Coding Competition

Pokyny pro soutežící

- Na řešení deseti příkladů máte 50 minut. Pořadí příkladů nutně nezohledňuje jejich předpokládanou časovou náročnost.
- Jaknile vyřešíš nějaký příklad, zaznamenej to do herního plánu na tabuli. Za každý zaznamenaný příklad získáš bod, soutěžící s nejvyšším počtem bodů vítězí.
- Je možné používat vlastní notebook včetně všech dostupných materálů. Příklady řeší každý samostatně. Není možné s nikým dalším komunikovat a to fyzicky, ani virtuálně.

Příklad 1: Hello World!

Argumentem funkce `welcome(n)` je přirozené číslo `n` zapsané v desítkové soustavě. Pokud má toto číslo v binárním zápise na konci jedničku, funkce vypíše na standardní výstup řetězec "Hello World!", v opačném případě (tedy pokud poslední cifrou binárního zápisu čísla `n` je nula) vypíše funkce řetězec "(no answer)".

```
>>> welcome(1)
Hello World!

>>> welcome(2)
(no answer)

>>> welcome(9)
Hello World!

>>> welcome(32)
(no answer)
```

Příklad 2: Násobilka

Funkce `multiple(n)` vytiskne prvních deset násobků čísla `n`.

```
>>> multiple(1)
1 2 3 4 5 6 7 8 9 10

>>> multiple(7)
7 14 21 28 35 42 49 56 63 70
```

Příklad 3: Mínus jedna

Na řádku v textovém řetězci `text` je napsáno několik i víceciferných přirozených čísel oddělených mezerami. Funkce `minus_one(text)` od každého čísla odečte jedničku a výsledek vytiskne na výstup opět na jeden řádek.

```
>>> minus_one('10␣12␣6␣345')
9 11 5 344

>>> minus_one('324593')
324592

>>> minus_one('1␣1␣1␣1␣1')
0 0 0 0 0
```

Příklad 4: Najdi soudělná čísla

Funkce `divisible(l, r)` dostane jako argument dvě přirozená čísla $l, r; l < r$, která udávají rozsah intervalu. Vaším úkolem je najít taková přirozená čísla a, b, pro která platí $l \leq a < b \leq r$ a zároveň a je dělitelem b. Pokud je možných řešení více, vypište libovolné z nich. Pokud není žádné, funkce vypíše řetězec "no solution".

```
>>> divisible(2, 12)
6 12

>>> divisible(1, 4)
2 4

>>> divisible(20, 29)
no solution
```

Příklad 5: Balíky

Představ si, že vlastníš balíkovou službu a rádi byste v ní zautomatizovali rozhodování, jestli nějaký předmět dokážete zabalit do vašich balíků, nebo ne. Používáte pouze tři typy balíků, mají rozměry $2 \times 3 \times 6$, $1 \times 2 \times 10$ a $1 \times 5 \times 5$. Funkce `package(a, b, c)` dostane tři čísla udávající rozměry zboží. Pokud se zboží vejde do některého z balíků, funkce vytiskne "YES", pokud ne, pak funkce vytiskne "NO". Zboží může být v balíku libovolně převráceno, počítejte ale pouze s ortogonálními pozicemi (zboží nemůže být v balíku šikmo).

```
>>> package(1, 3, 5)
YES
>>> package(8, 2, 1)
YES
>>> package(6, 3, 2)
YES
>>> package(7, 2, 3)
NO
```

Příklad 6: Nejbližší větší prvočíslo

Funkce `closest_prime(n)` spočítá a vráti nejbližší větší prvočíslo k zadanému číslu `n`. Pokud je už `n` prvočíslo, vrátí `n`.

```
>>> print(closest_prime(5))
5

>>> print(closest_prime(8))
11

>>> print(closest_prime(32))
37
```

Příklad 7: Filter seznamů

Je potřeba profiltrovat seznam čísel jen na dělitelná zadaným číslem. Funkce `list_filter(alist, divisor)` pro zadaný seznam `alist` vytvoří nový seznam, který bude obsahovat jenom taková čísla, která jsou dělitelná číslem `divisor`. Seznam obsahuje pouze přirozená čísla a může být i prázdný. Funkce vrací nový seznam.

```
>>> print(list_filter([], 5))
[]

>>> print(list_filter([1, 2, 3, 4], 2))
[2, 4]

>>> print(list_filter([1, 2, 3, 4, 5], 4))
[4]

>>> print(list_filter([1, 3, 5], 7))
[]

>>> print(list_filter([1, 3, 5], 1))
[1, 3, 5]
```

Příklad 8: Tři nejdelší slova

Funkce `three_words(text)` vybere z textového řetězce `text` tři nejdelší slova a vytiskne je na výstup v pořadí od nejdelšího po nejkratší. Stejná slova se nevypisují vícekrát. Slova se stejným počtem písmen jsou seřazena podle abecedy. Můžete předpokládat, že řetězec `text` neobsahuje interpunkci, slova jsou oddělena mezerami.

```
>>> three_words('lepsi␣je␣udelat␣malo␣veci␣dobre␣nez␣hodne␣veci␣
    spatne')
spatne udelat dobre

>>> three_words('kdo␣chce␣hybat␣svetem␣at␣nejprve␣hybe␣sam␣sebou')
nejprve svetem hybat
```

Příklad 9: Samohlásky

Funkce `last_three_vowels(text)` nahradí první tři samohlásky řetězce `text` znakem `'.'`. Pokud se v textu vyskytují méně než tři samohlásky, nahradí je všechny. Vstupní řetězec je složen pouze z malých písmen anglické abecedy a mezer, za samohlásky považujeme znaky a, e, i, o, u a y. Funkce vytiskne nový vstup přímo do konzole, nevrací jej.

```
>>> last_three_vowels("ticha␣voda␣brehy␣mele")
t.ch. v.da brehy mele

>>> last_three_vowels("hello␣world")
h.ll. w.rld

>>> last_three_vowels("oi")
..

>>> last_three_vowels("strc␣prst␣skrz␣krk")
strc prst skrz krk
```

Příklad 10: Kontrola uzávorkování

Řetězec `text` je složen pouze ze znaků `()[]{}`, funkce `parenthesis_check(text)` Zkontroluje, zda se jedná o korektní uzávorkování.

```
>>> parenthesis_check('([]({()}))[]{[()]}')
True

>>> parenthesis_check('([)]')
False
```