Politicky korektní vyjadřování

Termín odevzdání:	**05.12.2022 11:59:59**
Pozdní odevzdání s penalizací:	**31.12.2022 23:59:59** (Penále za pozdní odevzdání: 100.0000 %)
Hodnocení:	**4.0000**
Max. hodnocení:	**4.0000** (bez bonusů)
Odevzdaná řešení:	8 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	2 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Vaším úkolem je vytvořit funkci (ne program, pouze funkci), která dokáže pomoci PR oddělení při formulaci politicky korektních tiskových prohlášení.

S rozvojem naší společnosti je spojená nutnost formulovat své myšlenky tak, aby prohlášení nemohlo nikoho ranit či dokonce urazit. Tomuto trendu se říká politická korektnost, lidově též newspeak. Podobnost s pojmem newspeak z Orwellova románu je pochopitelně čistě náhodná. Označit někoho za "flákače" je prostě nepřípustné, naproti tomu pojem "osoba se specifickou potřebou plánování účasti v pracovním procesu" je zcela jistě univerzálně použitelné.

Vámi implementovaná funkce tomuto procesu bude napomáhat. Vypisovat dlouhé politicky korektní fráze odmítají i osoby bez specifických potřeb plánování účasti v pracovním procesu. Implementovaná funkce bude v textu nahrazovat nevhodné politicky nekorektní formulace za jejich správnou náhradu. Rozhraní funkce bude:

```
char * newSpeak ( const char * text, const char * (*replace)[2] );
```

Parametry funkce jsou:

- `text` - původní text, ve kterém má dojít k náhradě nevhodných formulací. Tento parametr je **`const`**, tedy funkce jej může pouze číst,
- `replace` - je pole s dvojicemi řetězců, k nahrazení. Pole `replace` je dvojrozměrné, obsahuje `n` řádek a dva sloupce. Ve sloupci 0 je řetězec nevhodné fráze a ve sloupci 1 je její politicky korektní náhrada. Počet řádek pole není explicitně určen, víte ale, že poslední řádka v poli obsahuje dva ukazatele s hodnotou `NULL`.
- Návratovou hodnotou funkce je řetězec s provedenými náhradami. Tento řetězec musí být funkcí dynamicky alokován (buď `malloc` nebo `realloc`). Volající řetězec použije a po jeho použití jej uvolní z paměti (zavolá `free`). V případě chybných parametrů vrací funkce `NULL`.

Chceme, aby náhrady v řetězci byly jednoznačné. Toho dosáhneme následujícím:

- text nahrazujeme zleva doprava,

- jednou nahrazený text již není znovu nahrazován, v náhradách pokračujeme za nahrazeným textem (viz např. ukázka "specialist"),
- nahrazované fráze (sloupec 0) jsou jednoznačné, tedy žádný vzor nesmí být předponou jiného vzoru. Funkce před zahájením práce zkontroluje, že toto platí, pokud ne, vrátí chybovou signalizaci hodnotou `NULL`. V ukázkovém běhu je to porušeno u dvojice `fail` a `failure`.

Všechny řetězce, které funkce dostává/vrací jsou C řetězce (nulou ukončené). Při porovnávání řetězců funkce rozlišuje malá a velká písmenka.

Odevzdávejte zdrojový soubor, který obsahuje implementaci požadované funkce `newSpeak`. Do zdrojového souboru přidejte i další Vaše podpůrné funkce, které jsou z `newSpeak` volané. Funkce bude volaná z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní funkce. Za základ pro implementaci použijte kód z přiloženého souboru. V kódu chybí vyplnit funkci `newSpeak` (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. Časové limity jsou nastavené tak, aby rozumná implementace naivního algoritmu prošla všemi povinnými testy. Řešení fungující v povinných testech může získat nominálních 100% bodů. Bonusový test vyžaduje časově a paměťově efektivní výpočet i pro dlouhé řetězce obsahující mnoho slov a slovníky obsahující stovky dlouhých frází. První bonus kontroluje rychlost, druhý paměťovou náročnost.

Pro implementaci musíte použít C řetězce. C++ řetězce (`std::string`) a STL jsou zakázané, jejich použití povede k chybě při kompilaci.

Řešení, které projde všemi povinnými a neovinnými testy na 100%, může být použito pro code review.