Funkcje operujące na liczbach

Rozwiązania poniższych zadań to proste funkcje operujące na liczbach całkowitych.
Dla każdej funkcji pomocniczej podaj jej specyfikację, tj. warunek wstępny i końcowy, oraz uzasadnij jej poprawność.

1. Stopień parzystości liczby całkowitej x, to największa taka liczba naturalna i, że x dzieli się przez 2^i.
   Liczby nieparzyste mają stopień parzystości 0, liczby 2 i -6 mają stopień parzystości 1, a liczby 4 i 12 mają stopień parzystości 2.
   Przyjmujemy, że 0 ma stopień parzystości -1.
   Napisz procedurę `parzystość` wyznaczającą stopień parzystości danej liczby całkowitej.
2. Napisz procedurę, która przekształca daną liczbę naturalną w taką, w której cyfry występują w odwrotnej kolejności, np. 1234 jest przekształcane na 4321.
3. Sumy kolejnych liczb nieparzystych dają kwadraty kolejnych liczb naturalnych, zgodnie ze wzorem: $\sum_{i=1}^{k}(2i-1) = k^2$
   Wykorzystaj ten fakt do napisania procedury `sqrt` obliczającej `sqrt(x)` = $\lfloor\sqrt{x}\rfloor$ i nie korzystającej z mnożenia, ani dzielenia.
4. Liczbę naturalną nazwiemy *rzadką*, jeżeli w jej zapisie binarnym żadne dwie jedynki nie stoją obok siebie.
   Napisz procedurę `int rzadkie(int n)`, która dla danej liczby naturalnej n, zwróci najmniejszą rzadką liczbę naturalną większą od n.
   Na przykład, dla $42 = 101010_2$, mamy `rzadkie(42) ==` 1000000_2 `== 64`.
5. Liczbę naturalną nazwiemy *rzadką*, jeżeli w jej zapisie binarnym żadne dwie jedynki nie stoją obok siebie.
   Napisz procedurę `int rzadkie(int n)`, która dla danej liczby naturalnej n, wyznaczy liczbę dodatnich liczb rzadkich, które nie przekraczają n.
   Na przykład, dla $42 = 101010_2$, mamy `rzadkie 42 = 20`.
6. Napisz procedurę, która sprawdza, czy dana liczba jest pierwsza.
7. Zaimplementuj kodowanie par liczb naturalnych jako liczby naturalne. To znaczy, napisz procedurę dwuargumentową, która koduje dwie liczby dane jako argumenty w jednej liczbie naturalnej. Dodatkowo napisz dwie procedury, które wydobywają z zakodowanej pary odpowiednio pierwszą i drugą liczbę. (W tym zadaniu nie musisz się przejmować tym, że liczby całkowite są ograniczone.)
8. Napisz procedurę, która dla danej liczby n sprawdzi czy pierścień reszt modulo n zawiera nietrywialne pierwiastki z 1 (tj. takie liczby $k \neq 1$, $k \neq n-1$, że $k^2 \equiv 1 \mod n$).
   Nota bene, jeśli takie pierwiastki istnieją, to liczba n nie jest pierwsza.
   Odwrotna implikacja jednak nie zachodzi -- np. dla n = 4 nie ma nietrywialnych pierwiastków z 1.
9. Napisz procedurę `int zera(int a, int b)`, która dla danych dodatnich liczb naturalnych *a* i *b* obliczy ile jest łącznie zer w zapisie dziesiętnym liczb *a*, *a*+1, ..., *b*.
   Na przykład, `zera(11, 230) == 42`.
10. Napisz procedurę `int zera(int n)`, która dla danej dodatniej liczby naturalnej *n* obliczy ile jest zer na końcu liczby *n*! (oczywiście w zapisie dziesiętnym).
11. Napisz procedurę `int podzielnosc(int n, int p)`, która dla danych: dodatniej liczby naturalnej *n* oraz liczby pierwszej *p* obliczy największą taką liczbę całkowitą *k*, że *n*! jest podzielne przez p^k.
    Na przykład, `podzielnosc 42 3 == 19`.
12. Napisz procedurę `int bity(unsigned int n)`, która dla danej nieujemnej liczby całkowitej *n* policzy ile jest 1-ek w zapisie binarnym liczby *n*.

?

W tym zadaniu przydadzą się operacje binarne z wykładu.
(Jeżeli znasz `__builtin_popcount()`, to go nie używaj, bo nie o to chodzi w tym zadaniu.)
Potrafisz rozwiązać to zadanie w złożoności lepszej niż O(log *n*)?

Last modified: Sunday, 27 November 2022, 12:07 PM

Get the mobile app