Uwagi wstępne:

1. Proszę nie wysyłać pełnych projektów – jedynie pliki z kodem do sprawdzenia.
2. Powinny to być prawidłowe pliki źródłowe – proszę nie przesyłać fragmentów kodu w plikach tekstowych, pdf itp.
3. Proszę nie łączyć zadań ze sobą – każde zadanie powinno być osobnym pliku lub katalogu.
4. Jeżeli zadanie polega na napisaniu funkcji, nie trzeba wysyłać kodu, który ją wywołuje (całego programu).
5. Proszę precyzyjnie czytać treści zadań. Rozwiązania nie powinny obejmować niczego, co nie jest ujęte w zleceniu – dotyczy to zwłaszcza wypisywania komunikatów na standardowym wyjściu.

Zadanie 1

Napisz program, który wypisuje liczby od 1 do 100. Ale dla wielokrotności trójki wyświetl "Fizz" zamiast liczby oraz dla wielokrotności piątki wyświetl "Buzz". Dla liczb będących wielokrotnościami trójki oraz piątki wyświetl "FizzBuzz".

Zadanie 2

Napisz funkcję sprawdzającą poprawność daty w latach 2001-2099 (daty spoza tego okresu uznaj za niepoprawne).
Wejście – trzy parametry liczbowe (dzień, miesiąc, rok).
Wyjście – parametr logiczny (true – data poprawna, false – data niepoprawna) .
Proszę zaimplementować własny algorytm kontroli – nie wolno korzystać z gotowych rozwiązań, np. LocalDate, Calendar, itp.

Zadanie 3

Napisz program symulujący grę w kości. W grze bierze udział dwóch graczy o nazwach: pierwszy, drugi. Każdemu graczowi przypada 5 tur. Tury graczy odbywają się naprzemiennie tzn. grę zaczyna gracz pierwszy, po jego turze następuje tura gracza drugiego, następna jest tura gracza pierwszego itd. Grę zawsze rozpoczyna gracz pierwszy. Podczas każdej tury gracz zbiera punkty karne według zasad opisanych poniżej. Suma punktów każdego gracza przed rozpoczęciem gry wynosi zero. Wygrywa gracz, który zbierze mniejszą liczbę punktów.

W każdej turze, gracz wykonuje maksymalnie 10 rzutów dwoma kośćmi jednocześnie. Możliwe wyniki rzutu jedną kością to jedna z następujących cyfr {1, 2, 3, 4, 5, 6}.

Zasady gry:
1. Jeżeli w pierwszym rzucie tury gracz uzyska sumę oczek z obu kości równą 7 lub 11, wygrywa turę przed czasem (rozpoczyna się tura przeciwnika).
2. Jeżeli gracz w pierwszym rzucie tury uzyska sumę oczek z obu kości równą 2 lub 12, przegrywa turę przed czasem (rozpoczyna się tura przeciwnika). Do punktów gracza doliczana jest maksymalna możliwa liczba punktów karnych za turę (tyle, ile by uzyskał w najbardziej pesymistycznym przebiegu swojej tury).
3. Jeżeli gracz uzyska sumę oczek z obu kości równą 5, wygrywa turę przed czasem (rozpoczyna się tura przeciwnika).
4. Jeżeli gracz uzyska sumę oczek z obu kości inną niż opisane powyżej, do jego punktów doliczona zostaje suma oczek uzyskanych w danym rzucie podzielona przez numer rzutu w danej turze.

Po zakończeniu wszystkich tur obydwu graczy na standardowe wyjście zostaje wypisany komunikat o sumie punktów zdobytych przez każdego z graczy oraz informacja o zwycięzcy.

Zadanie 4

Celem zadania jest napisanie funkcji sprawdzającej, czy podany tekst jest kodem kreskowym EAN-13 lub EAN-8.

Na wejściu funkcji są dwa parametry:
- wejściowy kod kreskowy: parametr tekstowy
- rodzaj kodu kreskowego - parametr numeryczny: 1 dla EAN-8, 2 dla EAN-13.

Niektóre towary (np. czasopisma) mają dodatkowe kody (tzw. add-on'y) - należy mieć na uwadze, że skaner może dokleić je bezpośrednio do właściwego kodu kreskowego (np. dla towaru o kodzie "6920702707721" oraz add-on'ie "12" na wejściu możliwy jest ciąg "692070270772112"). Należy założyć, że add-on'y mogą występować zarówno dla kodów EAN-8 jaki i EAN-13.

Należy mieć na uwadze, że niektóre skanery kodów kreskowych mogą wycinać z kodu kreskowego pierwsze wiodące zero (np. zamiast kodu "0075678164125" przesyłają "075678164125").

Na wyjściu funkcja powinna zwracać prawidłowy kod kreskowy (o długości 8 lub 13 znaków) bez ewentualnego add-on'u.
Ewentualne błędy w danych wejściowych powinny być sygnalizowane wyjątkami.

Informacje na temat kodów kreskowych: http://pl.wikipedia.org/wiki/EAN