test-e.ro

CAIET ELEV

Lecția 16

Cazuri particulare ale listei: stiva (LIFO) și coada (FIFO). Lista de frecvențe

*Aceeași listă, alte reguli. Astăzi nu învățăm o structură nouă, ci trei moduri de a folosi una pe care o cunoaștem: un indice care urcă și coboară face o **stivă**, doi indici care doar cresc fac o **coadă**, iar o listă în care **poziția însăși e informație** face o **listă de frecvențe**.*

DISCIPLINA Informatică **CLASA** a IX-a · mate-info **LIMBAJ** Python 3

NUMELE ELEVULUI .....

CUM FOLOSEȘTI CAIETUL

Rezolvă exercițiile pe hârtie: urmărești execuția pas cu pas, prezici ce afișează un program și scrii singur soluțiile. Din clasa `list` folosim în continuare doar indexarea `v[i]`, `len(v)` și crearea listei cu `[0] * n` — **fără** `append`, `pop`, `in`, `max` sau alte metode (acelea vin la L17). Fără `def`. Stiva și coada se **simulează**: lista rămâne aceeași, indicii sunt cei care le definesc. Regula de aur: **rezultatul îl calculezi tu, înainte de a rula.**

CUPRINS

1 Ce știm deja despre liste .....
2 Stiva: ultimul intrat, primul ieșit .....
3 Coada: primul intrat, primul ieșit .....
4 Stiva în lucru .....
5 Coada în lucru .....
6 Lista de frecvențe .....
7 Cod ghidat · cinci probleme .....

SECȚIUNEA 1 · REACTUALIZARE L15

1 Ce știm *deja* despre liste

Lecția trecută am învățat ce este o listă, cum ajungem la un element după poziția lui și cum o parcurgem cu indice. Reluăm exact aceste trei unelte — pe ele se construiește tot ce urmează.

Crearea unei liste de dimensiune fixă. Când știm câte elemente vor fi, dar nu și valorile, pregătim „sertarele" cu `[0] * n` și le umplem prin indexare.

```
v = [0] * n
for i in range(n):
    v[i] = int(input())
```

Accesul prin indice. `v[0]` este primul element, `v[len(v) - 1]` este ultimul; indicii valizi merg de la 0 la `len(v) - 1`, iar `v[len(v)]` este eroare. **Parcurgerea cu indice.** `for i in range(len(v))` produce exact indicii valizi.

Un element important pentru lecția de azi: **un indice nu trebuie neapărat să parcurgă toată lista**. Putem folosi o variabilă care ține minte o singură poziție și pe care o mutăm noi, când vrem. Exact pe această idee se construiesc stiva și coada.

> **ATENȚIE · POZIȚIE ≠ VALOARE**
>
> Pentru `v = [15, 4, 23, 8, 42]`, valoarea 23 se află pe poziția 2. Când citești un enunț, întreabă-te de fiecare dată dacă ți se cere un **indice** (`i`) sau un **element** (`v[i]`).

A EXERCIȚIUL A · GRILĂ

Recapitulare L15

GRILĂ

Încercuiește răspunsul corect la fiecare întrebare.

1. Pentru `v = [10, 20, 30, 40]`, ce afișează `print(v[1])`? (10) (20) (30) (40)
2. Ce construiește instrucțiunea `v = [0] * 4`?
   (un singur element, 0) (o listă goală) (patru elemente, toate 0) (elementele 0, 1, 2, 3)
3. Pentru o listă `v` cu 6 elemente, care este ultimul indice valid? (6) (5) (7) (1)
4. Ce produce `range(len(v))` pentru o listă cu 3 elemente?
   (0, 1, 2) (1, 2, 3) (0, 1, 2, 3) (chiar elementele listei)

HINT *numerotarea pozițiilor începe de la 0, deci ultimul indice este* `len(v) - 1`*, iar* `range(len(v))` *livrează exact indicii valizi.*

SPRIJIN · SCRIE CODUL

Numere din două cifre egale

SCRIERE COD

Se citește un număr natural `n`, apoi `n` numere naturale, fiecare de pe câte un rând. Afișează câte numere formate din **două cifre egale** există în listă. Ex.: pentru `n = 5` și numerele 555, 44, 23, 88, 42 se afișează `2` (44 și 88).

HINT *un număr are exact două cifre dacă este între 10 și 99; cifrele lui sunt egale dacă* `v[i] // 10 == v[i] % 10`*. Amândouă condițiile trebuie verificate.*

SOLUȚIA TA · PYTHON

SPRIJIN · SCRIE CODUL

Lista în ordine inversă

SCRIERE COD

Se citește un număr natural `n`, apoi `n` numere întregi, fiecare de pe câte un rând. Afișează elementele listei în ordine inversă față de cea în care au fost citite, fiecare pe câte un rând. Ex.: pentru `n = 3` și 5, 8, 2 se afișează 2, apoi 8, apoi 5.

HINT *parcurge normal, cu* `for i in range(len(v))`*, și afișează elementul de pe poziția* `len(v) - 1 - i`*: la* `i = 0` *acesta e ultimul, la* `i = 1` *penultimul.*

SOLUȚIA TA · PYTHON

2 SECȚIUNEA 2 · STIVA (LIFO)

Stiva: ultimul intrat, *primul ieșit*

Farfuriile din bucătărie și butonul „Undo" funcționează după aceeași regulă: iese mereu cel care a intrat ultimul. **Stiva** este un caz particular al listei — nu se deosebește prin cum arată, ci prin **regula după care intră și ies elementele**: iese întotdeauna elementul pus **cel mai recent**.

> **STIVA (LIFO)**
>
> Prescurtarea vine din engleză: **Last In, First Out** — ultimul intrat, primul ieșit. Elementele intră și ies **prin același capăt**, numit **vârf**. Cele două operații se numesc „pune" (push) și „ia" (pop).

O simulăm cu o listă de dimensiune fixă și **un singur indice**, `vf`, care spune **câte elemente sunt** în stivă și, în același timp, **unde se scrie următorul**: la **punere** scriem la poziția `vf`, apoi creștem `vf`; la **luare** scădem întâi `vf`, apoi citim de la noua poziție. Stiva e **goală** când `vf` este 0.

```
1  st = [0] * 5
```

```
vf = 0
st[vf] = 3
vf = vf + 1
st[vf] = 7
vf = vf + 1
st[vf] = 2
vf = vf + 1
vf = vf - 1
print(st[vf])
```

Codul afișează 2 — ultimul pus, primul luat. Pas cu pas: după cele trei adăugări `vf` este 3 și lista conține `[3, 7, 2, 0, 0]`; scăzând `vf` la 2, ajungem exact la elementul pus ultimul.

> **ATENȚIE**
>
> Observă că nu „ștergem" nimic: valoarea 2 rămâne scrisă în listă și după ce a fost luată. Ce s-a schimbat e doar `vf`, care a coborât. **Indicele e cel care definește stiva, nu conținutul listei.** O soluție care încearcă să „șteargă" elementul e greșită din start — nu avem cu ce, și nici nu e nevoie.

A

EXERCIȚIUL A · DRY-RUN

Cum se mișcă vârful stivei

DRY-RUN

Operațiile din prima coloană sunt date. Completează, după fiecare, valoarea indicelui `vf` și conținutul stivei de la bază spre vârf (elementele separate printr-un spațiu; scrie `(goală)` dacă nu are elemente). Primul rând e completat ca exemplu.

```
st = [0] * 5
vf = 0
```

operație (dată)	vf	conținutul stivei
(start)	0	(goală)
pune 3		
pune 7		
pune 2		
ia		
ia		

HINT *la adăugare scrii întâi la poziția `vf`, apoi crești `vf`; la eliminare scazi întâi `vf`, apoi citești. După cele două luări, în stivă rămâne un singur element.*

B

EXERCIȚIUL B · COMPLETEAZĂ

Regula stivei

COMPLETARE

Completează golurile folosind cuvintele din rezervă (trei cuvinte rămân nefolosite).

`recent` `devreme` `același` `celălalt` `vârf` `bază`

Într-o stivă iese întotdeauna elementul intrat cel mai _____ , pentru că elementele intră și ies prin _____ capăt, numit _____ .

HINT *diferența față de coadă stă chiar în al doilea gol: la stivă intrarea și ieșirea se fac prin același loc.*

C

EXERCIȚIUL C · SCRIE CODUL

Ultima acțiune

SCRIERE COD

Se citește un număr natural `n`, cel puțin egal cu 1, apoi `n` numere naturale, fiecare de pe câte un rând — codurile acțiunilor făcute de utilizator, în ordinea în care s-au petrecut. Simulează o stivă și afișează codul acțiunii care ar fi anulată de un „Undo". Ex.: pentru `n = 3` și 5, 8, 2 se afișează `2`.

HINT *simulează stiva cu `st = [0] * n` și un indice `vf` — nu folosi metode ale listei. Undo ia elementul din vârf: scazi `vf` cu 1, apoi afișezi `st[vf]`.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 3 · COADA (FIFO)

3 Coada: primul intrat, *primul ieșit*

La ghișeu nu iese primul cel care a venit ultimul. **Coada** e al doilea caz particular al listei și e opusul stivei ca regulă: iese întotdeauna elementul care **așteaptă de cel mai mult timp**. Imaginea e coada de la ghișeu sau mașinile oprite la semafor pe sens unic — cine a venit primul pleacă primul, nimeni nu sare peste.

COADA (FIFO)

Prescurtarea vine din engleză: **First In, First Out** — primul intrat, primul ieșit. Spre deosebire de stivă, elementele intră **printr-un capăt** și ies **prin celălalt**. De aceea coada are nevoie de **doi** indici: `prim` (de unde iese) și `ultim` (unde intră). Coada e goală când `prim` este egal cu `ultim`.

```
cd = [0] * 5
prim = 0
ultim = 0
cd[ultim] = 3
ultim = ultim + 1
cd[ultim] = 7
ultim = ultim + 1
```

```
8  print(cd[prim])
```

Codul afișează 3 — primul intrat, primul ieșit. Atenție însă la ce face exact ultima linie: ea doar **se uită** la elementul care urmează să iasă, fără să-l scoată. Ca elementul să iasă cu adevărat, ar mai trebui o linie, `prim = prim + 1`. Observă contrastul cu stiva: acolo un singur indice urca și cobora; aici **ambii indici doar cresc**, iar coada „înaintează" prin listă.

> **ATENȚIE**
>
> Nu confunda `prim` cu „primul element al listei". `prim` e **poziția** celui care urmează să iasă și se mărește pe măsură ce oamenii sunt serviți. După două ieșiri, `prim` este 2 — chiar dacă `cd[0]` și `cd[1]` încă există în listă, ei nu mai fac parte din coadă.

A

EXERCIȚIUL A · DRY-RUN

Cei doi indici ai cozii

DRY-RUN

Aceleași elemente ca la stivă (3, 7, 2), altă regulă. Operațiile sunt date; completează, după fiecare pas, valorile `prim` și `ultim` **așa cum arată ele după executarea operației** — reține că `ultim` este poziția pe care va intra următorul element, deci după o intrare el a crescut deja. În coloana „iese" scrie elementul care iese, iar acolo unde nu iese nimic scrie semnul minus.

```
1  cd = [0] * 5
2  prim = 0
3  ultim = 0
```

operație (dată)	prim	ultim	iese
(start)	0	0	–
intră 3			
intră 7			
intră 2			
iese			
iese			

HINT *la intrare crește doar `ultim`, la ieșire crește doar `prim`, iar cei doi indici nu scad niciodată. Primele care ies sunt 3, apoi 7.*

B

EXERCIȚIUL B · ASOCIERE

Stivă sau coadă?

ASOCIERE

Scrie în căsuță litera modelului potrivit pentru fiecare situație reală, împreună cu motivul.

☐ Butonul „Anulează ultima acțiune" (Ctrl+Z)

☐ Documentele trimise la imprimantă

☐ Teancul de farfurii din bucătărie

☐ Mașinile oprite la semafor pe sens unic

A Coadă — cine a ajuns primul pleacă primul, iar ceilalți nu îl pot depăși

B Stivă — se poate lua doar elementul aflat cel mai sus

C Coadă — cererile se rezolvă în ordinea în care au sosit

D Stivă — se desface întâi ce s-a făcut cel mai recent

HINT *o singură întrebare: cine iese primul — cel care a intrat ultimul sau cel care așteaptă de cel mai mult timp?*

C

EXERCIȚIUL C · SCRIE CODUL

Primul la ghișeu

SCRIERE COD

Se citește un număr natural `n`, cel puțin egal cu 1, apoi `n` numere naturale, fiecare de pe câte un rând — numerele de ordine ale persoanelor, în ordinea în care au ajuns la ghișeu. Simulează o coadă și afișează numărul persoanei care va fi servită prima. Ex.: pentru `n = 3` și 5, 8, 2 se afișează `5`.

HINT *simulează coada cu o listă și doi indici, `prim` și `ultim` — intrările cresc `ultim`, iar cel care iese primul se află la poziția `prim`, adică 0.*

SOLUȚIA TA · PYTHON

SECȚIUNEA 4 · STIVA ÎN LUCRU

4 Stiva *în lucru*

Simulările de la S2 au avut o formă simplă: întâi toate adăugările, apoi o eliminare. În practică, operațiile vin amestecate — un utilizator face trei acțiuni, apasă Ctrl+Z o dată, mai face o acțiune, mai apasă Ctrl+Z de două ori. Mecanismul nu se schimbă: la **punere** scriem la poziția `vf`, apoi `vf = vf + 1`; la **luare** `vf = vf - 1`, apoi citim de la noua poziție.

Apare însă o situație nouă: **ce facem când se cere o eliminare dintr-o stivă goală?** Dacă `vf` este 0 și scădem, ajungem la -1, iar `st[-1]` în Python nu dă eroare — dă ultimul element al listei, ceea ce e complet greșit. De aceea eliminarea se face **numai dacă** stiva are elemente:

```
1  if vf > 0:
2      vf = vf - 1
```

DE REȚINUT

Numărul de elemente rămase în stivă la final este chiar `vf`. Nu trebuie să-l numeri separat: indicele care ține minte unde se scrie următorul element spune, în același timp, câte elemente sunt deja acolo.

ATENȚIE

`st[-1]` **nu** este o eroare în Python — este ultimul element al listei. De aceea o stivă care coboară sub zero nu se plânge, ci începe să întoarcă valori din coada listei. Este exact tipul de greșeală care trece de teste superficiale și cade la cazul-limită.

A

EXERCIȚIUL A · DRY-RUN

Undo-uri amestecate cu acțiuni

DRY-RUN

Operațiile citite sunt date, în ordine: o valoare nenulă e o acțiune nouă, iar 0 e un Undo (fără efect pe stivă goală). Completează, după fiecare pas, valoarea lui `vf` și conținutul stivei. Primul rând e completat ca exemplu.

x citit (dat)	ce se întâmplă	vf	conținutul stivei
3	pune 3	1	3
7			
0			
0			
0			
2			

→ Câte acțiuni rămân pe stivă la final? _____

HINT *al treilea Undo găsește stiva goală, deci nu are niciun efect — `vf` rămâne 0, nu coboară la -1.*

B

EXERCIȚIUL B · SCRIE CODUL

Câte acțiuni rămân după Undo-uri

SCRIERE COD

Se citește un număr natural `n`, apoi `n` numere naturale, fiecare de pe câte un rând, reprezentând operațiile făcute într-un editor: o valoare nenulă înseamnă o acțiune nouă, care se pune pe stivă, iar valoarea 0 înseamnă un Undo, care ia acțiunea din vârf. Un Undo cerut când stiva este goală nu are niciun efect. Afișează câte acțiuni rămân pe stivă la final. Ex.: `n = 5` și 3, 7, 0, 2, 0 → `1`.

HINT *protejează eliminarea cu `if vf > 0:`. Răspunsul final este chiar `vf` — nu trebuie numărat separat.*

SOLUȚIA TA · PYTHON

EXERCIȚIUL C · REPARĂ CODUL

Cea mai mare valoare rămasă pe stivă

SCRIERE COD

Aceleași date de intrare ca mai sus (valoare nenulă = acțiune pusă pe stivă, 0 = Undo, Undo pe stivă goală fără efect), dar altă cerință: **cea mai mare valoare rămasă pe stivă la final**; dacă stiva e goală, se afișează 0. Ex.: pentru `n = 3` și 6, 9, 0 se afișează `6` — 9 a fost pus, dar apoi anulat. Programul de mai jos face corect eliminarea, dar greșește la căutarea maximului: uneori afișează o valoare care fusese deja anulată. Găsește **o singură** greșeală și rescrie linia corectată.

```
n = int(input())
st = [0] * n
vf = 0
for i in range(n):
    x = int(input())
    if x != 0:
        st[vf] = x
        vf = vf + 1
    else:
        if vf > 0:
            vf = vf - 1
maxim = 0
for i in range(n):
    if st[i] > maxim:
        maxim = st[i]
print(maxim)
```

HINT *după Undo-uri, lista* `st` *încă păstrează vechile valori pe pozițiile de peste* `vf` *— ele nu se șterg, doar nu mai fac parte din stivă. Până unde ar trebui deci să meargă căutarea maximului?*

LINIA CORECTATĂ · PYTHON

SECȚIUNEA 5 · COADA ÎN LUCRU

5 Coada *în lucru*

Și la coadă, în realitate, intrările și ieșirile se amestecă: la ghișeu sosesc oameni în timp ce alții sunt serviți. Mecanismul rămâne cel de la S3 — la **intrare** scriem la poziția `ultim`, apoi `ultim = ultim + 1`; la **ieșire** citim de la poziția `prim`, apoi `prim = prim + 1`. Și aici e nevoie de protecție: coada e goală când `prim` este egal cu `ultim`, deci ieșirea se face numai dacă `prim < ultim`.

Apare însă o întrebare care nu există la stivă: **câți oameni așteaptă acum?** La stivă răspunsul era chiar `vf`. La coadă, niciunul dintre cei doi indici nu e răspunsul — pentru că amândoi doar cresc. Numărul de elemente aflate în coadă este **diferența**:

```
1  print(ultim - prim)
```

Intuiția e simplă: `ultim` numără câți au intrat vreodată, `prim` numără câți au ieșit vreodată, iar cine e încă acolo e diferența dintre ele.

CONTABILITATEA CELOR DOUĂ MODELE

La **stivă**, un singur indice spune și unde se scrie, și câte elemente sunt. La **coadă**, indicii spun unde se scrie și de unde se citește, iar numărul de elemente e `ultim - prim`. Amândoi indicii cozii doar cresc; ei nu se întorc niciodată înapoi.

ATENȚIE

Pentru că `ultim` doar crește, o coadă simulată așa **consumă poziții** din listă chiar și când în ea așteaptă puțini oameni: locurile părăsite după ieșiri nu se refolosesc. De aceea lista trebuie să aibă cel puțin `n` poziții, câte intrări pot exista în total.

A

EXERCIȚIUL A · SCRIE CODUL

Câte persoane mai așteaptă

SCRIERE COD

Se citește un număr natural `n`, apoi `n` numere naturale, fiecare de pe câte un rând, reprezentând evenimentele de la un ghișeu: o valoare nenulă înseamnă că a sosit o persoană cu acel număr de ordine, iar valoarea 0 înseamnă că o persoană a fost servită și a plecat. O plecare cerută când coada este goală nu are niciun efect. Afișează câte persoane mai așteaptă la final. Ex.: `n = 5` și 3, 7, 0, 2, 0 → `1`.

HINT *numărul de persoane care așteaptă este `ultim - prim`, nu `ultim` și nici `prim`. Protejează ieșirea cu `if prim < ultim:`.*

SOLUȚIA TA · PYTHON

B

EXERCIȚIUL B · COMPLETEAZĂ

Stivă și coadă, față în față

COMPLETARE

Completează golurile folosind cuvintele din rezervă (patru cuvinte rămân nefolosite).

`un` `doi` `trei` `niciun` `inversă` `păstrată` `aleatoare` `inversată de două ori`

Stiva are nevoie de _____ indice, iar coada de _____ ; la stivă ordinea de ieșire este _____ față de cea de intrare, iar la coadă este _____ .

HINT *numără indicii din codul fiecărei simulări, apoi gândește-te ce iese primul din fiecare structură atunci când elementele au intrat în aceeași ordine.*

C

EXERCIȚIUL C · ASOCIERE

Ce nu poate face fiecare model?

ASOCIERE

Până acum ai comparat cele două modele prin ceea ce *pot* face. Întoarce acum întrebarea: pentru fiecare cerință, scrie în căsuță litera modelului care **nu** i se potrivește, împreună cu motivul pentru care dă greș.

☐ Vreau să servesc mereu persoana care așteaptă de cel mai mult timp.

☐ Vreau să desfac ultima operație făcută.

☐ Vreau să aflu oricând câte elemente sunt, fără niciun calcul.

☐ Vreau să refolosesc locurile rămase libere după ieșiri.

`A` Coada — ar scoate elementul cel mai vechi, deși e nevoie de cel mai nou

`B` Coada simulată cu doi indici crescători — indicii doar cresc, deci pozițiile părăsite se pierd

`C` Stiva — cine a venit primul ar fi luat ultimul

`D` Coada — răspunsul cere mereu o scădere între cei doi indici

HINT *atenție, aici se cere modelul* **nepotrivit** *. Două dintre motive se referă la ordinea de ieșire, iar celelalte două la felul în care se țin socotelile cu indicii.*

SECȚIUNEA 6 · LISTA DE FRECVENȚE

6 Lista de frecvențe: când *indexul* devine informație

Iată o problemă care pare să ceară zece variabile: *de câte ori apare fiecare cifră într-un număr?* Am avea nevoie de `nr0`, `nr1`, … `nr9` și de un `if` pentru fiecare — zece ramuri aproape identice. **Lista de frecvențe** rezolvă situația cu o idee elegantă: creăm o listă cu zece poziții, toate zero, și convenim că **poziția `c` numără aparițiile cifrei `c`**. Atunci e de ajuns o singură linie, `f[c] = f[c] + 1`, fără niciun `if`: cifra însăși **spune unde să numărăm**.

> **LISTA DE FRECVENȚE**
>
> Ideea-cheie: **indexul devine informație**. Într-o listă obișnuită, poziția e doar un loc; într-o listă de frecvențe, poziția *înseamnă* ceva — `f[7]` nu e „al optulea element", ci „de câte ori a apărut cifra 7". Zece ramuri `if` se topesc într-o singură linie.

```
n = 5883
f = [0] * 10
while n > 0:
    c = n % 10
    f[c] = f[c] + 1
    n = n // 10
```

După rulare: `f[3]` este 1, `f[5]` este 1, `f[8]` este 2, iar restul pozițiilor au rămas 0. Observă că **cifrele numărului se parcurg tot cu `while`** — regula nu s-a schimbat pentru că avem acum liste. Odată construită, lista de frecvențe **se poate parcurge ca orice altă listă**, cu `for i in range(len(f))`: la întrebări foarte diferite se răspunde cu aceeași structură, fără a mai folosi numărul inițial.

> **ATENȚIE · CAPCANA N = 0**
>
> Bucla `while n > 0` nu pornește niciodată, deci lista de frecvențe rămâne plină de zerouri, deși numărul 0 se scrie cu o cifră. Tratează cazul explicit, cu `if n == 0: f[0] = 1`, înainte de buclă.

A EXERCIȚIUL A · DRY-RUN

Cum crește lista de frecvențe

DRY-RUN

Fiecare rând este **o singură trecere prin buclă**, adică un pas. Ai dat, în prima coloană, valoarea lui `n` *înainte* ca pasul să înceapă; tu completezi celelalte trei coloane, în ordinea în care se execută liniile: cifra extrasă, valoarea pe care o are `f[c]` *după* incrementare și valoarea lui `n` *după* împărțire — ea devine tocmai `n`-ul rândului următor. Primul rând e completat ca exemplu.

```
n = 5883
f = [0] * 10
while n > 0:
    c = n % 10
    f[c] = f[c] + 1
    n = n // 10
```

n înainte de pas (dat)	c = n % 10	f[c] după incrementare	n după pas
5883	3	1	588
588			
58			
5			

HINT *cifra 8 apare de două ori, deci la al doilea 8 valoarea `f[8]` nu mai este 1. După ultimul pas `n` ajunge 0 și bucla se oprește — de aceea tabelul are exact patru rânduri.*

B

EXERCIȚIUL B · PROIECTARE

Proiectează înainte de a scrie cod

PROIECTARE

Un centru de asistență înregistrează zilnic erori de cinci tipuri, codificate de la 1 la 5. Înainte de a scrie o singură linie de cod, răspunde în scris la cele trei întrebări. Modelul se proiectează întâi și se scrie după aceea, nu invers.

1. Câte poziții are lista de frecvențe și ce anume numără fiecare poziție?

RĂSPUNSUL TĂU

2. Ce se schimbă în listă atunci când se înregistrează o eroare de tipul 3?

RĂSPUNSUL TĂU

3. Cum afli, la sfârșitul zilei, care tip de eroare a apărut cel mai des?

RĂSPUNSUL TĂU

HINT *tipurile merg de la 1 la 5, dar pozițiile listei încep de la 0 — hotărăște dinainte dacă lași poziția 0 nefolosită sau muți codurile cu unu.*

C

EXERCIȚIUL C · SCRIE CODUL

Câte cifre pare are numărul

SCRIERE COD

Se citește un număr natural `n`. Folosind o listă de frecvențe cu zece poziții, afișează câte dintre cifrele lui `n` sunt pare. Pentru `n = 0` se consideră că numărul are o singură cifră, 0. Ex.: pentru `1223` se afișează `2`.

HINT *construiește întâi lista de frecvențe, apoi parcurge-o și* **adună frecvențele** *pozițiilor pare — nu numeri pozițiile pare, ci aduni valorile de pe ele. Tratează separat cazul* `n = 0`.

SOLUȚIA TA · PYTHON

★

PERFORMANȚĂ · SCRIE CODUL

Cifra cea mai frecventă

SCRIERE COD

Se citește un număr natural `n`. Afișează pe același rând, separate printr-un spațiu, cifra care apare de cele mai multe ori în `n` și numărul ei de apariții. Dacă mai multe cifre apar de același număr maxim de ori, afișeaz-o pe cea mai mică. Pentru `n = 0` numărul are o singură cifră, 0. Ex.: pentru `2344124334` se afișează `4 4`.

HINT *parcurge lista de frecvențe de la poziția 0 în sus, păstrând maximul* **strict** *(`f[i] > mx`) — așa cifra cea mai mică la egalitate se alege de la sine. Nu folosi* `max()`*, nu a fost predată.*

SOLUȚIA TA · PYTHON

7 SECȚIUNEA 7 · COD GHIDAT

Scrie cod care *trece testele*

Cinci probleme, în ordine crescătoare de dificultate. Scrie soluția pe hârtie, urmărește-o mental pe exemplul dat și abia apoi tastează-o pe platformă, unde e verificată automat pe 5 cazuri de test. Toate folosesc doar `[0] * n`, `v[i]` și `len(v)`, plus indicii care simulează stiva și coada.

1 PROBLEMA 7.1 · COD GHIDAT

Câte cifre distincte are numărul

SCRIERE COD

Se citește un număr natural `n`. Folosind o listă de frecvențe, afișează câte cifre distincte apar în scrierea lui `n`. O cifră se numără o singură dată, indiferent de câte ori apare. Pentru `n = 0` numărul are o singură cifră, 0. Ex.: pentru `1223` se afișează `3`.

HINT *după construirea listei de frecvențe, numără* **pozițiile** *care au frecvența strict mai mare decât 0 — nu aduna frecvențele.*

SOLUȚIA TA · PYTHON

PROBLEMA 7.2 · COD GHIDAT

2 Al doilea element din vârful stivei

SCRIERE COD

Se citește un număr natural `n` mai mare sau egal cu 2, apoi `n` numere naturale, fiecare de pe câte un rând, care se pun pe o stivă în ordinea citirii. Afișează al doilea element de la vârf, adică cel care ar rămâne în vârf după o singură eliminare. Ex.: pentru `n = 3` și 5, 8, 2 se afișează `8`.

HINT *nu trebuie să elimini efectiv elementul din vârf; e de ajuns să cobori indicele `vf` de două ori și să citești de la poziția obținută.*

SOLUȚIA TA · PYTHON

3

PROBLEMA 7.3 · COD GHIDAT

Al doilea servit la ghișeu

SCRIERE COD

Se citește un număr natural `n` mai mare sau egal cu 2, apoi `n` numere naturale, fiecare de pe câte un rând — numerele de ordine ale persoanelor, în ordinea sosirii la ghișeu. Afișează numărul persoanei care va fi servită a doua. Ex.: pentru `n = 3` și 5, 8, 2 se afișează `8`.

HINT *aceeași structură ca la S3, dar `prim` se mută o dată înainte de citire.*

SOLUȚIA TA · PYTHON

4

PROBLEMA 7.4 · REPARĂ CODUL

Vârful stivei — găsește greșeala

SCRIERE COD

Programul de mai jos ar trebui să citească un număr natural `n`, cel puțin egal cu 1, apoi `n` numere naturale care se pun pe o stivă, și să afișeze elementul din vârful stivei. Programul este greșit: pentru `n = 3` și 5, 8, 2 el afișează 5 în loc de 2. Găsește greșeala și scrie programul corect.

```
1  n = int(input())
2  st = [0] * n
3  vf = 0
4  for i in range(n):
5      x = int(input())
6      st[vf] = x
7      vf = vf + 1
8  print(st[0])
```

HINT *elementele se pun corect pe stivă — acolo nu e greșeala. Uită-te la ultima linie: ce poziție citește ea și ce poziție ar trebui să citească, dacă `vf` arată spre primul loc liber?*

PROGRAMUL CORECTAT · PYTHON

PROBLEMA 7.5 · PROVOCARE

Ce iese din stivă și ce iese din coadă

SCRIERE COD

Se citește un număr natural `n`, apoi `n` numere naturale, fiecare de pe câte un rând. Aceleași elemente sunt puse, în aceeași ordine, într-o stivă și într-o coadă. Afișează pe același rând numărul `n`, un spațiu, apoi primul element scos din stivă, respectiv primul element scos din coadă. Ex.: pentru `n = 3` și 4, 9, 2 se afișează `3 2 4`.

HINT *nu e nevoie să simulezi efectiv ambele structuri: primul element scos din stivă este ultimul citit, iar primul scos din coadă este primul citit — deci `v[len(v) - 1]` și `v[0]`. Când lista are un singur element, cele două valori coincid.*

SOLUȚIA TA · PYTHON