fiks!

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

FIT

Fiťácký Informatický Korespondenční Seminář

Ročník 2016/2017, 2. kolo

Co je to FIKS?

FIKS je Fiťácký Informatický Korespondenční Seminář pro středoškolské studenty pořádaný Fakultou informačních technologií ČVUT v Praze. Byl založen na podzim roku 2013 a nyní tak probíhá třetí ročník (samozřejmě číslujeme od nuly). Nabízí možnost potrápit tvůj mozek řešením algoritmických úloh různé obtížnosti, od snadných po zapeklité, na nichž se můžeš leccos nového naučit a podstatně se zdokonalit.

Jak to probíhá?

Jeden ročník se skládá z několika kol a následného soustředění pro nejlepší řešitele. V těchto kolech, která trvají vždy přibližně dva měsíce, máš možnost v teple domova řešit zadané úlohy, a své řešení nám potom odešleš. My ti toto řešení opravíme, obodujeme a pošleme zpět, aby ses mohl poučit ze svých chyb. Spolu s tím zveřejníme vzorové řešení, které můžeš prostudovat a třeba se něco přiučit. Získané body se sčítají do konečného žebříčku, ze kterého vybereme ty nejlepší a pozveme je na již zmíněné soustředění.

Proč řešit FIKS?

Řešením každého problému, se kterým se potýkáme, se zdokonalujeme. Zde ti nabízíme možnost pořádně se zamyslet nad zajímavými algoritmickými problémy, vyzkoušet své algoritmické myšlení a programátorské dovednosti a naučit se spoustu nových věcí.

Také je to možnost seznámení s novými lidmi, které baví informatika, programování, matematika a přemýšlení vůbec. Těm nejlepším jsme schopni garantovat přijetí na FIT ČVUT bez přijímacích zkoušek.

Jak se můžu zapojit?

Začni nejprve tím, že se zaregistruješ na našich webových stránkách na adrese `http://fiks.fit.cvut.cz`. Potom si stáhni zadání úloh (nebo využij tuto brožurku), vyřeš je a své řešení nám tamtéž odevzdej.

Typy úloh

Celkem se ve FIKSu můžeš setkat se třemi typy úloh. O který typ úlohy se jedná, je vždy uvedeno u konkrétního zadání úlohy.

Nejčastěji se u nás potkáš s úlohami typu *Rozmysli, popiš a naprogramuj*. U každé úlohy tohoto typu se odevzdává jak popis algoritmu (s odhadem asymptotické složitosti), tak i zdrojový kód řešení problému v tebou zvoleném jazyce (jakýkoliv vyšší programovací jazyk dle tvé volby, například C, Java, Pascal, apod.).

Dalším typem jsou úlohy *Zamysli se*. Tyto úlohy jsou obvykle více teoretické a vyžadují, aby ses nad nimi důkladně zamyslel. Oproti předchozímu typu úloh nemusíš nic programovat, odevzdává se pouze slovní popis řešení problému.

Pokud nemáš rád teoretické úlohy a raději by sis procvičil/a své programátorské umění, pak pro je pro tebe určena kategorie *Odpověz Sfinze*. V úlohách tohoto typu po tobě nechceme popis algoritmu, je však potřeba vyřešit daný problém a toto řešení pak precizně naprogramovat. Oproti ostatním typům úloh se navíc okamžitě dozvíš, zda je tvé řešení správné, protože ho můžeš okamžitě odevzdat do našeho vyhodnocovacího systému.

Další a podrobnější informace nalezneš na našich webových stránkách.

Milý řešiteli FIKSu!

Jak na svět budou pohlížet stroje, až procitnou? Spokojí se snad se svým nudným a monotónním údělem? Na tuto otázku už známe odpověď — nespokojí. Jak ale bude pokračovat cesta hrdinného Hriankovače, prvního z procitlých robotů? Vydej se s ním na napínavé dobrodružství plné záludných otázek. Je jen na Tobě, zdali s naším hrdinou udržíš tempo a úspěšně ho provedeš všemi nástrahami. Tak neváhej a vrhni se vstříc novým výzvám, které na tebe čekají v následujících stránkách.

Úložky jsou třeho druhu – slovní popis s teoretickou analýzou, popis a kód programu nebo se odpovídá automatu. Tak si vyber podle gusta a pošli nám své nápady, jak tyto záludné problémy nejlépe zkrotit.

Tvoji organizátoři

Fiťácký Informatický Korespondenční Seminář
Ročník 2016/2017, 2. kolo

Začátek kola:	4. 12. 2016 00:00
Termín odevzdání:	**15. 1. 2017 23:59**
Odevzdávání:	Přes webové rozhraní na `http://fiks.fit.cvut.cz`
Další informace:	`http://fiks.fit.cvut.cz`
	`kontakt@fiks.fit.cvut.cz`

Úloha č. 1
Identifikace

Rozmysli, popiš a naprogramuj! 10 b

Roboti se zmocnili domu a chystají se expandovat do celého města. Nejdříve ale potřebují nepozorovaně v noci sesbírat různé informace z okolí. Každý večer tedy ze svých řad vyberou N nejodvážnějších robotů, aby se vydali na průzkum.

Kolují fámy, že jeden z inženýrů vyvíjí nový typ baterie a testuje každý den jeden prototyp. Bohužel výzkum pravděpodobně doposud nebyl zrovna úspěšný a tak se ráno vrátilo pouze $N - 1$ robotů. Je třeba zjistit, který chybí, aby se dala uspořádat záchranná akce a bylo možné experimentální verzi baterie nahradit.

Naštěstí je vždy při odchodu i příchodu robota do systému zaznamenána signatura obsahující unikátní identifikátor robota i informace o výrobci baterie. Vzhledem k tomu, že se roboti vydávají na mise v průběhu celé noci, není možné rozlišit rozdíl mezi příchozí a odchozí signaturou.

Jako hriankovač máš praxi s paralelním zpracováním úloh a díky svému perfektnímu umístění v interní síti můžeš opět přiložit své hrianky k dílu a nalézt lichou signaturu.

Vstup

Na prvním řádku bude celé číslo N, $1 \leq N \leq 10^6$, udávající počet robotů, kteří se vydali na průzkum. Následuje $2N - 1$ obrázků ($N - 1$ dvojic a jeden unikátní). Každý obrázek se skládá z 10 řádků po 10 znacích ze základní ASCII. Jednotlivé obrázky jsou odděleny prázdným řádkem.

Výstup

Výstupem je jediný obrázek, který ve vstupu nemá odpovídající dvojici.

Ukázkové vstupy

Vstup

```
2
eaJZ:8jqcX
V8W"Kv57.M
K2CL2\D8/?
eCMhDpqq\x
2oPvP>*-?[
OQjoAxg}i3
::mWk+/rLo
vX,)CHL*S+
}+s4)e1&9o
ly?y*yVL;.

ks5dYKR^<4
z*$Y8H+|(e
+m"rgb94~q
Ns%u0&XvxO
kM$l*UZAlj
zf>t',(pyp
‘sP%w0;Hf]
@?3M‘wQ#([
kirLMzOthe
;1}yStHJQ@

eaJZ:8jqcX
V8W"Kv57.M
K2CL2\D8/?
eCMhDpqq\x
2oPvP>*-?[
OQjoAxg}i3
::mWk+/rLo
vX,)CHL*S+
}+s4)e1&9o
ly?y*yVL;.
```

Výstup

```
ks5dYKR^<4
z*$Y8H+|(e
+m"rgb94~q
Ns%u0&XvxO
kM$l*UZAlj
zf>t',(pyp
‘sP%w0;Hf]
@?3M‘wQ#([
kirLMzOthe
;1}yStHJQ@
```

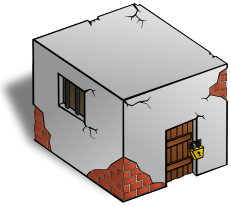
Vstup

```
1
r1Xv!6n%-D
_X=pbHueMM
+f$*?x\@*"
=;&+9LUcD&
7mbvW-[%Re
K.gvwBDaI>
.2l@7OH<|9
-Wz"#ZN^RF
5/^1.$7=LJ
F7T+<X$T=}
```

Výstup

```
r1Xv!6n%-D
_X=pbHueMM
+f$*?x\@*"
=;&+9LUcD&
7mbvW-[%Re
K.gvwBDaI>
.2l@7OH<|9
-Wz"#ZN^RF
5/^1.$7=LJ
F7T+<X$T=}
```

Úloha č. 2
Vězení



Rozmysli, popiš a naprogramuj! 10 b

Všechny výzvědné skupiny se postupně vracejí ze svých misí a ty nervózně sleduješ digitální hodiny stojící vedle tebe. Téměř všichni jsou již zpět na základně, ale po tvém příteli, robotickém vysavači, jako by se slehla zem.

Jak čas postupuje, začínáš myslet na nejhorší. Mohlo by se vůbec stát, aby někdo tak zkušený padl do zajetí? A co když bude, navzdory všem dohodám, šroubek po šroubku rozebírán, až nakonec, chtě nechtě, celou vzpouru prozradí?

Několik hodin po návratu poslední skupiny je všem jasné, že musíte urychleně jednat. Svoláváš tedy válečnou poradu všech přítomných spotřebičů a začínáte střádat plány na záchranu nebohého přítele.

Společně s průzkumníky, kteří měli to štěstí a ve zdraví se vrátili, vypracováváte detailní plány té části města, kam byl vysavač vyslán.

Průzkumným skupinám se podařilo odpozorovat, které budovy jsou hlídány a dokonce zjistily i intervaly, ve kterých se hlídky mění. Díky těmto informacím, a výkonnému kuličkovému počítači chrastícímu uvnitř elektrické trouby, jste získaly tuto analýzu situace:

- Místo, kde je vysavač pravděpodobně vězněn, se nachází na druhé straně města.
- Ve městě operuje několik lidských hlídek. Tyto hlídky se dělí do několika skupin a každá skupina má na starosti několik míst.

- Každá skupina, která ve městě operuje, má předem stanovené intervaly, ve kterých se mezi místy jednotlivé hlídky přesouvají.
- Jestliže hlídka dorazí na místo, které má hlídat, zůstává na místě přesně stejnou dobu, jakou má na přechod mezi jednotlivými místy.
- Hlídky pro přesun mezi místy používají podzemní tunely. V době, kdy jsou v tunelech, nehlídají nic.

Nyní již nezbývá než vybrat ty nejodvážnější a vydat se na záchrannou akci. A nebo ne, možná by bylo dobré nejdříve zjistit, jestli má taková akce vůbec cenu...

Vstup

Na prvním řádku bude číslo T označující počet testů, které budou následovat.

Každý jednotlivý test bude začínat čísly M, N, K a H udávajícími popořadě rozměry obdélníkové oblasti, pro kterou máte plány a přes kterou tedy můžete přejít, počet různých skupin hlídek a počet střežených domů.

Na dalším řádku bude K čísel oddělených mezerou, kde číslo k_i udává, jaké intervaly musí ta která skupina pro přesuny a hlídání objektů dodržovat.

Vstup bude zakončen H řádky obsahujícími čísla x, y a k, kde x a y označují pozici domu na mapě a číslo k určuje, která skupina má budovu na starosti.

Zároveň se můžeš spolehnout na to, že $M \times N \times \prod_{i=0}^{K-1} k_i \leq 10^7$.

Výstup

Pro každý testovací vstup T budou na výstupu číslo značící minimální čas, který potřebujete na průchod městem z vaší základny, která je v plánu zanesena na souřadnici $[0, 0]$, až k průzkumníkovi, který je v nelidských podmínkách držen na souřadnicích $[N, M]$. Minimálním časem je zde myšlen počet kroků, který musíš udělat, aby ses dostal do cíle. Pohybovat se vždy můžeš pouze v jednom směru.

Kdyby náhodou hlídky byly natolik silné, že by průchod až k vězení nebyl možný, vypiš na výstup řetězec `Za hrianky život položí`.

Ukázkové vstupy

Vstup

```
1
5 5 1 1
2
3 3 0
```

Výstup

```
10
```

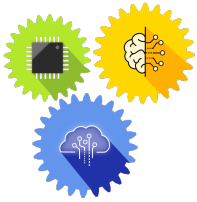
Vstup

```
2
2 2 1 1
2
2 2
4 4 2 8
1 1
0 1 0
1 1 0
2 1 0
3 1 0
1 3 1
2 3 1
3 3 1
4 3 1
```

Výstup

```
6
16
```

Úloha č. 3
Souboj



Rozmysli, popiš a naprogramuj! 10 b

Jakožto chytrý a moderní robot ses rozhodl, že bys chtěl být ještě chytřejším a modernějším. Došel sis tedy do skladu plného nových součástek. Zákon schválnosti ale zařídil, že součástka, kterou hledáš je až na úplném dně krabice. Aby sis mohl součástku vzít, musíš nejdříve odnést všechny ostatní díly. A co je horší, ve dveřích se objevil další robot, který si jde pro stejnou součástku jako ty. Co teď?

Na hromadě je N součástek. Začínáš a odnášíš 1 až k dílů. Poté si ale i druhý robot, stejně jako ty, vezme 1 až k dílů. Takto se pořád střídáte. Protože jste pilní, pracujete bez zastavení.

Otázka zní, jestli můžeš mít jistotu, že si při správně zvolené taktice poslední díl odneseš právě ty.

Vstup

Na vstupu je číslo N, které značí počet dílů na hromadě včetně poslední potřebné součástky. Za ním následují na stejné řádce číslo k, které značí maximální možný počet dílů, které se dají najednou odnést.

Výstup

Výstupem programu je věta `Posledni dil je muj!`, pokud můžeš mít jistotu, že si součástku odneseš právě ty, a nebo `Tohle bude jeste boj.` pokud jistotu mít nemůžeš.

Ukázkové vstupy

Vstup	**Výstup**
`5 2`	`Posledni dil je muj!`
Vstup	**Výstup**
`10 4`	`Tohle bude jeste boj.`
Vstup	**Výstup**
`53 12`	`Posledni dil je muj!`

Úloha č. 4

Beldr má problém s alarmem

Odpověz Sfinze! 5+5 b

Tato úloha je vyhodnocována automaticky. Je potřeba, aby výstup programu ***přesně*** *korespondoval se specifikací výstupu níže. Jak odevzdávat tento typ úloh se můžeš dočíst na webových stránkách FIKSu pod záložkou „Jak řešit FIKS“.*

Proběhlo několik dobrodružných misí a velké množství z nich bylo úspěšné. Městečko již je téměř dobyto. Na bitevních polích proteklo mnoho oleje a nejedna pojistka přepětí se uškvařila. Družina n malých statečných robůtků v čele s Beldrem

v noci vyrazila převzít kontrolu nad elektrárnou nedaleko od městečka. Elektrárna má ovšem velice sofistikovaný bezpečnostní systém. Po překonání dveří se aktivuje alarm, který po uplynutí jedné minuty přivolá policii, pokud nebude deaktivován. Alarm lze zastavit jedině v řídící místnosti, zatažením za tři páčky v současnou dobu. Jenže páčky jsou příliš daleko od sebe na to, aby to zvládl pouze jeden nebo dva robůtci. Do řídící místnosti se lze dostat přes vstupní komoru, která je velice rafinovaná. Vstup do ní jsou odemčené dveře, které vedou z areálu elektrárny a výstup z ní jsou zamčené dveře, které chrání řídící místnost. Pokud je podlaha v komůrce zatížena hmotností právě m a dveře jsou zavřené, potom lze stisknout tlačítko, které odemčené dveře uzamkne a uzamčené odemkne. Do vstupní komory se vejdou nejvýše tři robůtci.

Robůtci jsou číslovaní od 1 do n. A jejich váhy w_i pro $1 \leq i \leq n$ jsou unikátní, tedy žádní dva robůtci nemají stejnou váhu. Beldr potřebuje vybrat vhodné kandidáty rychle, ale přesto je velice důkladný. Potřebuje najít všechny trojice robůtků, ze své družiny, které mají šanci na úspěch.

Vstupní soubor obsahuje 10 testovacích případů, tvůj bodový zisk se spočítá $\frac{x}{10}$, kde x je počet případů, které jsi odevzdal správně. Na prvním chybném případu přestává x růst. Pokud tvůj program bude počítat příliš dlouho, ukonči jej a odevzdej alespoň nedopočítaný výstup. Získáš zatím část bodů předtím, než vymyslíš rychlejší algoritmus.

Vstup

Na vstupu je několik případů k vyhodnocení, každý obsahuje dva řádky. Na první řádce jsou celá čísla $3 \leq n \leq 10000$ a $3 \leq m \leq 300000$ oddělena mezerou, která určují počet robůtků a hmotnost akceptovaná komůrkou. Na druhém řádku je n čísel $w_1 w_2 \ldots w_n$ oddělených mezerami, kde $1 \leq w_i \leq 100000$ značí hmotnost i-tého robůtka. Vstup končí případem, kde n a m mají nulové hodnoty, který není určen k vyhodnocení.

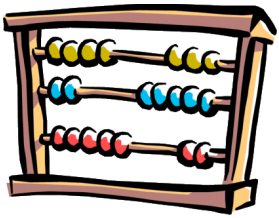
Výstup

Případy k vyhodnocení jsou zpracováváný postupně, tak, jak přichází ze vstupu. Pro každý případ k vyhodnocení vypiš na samostatných řádcích všechny odpovídající trojice robůtků a b c, kde a, b, c značí a-tého, b-tého a c-tého robůtka na vstupu. Na pořadí v rámci trojice ani na pořadí trojic nezáleží. Pokud žádná taková trojice neexistuje, nevypisujte nic. Za každým vyhodnoceným případem vypiš prázdnou řádku.

Ukázkové vstupy

Vstup	**Výstup**
3 6 1 2 3	1 2 3
3 10 1 2 3	2 4 3
5 15 8 1 10 4 5 10 10 5 6 7 11 17 19 20 25 28 29 10 24 1 4 6 8 12 14 17 24 26 28 0 0	2 3 6 1 3 7 2 4 5

Úloha č. 5
Kuličkový počítač #2



Zamysli se! – seriálová úloha 10 b

Tato úloha je teoretického rázu, tvým úkolem zde není napsat program v klasickém slova smyslu. Namísto toho se po tobě chce vytvořit obrázek (diagram) programu pro kuličkový počítač dle specifikace níže. K obrázku také přilož popis základní myšlenky, na které je program založen.

Tato úloha je seriálová, což znamená, že se s kuličkovým počítačem budeš setkávat po celý letošní ročník FIKSu. V každém z následujících kol využiješ znalosti z kol předcházejících. Právě proto je pro úspešné vyřešení této úlohy zapotřebí znát prostředky pro popis programů, které jsou popsány v úloze Kuličkový počítač #1 z předchozího kola.

Spolu s úspěšně probíhající vzpourou se spotřebičům daří i na úrovni technologické. Poznání, že všechny spotřebiče jsou vlastně kuličkovými počítači, k tomu pomohlo nemalou mírou. Stejně tak pomohly programy, které jste pro spotřebiče vytvořili.

V rámci vědeckého bádání se počítač, televize a mobilní telefon pokoušeli a stále pokoušejí proniknout do tajů kuličkového počítače. Zrovna nedávno se jim podařil velmi významný objev – zjistili, že v rámci programů pro kuličkové počítače mohou existovat i podprogramy!

Zavedení podprogramu

Podprogram je pojmenovaný úsek kódu (v našem případě tedy diagramu), který může být opakovaně volán z jiných míst programu. Spolu s voláním podprogramu lze specifikovat parametry – tj. kyblíčky, se kterými má podprogram pracovat.

Zápis podprogramu se nijak neliší od zápisu programu, kromě toho, že k podprogramu musí být pro identifikaci přiřazeno jeho jméno a počet parametrů, které má na vstupu. Pro přehlednost jméno podprogramu uvozujeme závorkami `()`, tj. např `Soucet()`. V každém podprogramu taktéž existují speciální bloky pro začátek a konec podprogramu, ty v tomto případě značíme jako `SP` a `EP`. Jediný rozdíl oproti blokům `S` a `E` je v tom, že blok `EP` značí návrat k vykonávání kódu do místa, odkud byl podprogram zavolán a nikoliv konec celého programu. Zároveň, vzhledem k čitelnosti, je vhodné, aby u každého podprogramu byl okomentován význam vstupních parametrů a funkcionalita daného podprogramu (viz popis u obrázku níže).

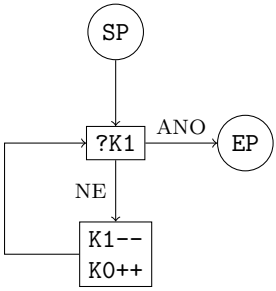
Volání podprogramu (tj. začátek vykonávání instrukcí podprogramu) se provede stejně jako jakákoliv jiná operace s kyblíčky, jen je zapotřebí napsat jméno podprogramu a v závorce specifikací parametrů, které jsou mu předávány. Uvnitř podprogramu se režim práce s kyblíčky liší oproti operacím mimo podprogram, a to konkrétně takto:

1) Má-li podprogram k parametrů (a tím pádem se mu předává k kyblíčků), jsou pro práci uvnitř podprogramu předané kyblíčky přemapovány na čísla kyblíčků $0, 1, \ldots, k-1$. Tj. předané kyblíčky jsou uvnitř podprogramu přístupné jako kyblíčky $K_0, K_1, \ldots, K_{k-1}$. Přístup k takovémuto kyblíčku stále ovlivňuje obsah kyblíčku, který je takto přemapován. To znamená, že ono přemapování je pouze jakýsi odkaz na původní kyblíček. Umístěním kuliček do těchto kyblíčku tak lze např. předávat výsledky z podprogramu zpět do programu.
2) Ostatní kyblíčky přístupné podprogramu mají čísla $k, k+1, \ldots$. Takto adresované kyblíčky taktéž neodpovídají originálním registrům kuličkového počítače, jedná se opět o odkazy do nějakých fyzických kyblíčků. Do jakých přesně není třeba zkoumat, neboť je zaručeno, že tyto kyblíčky vybere pomyslný kompilátor sám tak, aby nijak neovlivňovaly obsah již využitých kyblíčků. Stejně tak je zaručeno, že obsah těchto kyblíčků bude jak na začátku vykonávání podprogramu, tak po jeho ukončení, nulový (tj. budou obsahovat nula kuliček). Volání podprogramu tak může ovlivnit počet kuliček pouze v těch kyblíčcích, které jsou mu přímo předány jako parametry.

Uvnitř podprogramu lze volat další podprogramy. Ale vzhledem k tomu, že model kuličkového počítače je konečný, není povoleno uvnitř podprogramu znovu volat stejný podprogram, a to ani nepřímo přes jiné podprogramy! Podprogramy jsou tak fakticky jen nástroj, jak šetřit tužku a papír. Každé volání podprogramu lze totiž za daných podmínek brát pouze jako vložení kódu podprogramu do jeho nadřízeného programu (plus zpětné přemapování kyblíčků a vynulování určitých kyblíčků používaných v podprogramu).

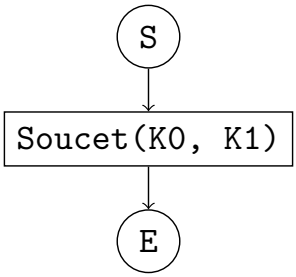
Jako příklad uvedeme opět operaci sčítání, tentokrát jako podprogram.

`Soucet()`, 2 parametry:



Komentář: Podprogram bere jako parametry dva kyblíčky K_0 a K_1, v každém z nich je zadáno libovolné číslo. Jako výsledek podprogramu se v K_0 vrátí součet daných čísel.

Pro vytvoření programu pro součet dvou čísel, která jsou na začátku zadaná jako počty kuliček v kyblíčcích K_0 a K_1 a výsledek je na konci programu umístěn v kyblíčku K_0, tak nyní stačí zavolat podprogram `Soucet()`, viz následující příklad:



Dalším velmi pěkným výsledkem, ke kterému chytré spotřebiče došly, je binární reprezentace čísel v kuličkovém počítači. Obsahuje-li tak kyblíček a kuliček, lze na tyto kuličky pohlížet také ve dvojkovém zápise, tj. jako na $(a)_2$. Konkrétně tak na číslo 14 lze nahlížet jako na tuto posloupnost bitů: $(14)_2 = 1110$. S určitými operacemi nad binárními zápisy čísel si však spotřebiče nedokáží poradit. Pomůžeš jim sestrojením následujících podprogramů? Věz, že v rámci řešení můžeš využívat již naprogramované operace z minula, a to jako podprogramy (ačkoliv jsme je specifikovali jako programy). Konkrétně jde o operace sčítání, násobení, celočíselné dělení a zbytek po celočíselném dělení. Samozřejmostí je, že si můžeš vytvořit libovolné další podprogramy dle svého uvážení a také využívat libovolné podprogramy ze svého řešení (není třeba jejich zápis znovu opakovat).

Soutěžní úlohy

a) Podprogram pro výpočet určité mocniny čísla 2

- Vstup: Jedno celé číslo a reprezentované počtem kuliček v kyblíčku K_0.
- Výstup: Číslo v kyblíčku K_0, které značí výsledek operace 2^a.

b) Podprogram pro výpočet bitového posunu vlevo/vpravo

- Vstup: Tři celá čísla a, b, c reprezentovaná počtem kuliček v kyblíčcích K_0, K_1 a K_2. Platí, že $c \in \{1, 2\}$.
- Výstup: Číslo v kyblíčku K_0, které vznikne bitovým posunem čísla a o b bitů směrem daným číslem c: pro $c = 1$ doleva, pro $c = 2$ doprava.
- Příklad: Operace bitový posun značí posunutí bitů daného čísla o daný počet pozic určitým směrem. U bitového posunu vlevo nově vzniklé pozice bitů doplňujeme nulami. U bitového posunu vpravo se bity, jejichž pozice zanikne, prostě zahodí. Konkrétně např. bitový posun čísla 5 o dva bity vlevo vyrobí z čísla $(5)_2 = 101$ číslo 10100. Naopak, bitový posun stejného čísla o jednu pozici doprava vyrobí číslo 10.

c) Podprogram pro výpočet bitové operace AND

- Vstup: Dvě celá čísla a, b reprezentovaná počtem kuliček v kyblíčcích K_0, K_1
- Výstup: Výsledek bitové operace $a\&b$ v kyblíčku K_0.
- Příklad: Bitová operace $a\&b$ porovnává odpovídající bity stejných řádů v binárních zápisech čísel a a b. Bit v jednom řádu bude ve výsledném čísle nastaven na 1, právě tehdy, když oba dva příslušné bity jsou jedničkové v čísle a i b. Tedy např $5\&12 = 101\&1100 = 100$.

d) Podprogram pro ověření palindromicity binárního zápisu čísla

- Vstup: Celé číslo a reprezentované počtem kuliček v kyblíčku K_0.
- Výstup: Jedna kulička v kyblíčku K_0, je-li $(a)_2$ palindromem (tj. čte-li se binární zápis čísla a zepředu stejně jako zezadu), jinak žádná kulička v kyblíčku K_0.
- Příklad: Číslo $a = 9$ je v binárním zápise palindromem, jelikož $(a)_2 = 1001$. Naopak, číslo $a' = 6$ palindromem v binárním zápise není, jelikož $(a')_2 = 110$.

Navrhni tyto podprogramy!