1. zadatak — objašnjenje bodovanja i zabrane nizova

1. Povod za cijelu priču

Mojom greškom (S. Singer — nisam provjeravao rješenje) na službenom webu kolegija je objavljeno rješenje 1. zadatka od prošle godine (2015/16) iz kojeg se može zaključiti

- da je **dozvoljeno** korištenje nizova u rješenju — u prvoj varijanti rješenja koristi se niz fiksne duljine od 100 elemenata.

Ta varijanta s nizovima je **pogrešna** i **zabranjena** (v. malo niže).

2. Bodovanje i ispravak rezultata

Nažalost, svi studenti koji su "učili" po tom rješenju mogli su zaključiti da se rješenje s nizovim tretira kao, ne samo dozvoljeno, nego još i **ispravno**. U tom smislu, greška je naša, i zato ispravak rezultata na 1. zadatku

- **snižavanjem** kriterija "za prolaz" s 80% (16 bodova) na 65% (13 bodova).

Međutim, da ne bismo nastavili s nekorektnim bodovanjem pogrešnih rješenja,

- zarađeni bodovi **ostaju** kao u početnoj varijanti rezultata.

Dakle, bodove nećemo "dizati", tj. ostaje **kazna** za korištenje nizova. Ta kazna je bitno premala, jer rješenje s nizovima **ne valja** i stvarno vrijedi 0 bodova. Tako je bilo ranijih godina i tako mislimo postupiti ubuduće.

3. Zašto su nizovi zabranjeni — ili, "što je pogrešno"?

Za početak, u uputi (na istoj stranici gdje je 1. zadatak) lijepo piše da je "zabranjena upotreba dodatnih nizova, osim ako je u zadatku drugačije navedeno".

- Pod pojmom "dodatni" se misli na nizove koji se **ne spominju** u tekstu zadatka (to je, valjda, očito).

Ni ove, ni prošle godine, u 1. zadatku se **ne spominju** nikakvi nizovi, i **nema** dozvole korištenja "dodatnih" nizova. To je vrlo namjerno,

- jer se zadatak može riješiti i **bez** nizova, i baš takvo rješenje tražimo,
- a definiranje bilo kojeg niza **konstantne** duljine **ne radi** za **svaki** dozvoljeni ulaz, tj. za svaku dozvoljenu bazu b.

Naime, jedan od ciljeva kolegija Programiranje 1 i 2 je pisanje "**matematički**" korektnih algoritama, koji

- korektno rade za **svaki** dozvoljeni ulaz — u matematičkom smislu (prirodni, cijeli ili realni brojevi, tj. beskonačni skupovi),
- bez obzira na to što konkretna implementacija algoritma u nekom programskom jeziku (poput C-a), postavlja i dodatna ograničenja — poput prikaza brojeva u računalu (skupovi dozvoljenih ili mogućih ulaznih vrijednosti mogu postati konačni).

Što to znači — ilustrirajmo primjerom 1. zadatka s drugog kolokvija prošle i ove godine. Bitni dio teksta zadatka glasi (pisano prema ovogodišnjoj verziji):

> Za prirodni broj n kažemo da je "*nekakav*" (*sveznamenkast* ili *pravilan*) bazi b ako vrijedi "neko svojstvo" za (sve) znamenke broja n u bazi b.

(a) Napišite funkciju `nekakav` (`sveznamenkast` ili `pravilan`) koja prima prirodne brojeve n i b te vraća jedinicu ako je n "nekakav" (sveznamenkast ili pravilan) u bazi b, a nulu u suprotnom.

Dakle, "matematički" rečeno, **ulazni** argumenti funkcije su

$$n \in \mathbb{N}, \qquad b \in \mathbb{N} \setminus \{1\},$$

tj. pretpostavljamo da je $b \geq 2$ (zbog teorema o prikazu prirodnog broja n u bazi $b \geq 2$). Uočiti da nema **gornje** granice na n i b!

Ako baš "cjepidlačimo", moglo bi se dozvoliti i $b = 1$, a onda broj n ima točno n znamenki jednakih nula u bazi $b = 1$. No, to nije bitno i nitko razuman neće gledati što funkcija (ili program) radi za $b = 1$. Inače bismo u tekstu napisali što treba napraviti za $b = 1$. Dakle, slobodno je pretpostaviti da je $b \geq 2$.

Funkcija `nekakav` radi **matematički** korektno, ako i samo ako je sljedeća tvrdnja **istinita**:

> Za **svaki** $n \in \mathbb{N}$ i za **svaku** bazu $b \in \mathbb{N} \setminus \{1\}$, funkcija `nekakav(n, b)` vraća korektan rezultat.

Naglasak je na kvantifikatorima "za svaki", uz matematičke pretpostavke na dozvoljeni ulaz, jer nikakve druge pretpostavke nisu navedene u tekstu zadatka.

3.1. Rješenje s nizom (sa službenog weba)

U varijanti s korištenjem nizova, rješenje prve grupe 1. zadatka iz prošle godine izgleda ovako (umjesto `n`, ovdje piše `x`):

```
/* provjerava sadrzi li x u bazi b sve znamenke od 0 do b-1 tocno jednom */
/* rjesenje s nizovima */
int sveznamenkast(int x, int b)
{
    int znamenke[100] = {0};
    int brZnam = 0;

    while (x > 0)
    {
        /* ako je broj vec imao ovu znamenku, ne moze biti sveznamenkast */
        if (znamenke[x%b])
            return 0;

        znamenke[x%b] = 1;
        brZnam++;
        x /= b;
    }

    /* ako imamo tocno b znamenki, broj je sveznamenkast,
       jer su sve morale biti razlicite */
    return (brZnam == b);
}
```

Polje `znamenke` služi pamćenju je li se određena znamenka `x%b` (= indeks elementa), **pojavila** u broju ili ne. Za ovogodišnji zadatak, u tom polju bi se pamtilo **koliko puta** se određena znamenka javlja u broju (element niza bi služio kao brojač).

Uočite da, za zadanu bazu b, znamenke = indeksi mogu ići od 0 do $b - 1$ (= najveća moguća znamenka u bazi b).

Međutim, polje je deklarirano tako da ima 100 elemenata, tj. najveći dozvoljeni indeks je 99. Dakle, ako želimo da ova funkcija radi korektno, tj. ne "gazi" po memoriji preko rezerviranog dijela,

- onda zadana baza b **mora** zadovoljavati $b \leq 100$.

Takva pretpostavka **ne piše** u zadatku. Dakle, ovo rješenje **ne radi** korektno za bilo koju bazu $b > 100$. To je jako **daleko** od korektnosti za svaki dozvoljeni ulaz, tj. od

- "za **svaki** $b \geq 2$" (matematička korektnost),
- odnosno, "za svaku **prikazivu** bazu $b \geq 2$" (tzv. implementacijska korektnost).

Ako vam se čini da bi rješenje s "duljim" nizom (recimo, 10000) bilo bolje, imate krivo — matematički je ekvivalentno neprihvatljivo. I dalje je sasvim jednostavno "skršiti" funkciju.

Osim toga, ako pokušate staviti "dovoljno veliku" duljinu (recimo, 10^9), onda će i pokušaj implementacije odmah "krepati":

- ili će kompajler/linker javiti grešku zbog premalog programskog stoga (polje ne stane u njega),
- ili će raspoložive memorije biti premalo, čak i kad se polje proba dinamički alocirati (v. Prog2).

Kako god okrenuli, **ne radi** korektno, jer nema gornje ograde na b. Kad bi takva pretpostavka bila dozvoljena, onda bi u zadatku pisalo

- "Smijete pretpostaviti da za bazu b vrijedi $b \leq b_{\max}$", gdje je $b_{\max}$ neka **konkretna** zadana vrijednost, poput 100.

Zato što **ne piše** $\implies$ **nije dozvoljeno** postaviti dodatnu pretpostavku o gornjoj ogradi na bazu b. Naravno, isto vrijedi i za n.

Dodatno, rješenje s nizovima **krši** zabranu korištenja dodatnih nizova (piše u uputi). Ipak smo neki matematičari, pa nećemo odmah "skočiti sami sebi u usta". Kad bi dodatni niz bio dozvoljen, u zadatku bi pisalo još i

- "Dozvoljeno je korištenje jednog dodatnog niza duljine te i te".

Dakle, nije dozvoljen, jer **ne piše**.

4. Završna napomena

Na kraju, da ne biste pomislili da pretjerano "logičarski" ili "formalno" inzistiramo na korektnosti algoritama,

- precizno logičko zapisivanje pretpostavki i iskaza korektnosti pojedinih dijelova programa je osnovni dio tzv. "formalnih" metoda u matematičkom računarstvu.

Ove "formalne" metode se zaista koriste u **praksi** za automatsku **verifikaciju** algoritama i programa!