Soutěž dětí a mládeže v programování

Obvodní kolo Prahy 1 a 2, rok 2021

Autoři: Adam Benda, Pavel Hübner, Martin Horský; verze 1.01, 2021-02-10

Každá úloha má koeficient obtížnosti, kterým se násobí hodnocení vašeho řešení.
Odevzdejte nám prosím řešení přes Google Classroom jako komprimovanou složku zip.
Krom splnění zadané funkčnosti se hodnotí přehlednost vašeho kódu (komentáře, popisující názvy proměnných) a efektivita vašeho řešení.

Při soutěži je zakázána komunikace (přítel na telefonu). Vyhledávač a Internetové zdroje používat můžete ALE: cílem je zjistit vaši schopnost algoritmizace. Zkopírovat si několik řádků kódu ze stackoverflow (uveďte zdroj) nebo použít knihovní funkci řešící dílčí část problému je v pořádku. Zkopírovat odněkud hotové řešení celé úlohy není přípustné. V případě pochybností konzultujte porotce.

Používání knihoven je povoleno, jelikož mnoho programovacích jazyků má relativně omezenou standardní knihovnu (např. grafické zobrazování v okýnkách). Nepoužívejte ale knihovny, které víceméně vyřeší úlohu za vás.

Testovací vstupy k úlohám naleznete v classroomu

Úloha 1: Akvárium (koef. 2)

Vytvořte program vykreslující akvárium s rybičkami. Uživatel zadá počet rybiček. Zobrazte akvárium – nehybné za méně bodů, animované za více bodů. V případě animovaného se rybička posouvá v náhodném směru (i šikmo). Rybičky nesmí vyplout z akvária (obrazovky). V případě animovaného, rybičku zobrazujte střídavě alespoň třemi různými snímky (například vlnění ploutví).
Doporučujeme použít přiložené grafické podklady (png obrázky), přípustné je ale i řešení v textovém režimu :)

Úloha 2: Kontrola časoprostových tras (koef. 4)

Úřad pro cestování časem pravidelně navrhuje síť tras pro cestování v čase i v prostoru, aby bylo možné přicházet na schůze včas, ačkoliv by cestující zaspal. Určuje tedy trasy mezi místy s nějakým posunem v čase. Je tedy možné přesunout se z Prahy do Brna za -3 hodiny a z Brna do Prahy za 4 hodiny. Tedy tam a zpátky a 1 hodinu. Je ale nutné, aby v rámci této sítě nebylo možné cestovat nekonečně daleko do minulosti. To je možné právě tehdy, pokud je možné se dostat z nějakého místa tam a zpátky za záporný čas (nebereme v úvahu, že někdo rychle

přeběhne mezi městy pěšky). Vaším úkolem bude zkontrolovat, zda v zadané síti není takováto "záporná smyčka", a pokud ano, vypsat ji. Pokud síť obsahuje více záporných smyček, vypište jednu z nich. Program by zároveň měl být relativně rychlý.

Vstup:

V podobě textu buď na standardním vstupu nebo ze souboru:

- Na prvním řádku seznam měst (jednoslovné názvy oddělené mezerami)
  - Praha Brno Ostrava
- Na dalších řádcích seznam spojů ve formátu "START CÍL POSUN_V_ČASE"
  - Praha Brno -3
  - Brno Praha 4
  - Brno Ostrava -2
  - Ostrava Praha 4

Můžete se spolehnout, že posuny v čase jsou celá čísla a nebudou astronomicky velká a můžete předpokládat korektní vstup.

Výstup:

Jeden z následujících:

- Síť neobsahuje zápornou smyčku
- Síť obsahuje zápornou smyčku
- Síť obsahuje smyčku s posunem -1
- Síť obsahuje smyčku s posunem -1:
  Praha Brno -3
  Brno Ostrava -2
  Ostrava Praha 4

Úkoly a hodnocení:

1. Určit, zda síť obsahuje zápornou smyčku (60% hodnocení)
2. Vypsat posun záporné syčky (10% hodnocení)
3. Vypsat všechny spoje, které zápornou smyčku tvoří (30% hodnocení)

Úloha 3: Hra Timeline (koef. 2)

Vytvořte hru Timeline pro jednoho hráče.

Vstupem bude balíček kartiček (v textovém souboru, kódování UTF-8), reprezentující události z historie - např:

```
1492,Objevení Ameriky
1918,Konec 1. Světové války
```

V každém kole hry si hráč vezme jednu kartičku, kterou umisťuje. Vidí však pouze název události a nikoliv její letopočet. Hráč musí umístit kartičku do správného pořadí na existující časové ose (mezi kartičky na stole, nebo na kraj). Po přiložení se ukáže letopočet a vyhodnotí se: pokud byla kartička přiložena správně (letopočty jsou ve správném pořadí od menšího k většímu), kartička zůstává na stole. Pokud byla kartička přiložena nesprávně, ze stolu se maže a hráč dostává záporný bod. Cílem hráče je, mít na konci hry (po prolízání všech kartiček z balíčku) co nejméně záporných bodů (tedy co nejvíce správně umístěných událostí).

Umožněte hru s libovolným počtem kartiček (ze vstupního souboru, méně než 100 událostí) a kartičky na začátku hry náhodně zamíchejte.
Hra by měla zobrazovat zbývající počet kartiček a vypořádat se s nevalidním vstupem elegantněji, než pádem (nevalidní řádky ignorujte, nebo po načtení souboru zobrazte, že je vstupní soubor nevalidní).
Letopočty jsou celá čísla větší než -10000 a menší než 2100

Úloha 4: Samoobslužná pokladna (koef. 1,5)

Obchodní řetězec Bylla si u vás objednal vytvoření systému pro automatické odbavování zákazníků.
Zákazník po jednom skenuje čárové kódy zboží a po pokládá je na kontrolní váhu. Po načtení posledního zboží zákazník zaplatí útratu na platebním terminálu.
Vy máte naprogramovat jádro pokladny, které dostává zprávy (jako řádky textu):

1) SKEN [cena_zbozi] [hmotnost_zbozi]
    - Zákazník naskenoval jednu položku svého nákupu
2) VAHA [celkova_hmotnost_na_vaze]
    - Změnila se hmotnost detekovaná na kontrolní váze
3) PLATBA [zaplacena_suma]
    - Zákazník zaplatil na platebním terminálu

Po každé akci SKEN následuje jedna akce VAHA.
Na konci nákupu je právě jedna akce PLATBA.
Přírůstek navážené hmotnosti musí odpovídat hmotnosti položky z akce SKEN +-5%
Zaplacená suma akce PLATBA musí být stejná jako je součet cen všech položek.
Po každé korektní akci VAHA vypište OK.
Po korektní akci SKEN vypište OK [celkova_cena_zbozi]
Po korektní akci PLATBA vypište OK a účtenku (seznam zboží z nákupu) - a běh programu ukončete.
Po nekorektní akci (váha nesouhlasí, placená částka nesouhlasí, neprávné pořadí akcí) vypište ERROR, čímž přivoláte obsluhu, a běh programu ukončete.

V příloze naleznete několik testovacích nákupů a očekávaný výstup. Nezapomeňte, že ERROR musíte hlásit už ve chvíli, kdy problém vznikne (načítáte ze standardního vstupu po jednotlivých řádcích) – řešení, které vstup čte ze souboru, bude penalizováno.