Úkolem je realizovat funkce (ne celý program, pouze funkce), které budou usnadňovat plánování dopravy.

Předpokládáme dopravní spojení dvou míst A a B. V jeden den jede z A do B nula nebo více spojů. Počet spojů je daný celým číslem. Dále můžeme omezit konkrétní dny v týdnu, ve kterých spoj jezdí. Například můžeme definovat, že spojení je v provozu v pondělí, v pátek a v sobotu (v ostatní dny týdne ne). Pro zjednodušení uvažujeme, že počet spojů je daný jedním celým číslem perWorkDay:

- pokud je spojení v provozu v daný všední den, pak jede perWorkDay spojů,
- pokud je spojení v provozu v sobotu, pak jede perWorkDay / 2 spojů, desetinnou část zaokrouhlíme nahoru,
- pokud je spojení v provozu v neděli, pak jezdí perWorkDay / 3 spojů, desetinnou část zaokrouhlíme nahoru.

Následující tabulka shrnuje počty spojení za jeden týden pro různé kombinace perWorkDay a masku dní v týdnu dowMask:

	perWorkDay = 6	perWorkDay = 19
DOW_MON	6	19
DOW_SAT	3	10
DOW_SUN	2	7
DOW_WORKDAYS	5 × 6	5 × 19
DOW_WEEKEND	3 + 2	10 + 7
DOW_ALL	5 × 6 + 3 + 2	5 × 19 + 10 + 7
DOW_MON | DOW_THU | DOW_SUN	2 × 6 + 2	2 × 19 + 7

Pro plánování potřebujeme vědět, kolik spojů bude celkem potřeba objednat v zadaném časovém intervalu. Dále se hodí i opačná funkce na výpočet časového intervalu, který lze pokrýt objednaným počtem spojů. Tyto výpočty budou realizované požadovanými funkcemi:

countConnections (from, to, perWorkDay, dowMask)
    funkce dostane v parametrech časový interval <from; to>, počet spojů během jednoho všedního dne perWorkDay a masku dní v týdnu, kdy je spoj v provozu dowMask. Na základě těchto parametrů funkce vypočte počet spojů, které je potřeba objednat pro pokrytí zadaného intervalu dní. Interval chápeme jako uzavřený, tedy obsahuje celý první i celý poslední den. Pokud jsou parametry neplatné, funkce vrátí hodnotu -1. Za chybu považujeme:

    - neplatné datum from nebo to nebo
    - počátek intervalu from nastane po konci intervalu to (tj. nesprávné je from > to).

    Implementace této funkce je Vaším úkolem.
endDate (from, connections, perWorkDay, dowMask)
    funkce dostane počáteční datum from, počet objednaných spojení connections, počet spojů během jednoho všedního

dne perWorkDay a masku dní v týdnu, kdy je spoj v provozu dowMask. Na základě těchto parametrů funkce vypočte datum posledního dne, který jsme se zadaným počtem spojeni schopni pokrýt (tj. pro další den již nebude dostatek objednaných spojení). Návratovou hodnotou je nalezené datum nebo datum 0000-00-00 pro neplatnou kombinaci parametrů:

- neplatné datum from,
- záporný počet objednaných spojení connections,
- počet objednaných spojení connections nestačí ani na den from,
- nulová hodnota spojení perWorkDay nebo
- prázdná maska dowMask.

Implementace této funkce je Vaším úkolem.

TDATE
: je struktura reprezentující datum. Tvoří ji složky rok, měsíc a den. Struktura je deklarovaná v testovacím prostředí, Vaše implementace ji může/musí používat. Nelze ale měnit deklaraci struktury.

makeDate(y, m, d)
: pomocná funkce deklarovaná v testovacím prostředí. Funkci můžete použít pro usnadnění ladění. Implementaci nelze změnit.

konstanty DOW_MON, DOW_TUE, ... , DOW_ALL
: konstanty jsou deklarované v testovacím prostředí a nelze je měnit. Používají se pro masky dní, ve kterých spoj jezdí.

Odevzdávejte zdrojový soubor, který obsahuje implementaci požadovaných funkcí Do zdrojového souboru přidejte i další Vaše podpůrné funkce, které pro implementaci potřebujete. Implementované funkce countConnections a endDate budou volané z testovacího prostředí, je proto důležité přesně dodržet zadané rozhraní. 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 testů rychlosti. Pro zvládnutí testů rychlosti je potřeba použít výkonnější algoritmus.

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.

- 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).

- Zadávaný rok bude vždy nejméně 2000 a nejvýše 1000000000. Pro povinné testy bude interval relativně malý, tedy zadávaný rok nebude vysoký. V bonusovém testu jsou naopak zadávané velmi dlouhé intervaly.
- Pozor – v testovacím prostředí nejsou vložené hlavičkové soubory time.h.