulová délka i kapacita).

`addContact ( record, newContact )`
: funkce přidá do záznamu `record` nový kontakt (`newContact`). Záznam bude přidán na konec pole `m_Contacts`, funkce podle potřeby alokuje paměťový prostor tohoto pole.

`freeList ( list )`
: funkce k uvolnění prostředků alokovaných ve spojovém seznamu. Parametrem funkce je odkaz na první prvek spojového seznamu zločinců dříve vytvořených pomocí funkce `createRecord`.

`cloneList ( list )`
: funkce vytvoří kopii spojového seznamu zločinců. Nově vytvořený seznam musí zachovat jména zločinců, jejich pořadí v seznamu a musí správně zkopírovat i pole s kontakty tak, aby kontakty odkazovaly na správné kopie záznamů v nově vytvořeném seznamu. Pozor, nově vytvořený seznam musí být zcela nezávislý na originále, tedy i odkazy na zástupce musí směřovat na prvky nově vytvořeného seznamu. Návratovou hodnotou funkce je odkaz na první prvek vytvořené kopie seznamu.

Odevzdávejte zdrojový soubor, který obsahuje implementaci požadovaných funkcí. Do zdrojového souboru přidejte i další Vaše podpůrné funkce, které jsou z nich volané. Implementované funkce budou volané z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní funkcí. Za základ pro implementaci použijte kód z přiloženého souboru. V kódu chybí vyplnit těla požadovaných funkcí (a případné další podpůrné funkce). Ukázka obsahuje testovací funkci `main`, uvedené hodnoty jsou použité při základním testu. Všimněte si, že vkládání hlavičkových souborů a funkce `main` jsou zabalené v bloku podmíněného překladu (`#ifdef/#endif`). Prosím, ponechte bloky podmíněného překladu i v odevzdávaném zdrojovém souboru. Podmíněný překlad Vám zjednoduší práci. Při kompilaci na Vašem počítači můžete program normálně spouštět a testovat. Při kompilaci na Progtestu funkce `main` a vkládání hlavičkových souborů "zmizí", tedy nebude kolidovat s hlavičkovými soubory a funkcí `main` testovacího prostředí.

Váš program bude spouštěn v omezeném testovacím prostředí. Je omezen dobou běhu (limit je vidět v logu referenčního řešení) a dále je omezena i velikost dostupné paměti. Rozumná implementace naivního algoritmu by měla projít všemi testy kromě testu rychlosti. Pro zvládnutí testu rychlosti je potřeba použít výkonnější algoritmus, který efektivně kopíruje seznamy.

Nápověda:

- Jako základ Vašeho řešení použijte zdrojový kód z přiloženého souboru.
- Do funkce `main` si můžete doplnit i další Vaše testy, případně ji můžete libovolně změnit. Důležité je zachovat podmíněný překlad.
- V ukázce je použito makro `assert`. Pokud je parametrem nenulová hodnota, makro nedělá nic. Pokud je parametrem nepravda (nula), makro ukončí program a vypíše řádku, kde k selhání došlo. Pokud tedy Vaše implementace projde ukázkovými testy, program doběhne a nic nezobrazí.
- O jménech zločinců není obecně nic známo. Tedy jména mohou, ale nemusí být unikátní.
- Řešení této úlohy nelze použít pro code review.

Vzorová data: **Download**

☐ **Referenční řešení**

2	**23.12.2023 20:58:49**	**Download**
Stav odevzdání:	Ohodnoceno	

Hodnocení: 3.9600

- **Hodnotitel: automat**
  - Program zkompilován
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test meznich hodnot': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.013 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty + mem debugger': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.041 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Bonus - rychlost': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.621 s (limit: 1.000 s)
    - Úspěch v bonusovém testu, hodnocení: 120.00 %
  - Celkové hodnocení: 120.00 % (= 1.00 * 1.00 * 1.00 * 1.00 * 1.20)
- Celkové procentní hodnocení: 120.00 %
- Bonus za včasné odevzdání: 0.30
- Celkem bodů: 1.20 * (3.00 + 0.30) = 3.96

SW metriky:

1	**23.12.2023 20:54:59**	**Download**
Stav odevzdání:	Ohodnoceno	
Hodnocení:	3.5640	

- **Hodnotitel: automat**
  - Chyba při kompilaci v režimu 'pedantic' - 10% penalizace **[Zpřístupnit nápovědu (721 B)]**
  - Test 'Zakladni test podle ukazky': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test meznich hodnot': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.000 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.013 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Test nahodnymi daty + mem debugger': Úspěch
    - Dosaženo: 100.00 %, požadováno: 50.00 %
    - Celková doba běhu: 0.045 s (limit: 2.000 s)
    - Úspěch v závazném testu, hodnocení: 100.00 %
  - Test 'Bonus - rychlost': Úspěch
    - Dosaženo: 100.00 %, požadováno: 100.00 %
    - Celková doba běhu: 0.632 s (limit: 1.000 s)
    - Úspěch v bonusovém testu, hodnocení: 120.00 %
  - Celkové hodnocení: 108.00 % (= (1.00 * 1.00 * 1.00 * 1.00 * 1.20) * 0.9)
- Celkové procentní hodnocení: 108.00 %
- Bonus za včasné odevzdání: 0.30
- Celkem bodů: 1.08 * (3.00 + 0.30) = 3.56

SW metriky: