Romantika při úplňku

Termín odevzdání:	**26.11.2017 23:59:59**
Hodnocení:	**5.0000**
Max. hodnocení:	**5.0000** (bez bonusů)
Odevzdaná řešení:	4 / 20 Volné pokusy + 10 Penalizované pokusy (-10 % penalizace za každé odevzdání)
Nápovědy:	1 / 2 Volné nápovědy + 2 Penalizované nápovědy (-10 % penalizace za každou nápovědu)

Jeden nejmenovaný vyučující PA1 je velký romantik. Na rande zásadně chodí pouze, pokud je měsíc v úplňku. V této úloze mu pomůžeme optimalizovat jeho schůzkový kalendář. Úkolem je realizovat funkce (ne celý program, pouze funkce), které budou usnadňovat výpočty dní s úplňkem. Požadované funkce mají následující rozhraní:

`int isFullMoon ( int y, int m, int d )`
: Funkce rozhodne, zda je v daný den úplněk. Parametrem je rok, měsíc a den, který zkoumáme. Návratovou hodnotou je celé číslo 1, pokud je v zadaný den úplněk, 0 pokud není úplněk a hodnota `INVALID_DATE` pokud jsou vstupní parametry nesprávné (nejedná se o platné datum).

`int prevFullMoon ( int y, int m, int d, int *prevY, int * prevM, int * prevD )`
: Funkce vypočte datum předchozího úplňku, tedy poslední úplněk předcházející zadanému datu. Vstupní parametry funkce jsou rok, měsíc a den (`y`, `m` a `d`), pro který počítáme předchozí úplněk. Výstupními parametry jsou `prevY`, `prevM` a `prevD` s vypočteným datem předchozího úplňku. Návratovou hodnotou funkce je hodnota 1 (úspěch) nebo `INVALID_DATE` (neúspěch). Pokud funkce zjistí, že zadané vstupní datum není platné, musí pouze vrátit návratovou hodnotu `INVALID_DATE` a nesmí měnit žádný ze svých výstupních parametrů (`prevY`, `prevM` a `prevD`).

`int nextFullMoon ( int y, int m, int d, int *nextY, int * nextM, int * nextD )`
: Funkce vypočte datum následujícího, tedy první úplněk následující po zadaném datu. Vstupní parametry funkce jsou rok, měsíc a den (`y`, `m` a `d`), pro který počítáme následující úplněk. Výstupními parametry jsou `nextY`, `nextM` a `nextD` s vypočteným datem příštího úplňku. Návratovou hodnotou funkce je hodnota 1 (úspěch) nebo `INVALID_DATE` (neúspěch). Pokud funkce zjistí, že zadané vstupní datum není platné, musí pouze vrátit návratovou hodnotu `INVALID_DATE` a nesmí měnit žádný ze svých výstupních parametrů (`nextY`, `nextM` a `nextD`).

`INVALID_DATE`
: Tato symbolická konstanta je definovaná v testovacím prostředí. Funkce ji používají jako indikaci neúspěchu, když je zadané neplatné datum.

Správné hodnoty vstupních parametrů musí splňovat:

- rok je větší roven 2000 (všechna data před rokem 2000 považujeme za neplatná),
- měsíc je platný (1 až 12),
- den je platný (1 až počet dní v měsíci).

Odevzdávejte zdrojový soubor, který obsahuje implementaci všech požadovaných funkcí. Do zdrojového souboru přidejte i další Vaše podpůrné funkce, které jsou z nich volané. Funkce budou volané z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní funkce. 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. Časové limity jsou nastavené tak, aby rozumná implementace naivního algoritmu prošla všemi testy kromě testů bonusových, tedy byla hodnocena 100% bodů. Bonusové testy vyžadují časově efektivní výpočet i pro data v daleké budoucnosti (vysoké roky hodně převyšující 4000).

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.

- Při výpočtu času uvažujeme Gregoriánský kalendář. Tedy měsíce mají vždy 30 nebo vždy 31 dní, výjimkou je únor, který má 28 dní (nepřestupný rok) nebo 29 dní (přestupný rok). Podle Gregoriánského kalendáře platí:

    1. roky nejsou přestupné,
    2. s výjimkou let dělitelných 4, které jsou přestupné,
    3. s výjimkou let dělitelných 100, které nejsou přestupné,
    4. s výjimkou let dělitelných 400, které jsou přestupné,
    5. s výjimkou let dělitelných 4000, které nejsou přestupné.

Tedy roky 2001, 2002, 2003, 2005, ... nejsou přestupné (pravidlo 1), roky 2004, 2008, ..., 2096, 2104, ... jsou přestupné (pravidlo 2), roky 2100, 2200, 2300, ... nejsou přestupné (pravidlo 3), roky 2000, 2400, ..., 3600, 4400, ... jsou přestupné (pravidlo 4) a roky 4000, 8000, ... nejsou přestupné (pravidlo 5).

- Funkce `prevFullMoon` a `nextFullMoon` vrací vždy datum předešlého/následujícího úplňku. Tedy pokud za vstupní parametr těmto funkcím předáte datum, ve kterém je úplněk, bude výsledkem datum dřívější než zadané (`prevFullMoon`) resp. pozdější než zadané (`nextFullMoon`).
- Při výpočtu úplňku využijte fakt, že úplněk nastává se (střední) periodou 29.53059027 dne, tedy 29 dní a 12:44:03. Dále uvažujte, že referenční úplněk nastal dne 16.7.2000 v 13:55:12 UTC. Pro jednoduchost předpokládáme, že všechny parametry (den, měsíc, rok) jsou rovněž v UTC.
- Skutečná doba mezi dvěma úplňky není konstantní, mění se podle vzájemné polohy Měsíce a Země (roční období) a osciluje mezi 29.18 dne a 29.93 dne (**Wikipedia**, **další vysvětlení**). Přesný výpočet je proto mnohem komplikovanější a není účelem této úlohy. Zjednodušený výpočet ale způsobí, že Vámi vypočtená data úplňků se budou trochu lišit od skutečných hodnot, které jsou zveřejněné na internetu. To je vidět i v ukázkových datech, v prosinci 2019 nastane úplněk až 12.12 brzy ráno, náš zjednodušený model vypočte úplněk už 11.12 pozdě v noci. Dále, pokud budete při kontrole Vašeho programu používat data zveřejněná na internetu, budete je muset přepočítat do UTC (zveřejněná data jsou často v PST, EST nebo v lokální časové zóně - CET/CEST).
- S trojicí hodnot (rok, měsíc, den) se špatně pracuje. Je lepší si tyto hodnoty převést na nějakou jinou reprezentaci, ideálně tak, aby vzniklo pouze jedno číslo.
- V programu musíte mnoho výpočtů dělat 2x - např. pro předchozí/následující úplněk. Je dobrý nápad vytvořit si pomocné funkce, které 2x zavoláte.
- V povinných testech jsou zadávané roky nepřevyšující rok 2200.
- 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.