Variante Bac 2008

1.

Se consideră o listă liniară simplu înlănțuită alocată dinamic, cu cel puțin două elemente. Fiecare element al listei reține în câmpul `urm` adresa elementului următor din listă sau `NULL` dacă nu există un element următor.

```
while (...)
  p=p->urm;
delete p->urm; | free (p->urm);
p->urm=NULL;
```

Ştiind că variabila `p` reține adresa primului element din listă, care dintre expresiile următoare poate înlocui punctele de suspensie în secvența de instrucțiuni de mai sus astfel încât, în urma executării acesteia, să fie eliminat ultimul element al listei? **(4p.)**

a. `p->urm->urm!=NULL`
b. `p->urm!=NULL`
c. `p!=NULL`
d. `p->urm->urm==NULL`

2.

O listă liniară simplu înlănțuită cu cel puțin două elemente, alocată dinamic, reține în câmpul `info` al fiecărui element câte un număr natural de maximum `4` cifre, iar în câmpul `urm` adresa elementului următor din listă sau `NULL` dacă nu există un element următor.

```
while (p->urm!=NULL)
{ if (p->urm->info<p->info)
    p->urm->info=p->info;
  p=p->urm;}
cout<<p->info; | printf("%d",p->info);
```

Dacă variabila `p` reține adresa primului element al listei atunci, în urma executării secvenței de program de mai sus se afişează întotdeauna: **(4p.)**

a. cea mai mică dintre valorile memorate de elementele din listă
b. cea mai mare dintre valorile memorate de elementele din listă
c. valoarea memorată de penultimul element din listă
d. valoarea memorată de primul element din listă

3.

Într-o listă liniară simplu înlănțuită cu cel puțin `4` elemente, fiecare element reține în câmpul `urm` adresa elementului următor sau `NULL` dacă nu există un element următor, iar în câmpul `info` o valoare întreagă. Ştiind că variabila `p` reține adresa primului element din listă, înlocuiți punctele de suspensie cu expresiile corespunzătoare, astfel încât secvența alăturată să calculeze în variabila `s` suma tuturor valorilor elementelor listei.

```
s=...;
while ( ... )
{ p=p->urm;
  s=s+p->info;
}
cout<<s; | printf("%d",s);
```

(6p.)

4.

Într-o listă liniară simplu înlănțuită fiecare element reține în câmpul `info` o valoare întreagă, iar în câmpul `urm` adresa elementului următor din listă sau `NULL` dacă nu există un element următor. Variabila `p` reține adresa primului element din listă.

Lista conține, începând de la primul element, în această ordine, valorile: 2, 3, 4, 5, 6, 7, 8. Ce se va afișa în urma executării secvenței de instrucțiuni alăturate? **(4p.)**

```
while((p->urm!=NULL)&&(p!=NULL))
{ p->urm=p->urm->urm;
  p=p->urm;
  cout<<p->info<<' '; | printf("%d ",p->info);
}
```

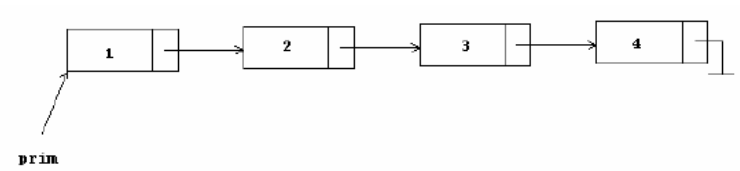
a. 2 5 8
b. 2 4 8
c. 2 4 6 8
d. 4 6 8

5.

Într-o listă liniară simplu înlănțuită cu cel puțin 3 noduri, fiecare element reține în câmpul `nr` un număr real, iar în câmpul `urm` adresa următorului element din listă sau valoarea `NULL` în cazul în care este ultimul nod al listei. Dacă `p` reține adresa primului element din listă, scrieți o **expresie** C/C++ a cărei valoare este egală cu suma valorilor reale reținute în primele trei noduri ale listei. **(6p.)**

6.

Într-o listă liniară simplu înlănțuită, fiecare element reține în câmpul `inf` un număr întreg, iar în câmpul `ref` adresa următorului nod din listă sau `NULL` în cazul ultimului nod al listei. Adresa primului element al listei este reținută în variabila `prim`, iar `p` este o variabilă de același tip cu `prim`. Dacă în listă sunt memorate, în această ordine, numerele 1, 2, 3, 4 ca în figura de mai jos, care va fi conținutul listei în urma executării secvenței alăturate de instrucțiuni? **(4p.)**




```
p=prim->ref->ref;
prim->ref->ref=p->ref;
p->ref=prim->ref;
prim->ref=p;
```

a. 1 3 2 4
b. 1 2 4 3
c. 1 4 2 3
d. 1 4 3 2

O listă liniară simplu înlănțuită, alocată dinamic, memorează în câmpul `ref` al fiecărui nod adresa următorului nod din listă sau `NULL` în cazul în care nu există un nod următor. Lista conține cel puțin 5 noduri, adresa primului nod este memorată în variabila `p`, iar variabilele `a` și `b` sunt de același tip cu `p`. Adresa cărui nod va fi memorată în variabila `b`, după executarea secvenței alăturate de program? **(6p.)**

```
a=p;
while (a->ref != NULL)
{ b=a
  a=a->ref;
}
```

a. Nodul aflat în mijlocul listei
b. Penultimul nod al listei
c. Ultimul nod al listei
d. Nodul al treilea din listă

8. 8.

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul **inf** un număr întreg, iar în câmpul **ref** adresa următorului nod din listă sau **NULL** în cazul ultimului element al listei.

Adresa primului element al listei este reținută în variabila **prim**, iar variabila **p** este de acelaşi tip cu **prim**. Dacă în listă sunt memorate, în această ordine, numerele **3**, **5**, **18**, **20**, ce se va afişa pe ecran în urma executării secvenței alăturate de program? **(6p.)**

```
p=prim;
while (p->ref!=NULL)
{ if (p->inf%5==0)
    s=s+p->inf;
  p=p->ref;
}
cout<<s;   |   printf("%d",s);
```

a. 21 **b.** 25 **c.** 5 **d.** 46

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul **ref** adresa următorului nod din listă sau **NULL** în cazul ultimului element al listei, iar în câmpul **inf** un număr întreg.

Adresa primului element al listei este reținută în variabila **prim**, iar **p** este o variabilă de acelaşi tip cu **prim**. Ce va afişa pe ecran secvența alăturată? **(6p.)**

```
p=prim;
while ((p->inf%2==0) && (p!=NULL))
  p=p->ref;
if (p!=NULL)
  cout<<(p->inf);   |   printf("%d",p->inf);
else
   cout<<"NU";   |   printf("NU");
```

a. Prima valoare impară din listă, dacă aceasta există şi **NU** în caz contrar.

b. Prima valoare pară din listă, dacă aceasta există şi **NU** în caz contrar.

c. Toate valorile impare din listă dacă astfel de valori există şi **NU** în caz contrar.

d. Toate valorile pare din listă dacă astfel de valori există şi **NU** în caz contrar.

10.

Într-o listă liniară simplu înlanțuită, alocată dinamic, fiecare element conține în câmpul **nr** un număr real, iar în câmpul **urm** adresa elementului următor. Lista are cel puțin două elemente, iar variabila **x** memorează adresa primului element din listă. Cu ce pot fi completate punctele de suspensie din secvența următoare, astfel încât să afişeze cuvântul **ADEVARAT** dacă media aritmetică dintre valorile câmpului **nr** ale primelor două elemente din listă este mai mică sau cel puțin egală cu **4.75**, respectiv cuvântul **FALS** în caz contrar?

```
if (.....)cout << "ADEVARAT";   |   printf("ADEVARAT");
else cout << "FALS";   |   printf("FALS");
```

(6p.)

11.

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element memorează în câmpul **nr** un număr întreg, iar în câmpul **urm** adresa elementului următor din listă sau valoarea **NULL** dacă nu există un element următor. Lista conține exact trei elemente ale căror adrese sunt memorate în variabilele **p**, **q** şi **r**. Ştiind că **p->nr==1**, **q->nr==2**, **r->nr==3**, **p->urm!=NULL** şi **r->urm==q**, care este ordinea numerelor din listă? **(4p.)**

a. 1 3 2 **b.** 1 2 3 **c.** 2 1 3 **d.** 3 2 1

12.

Fiecare element al unei liste circulare, nevide, alocată dinamic, memorează în câmpul `val` o valoare întreagă, iar în câmpul `adr` adresa elementului următor. Ştiind că variabila `p` reține adresa unui element oarecare din listă, iar variabila `q` este de acelaşi tip cu `p`, precizați care dintre următoarele variante tipăreşte toate elementele listei? **(4p.)**

a.
```
q=p;
while(q!=p) {cout<<q->val;
             q=q->adr;}
```
```
q=p;
while(q!=p) {printf("%d",q->val);
             q=q->adr;}
```

b.
```
q=p;
while(q->adr!=p){cout<<q->val;
                 q=q->adr;}
```
```
q=p;
while(q->adr!=p){printf("%d",q->val);
                 q=q->adr;}
```

c.
```
q=p;
do{ cout<<q->val;
    q=q->adr;
  }while(q!=p);
```
```
q=p;
do{ printf("%d",q->val);
    q=q->adr;
  }while(q!=p);
```

d.
```
q=p->adr;
while(q!=p) {cout<<q->val;
             q=q->adr;}
```
```
q=p->adr;
while(q!=p) {printf("%d",q->val);
             q=q->adr;}
```

Fiecare element al unei liste liniare, simplu înlănțuite, alocată dinamic, reține în câmpul `nr` un număr întreg, iar în câmpul `adr` adresa elementului următor din listă. Dacă `p` reține adresa primului element, iar lista are cel puțin două elemente, care dintre următoarele secvențe de instrucțiuni copiază în câmpul `nr` al celui de-al doilea element al listei, conținutul câmpului `nr` al primului element din listă? **(4p.)**

a. `p->nr=p->adr->nr;`

b. `p->adr=p->nr;`

c. `p->adr->nr=p->nr;`

d. `p->adr->adr->nr=p->nr;`

Într-o listă circulară, nevidă, alocată dinamic, cu exact `9` elemente, fiecare element memorează în câmpul `val` o valoare întreagă, iar în câmpul `adr` adresa elementului următor.

Ştiind că în listă sunt memorate, în ordine, numerele de la `1` la `9`, şi variabila `p` reține adresa elementului cu valoarea `4`, iar variabila `q` este de acelaşi tip cu `p`, precizați ce va afişa secvența alăturată? **(6p.)**

```
q=p;
cout<<q->adr->val<<endl;
      | printf("%d",q->adr->val);
while(q->adr!=p)
   q=q->adr;
cout<<q->adr->val;
       | printf("%d",q->adr->val);
```

O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul `nr` al fiecărui element câte un număr natural, iar în câmpul `urm`, adresa elementului următor din listă sau `NULL` dacă nu există un astfel de element. Lista memorează, în ordine, numai elementele `1`, `2`, `3`, `4`, `5`.

Ştiind că variabila `prim` memorează adresa primului element al listei şi că `p` este o variabilă de acelaşi tip cu `prim`, care va fi valoarea reținută în ultimul element după executarea secvenței de mai sus? **(4p.)**

```
p=prim;
while(p->urm!=NULL)
{p->urm->nr=p->nr*p->urm->nr;
 p=p->urm;
}
```

O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul **nr** al fiecărui element câte un număr natural, iar în câmpul **urm**, adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Lista memorează, în ordine doar elementele **1**, **2**, **3**, **4**, **5**.
Știind că variabila **prim** memorează adresa primului element al listei şi că **p** este o variabilă de acelaşi tip cu **prim**, care sunt, în ordine, elementele listei după executarea secvenței alăturate? **(6p.)**

```
p=prim; x=p->nr;
while(p->urm!=NULL)
 {p->nr = p->urm->nr;
  p=p->urm;
 }
p->nr=x;
```

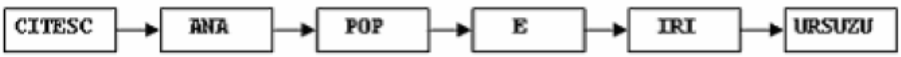
O listă liniară dublu înlănțuită, alocată dinamic, reține în câmpul **nr** al fiecărui element câte un număr natural, în câmpul **urm**, adresa elementului următor din listă, iar în câmpul **prec**, adresa elementului precedent din listă.
Variabilele **p** şi **q** memorează adresa primului, respectiv ultimului element al listei.
Care este numărul maxim de elemente pe care le poate avea lista astfel încât, după executarea secvenței alăturate, valoarea variabilei **n** să fie **3**? **(6p.)**

```
n=0;
 while(p!=q && q->urm!=p)
 { p=p->urm;
   q=q->prec;
   n=n+1;
 }
```

O listă liniară simplu înlănțuită, cu cel puțin **5** elemente, alocată dinamic, reține în câmpul **nr** al fiecărui element câte un număr natural, iar în câmpul **urm**, adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **prim** memorează adresa elementului aflat pe prima poziție în listă, **ultim** adresa elementului aflat pe ultima poziție în listă, iar **p şi q** sunt două variabile de acelaşi tip cu **prim**. Pe ce poziție se va găsi în lista modificată în urma executării secvenței alăturate, elementul aflat pe poziția a doua, în lista inițială? **(4p.)**

```
p=prim;
prim=ultim;
while(p!=prim)
{q=p;p=p->urm;
 ultim->urm=q;
 ultim=q;
}
ultim->urm=NULL;
```

Se consideră o listă liniară simplu înlănțuită alocată dinamic în care fiecare nod memorează în câmpul **info** un cuvânt, iar în câmpul **urm**, adresa următorului nod al listei.
Scrieți un program **C/C++** care citeşte de la tastatură un număr natural **n** (**n≤100**) şi apoi **n** cuvinte distincte, fiecare cuvânt având maximum **20** de litere, toate majuscule, şi construieşte o listă simplu înlănțuită, cu acele cuvinte citite, care încep şi se termină cu aceeaşi literă. Cuvintele se vor memora în listă în ordinea inversă a citirii lor.
Exemplu: pentru **n=9** şi cuvintele citite:
URSUZU IRI E SUPARAT POP DORIS SI ANA CITESC REFAC DESEN lista va fi



(10p.)

O listă dublu înlănțuită, alocată dinamic, memorează în câmpul **info** al fiecărui nod un număr real, iar în câmpurile **urm** şi **prec** adresa nodului următor, respectiv precedent din listă. Adresa primului nod este memorată în variabila **prim**.
Considerându-se creată lista dublu înlănțuită, scrieți declarările de date necesare definirii listei precum şi secvența de program **C/C++** care modifică lista, inserând după fiecare valoare negativă din listă, un nou nod, în care se va memora valoarea **0**, şi afişează toate informațiile din nodurile listei după efectuarea acestei operații. **(10p.)**

Într-o listă simplu înlănțuită alocată dinamic sunt memorate în ordine, următoarele valori:

2 ->3 ->4 ->7 ->5 ->9 ->14

Dacă `p` este adresa primului element al listei şi fiecare element reține în câmpul `urm` adresa elementului următor, care este informația din elementul a cărui adresă o va reține `p` în urma executării secvenței alăturate: **(6p.)**

```
p=p->urm;
while(p->urm->urm!=0)
   p=p->urm->urm;
```

Se consideră o listă alocată dinamic care are cel puțin `10` elemente şi fiecare element al listei memorează în câmpul info un număr întreg, iar în câmpul `next` adresa elementului următor în listă sau `NULL` dacă nu există un element următor.

Variabila `p` memorează adresa de început a listei, iar variabila `aux` este de acelaşi tip cu `p`. Dacă în urma executării secvenței alăturate de program, variabila `p` are valoarea `NULL`, atunci: **(4p.)**

```
while (p!=NULL && p->info%5!=0)
{ aux=p;
  p=aux->next;
  delete p;   |  free(p);
}
```

a. toate numerele din listă sunt divizibile cu 5

b. doar primul element din listă este divizibil cu 5

c. în listă nu s-a memorat niciun număr divizibil cu 5

d. doar ultimul element memorat în listă este divizibil cu 5

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul `adr` adresa următorului element din listă, iar în câmpul `info` un număr întreg. Adresa primului element al listei este memorată în variabila `p`. Ştiind că lista conține exact `4` elemente, atunci expresia `p^.adr^.info` reprezintă: **(4p.)**

a. adresa celui de al doilea element

b. adresa celui de al treilea element

c. valoarea memorată în al doilea element

d. valoarea memorată în al treilea element

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul `adr` adresa următorului nod din listă, iar în câmpul `info` un număr întreg. Variabilele `d` şi `q` rețin adresele câte unui nod din listă. Să se identifice secvența de instrucțiuni care realizează inserarea corectă, în listă, a nodului memorat la adresa `q`, ca succesor al nodului reținut la adresa `d`. **(4p.)**

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul `adr` adresa următorului nod din listă, iar în câmpul `info` un număr întreg. Considerăm că o astfel de listă memorează, în ordine, doar valorile `7`, `5`, `4`, `9`, `3`. Variabila `d` reține adresa nodului la care este memorată valoarea `4`. Care este secvența de instrucțiuni care trebuie executată pentru ca lista să conțină, în ordine, doar valorile `7`, `5`, `9`, `3`? **(6p.)**

a. `d->adr=d->adr; d->info=d->adr->info;`

b. `d->adr=d->adr->adr;`

c. `d->info=d->adr->info; d->adr=d->adr->adr;`

d. `d->adr->adr=d->adr; d->adr->info=d->info;`

Într-o listă liniară simplu înlănțuită, alocată dinamic, fiecare element reține în câmpul `info` un număr întreg, iar în câmpul `leg` adresa următorului nod din listă sau `NULL` dacă nu există un nod următor. Adresa primului element al listei este memorată în variabila `p`. Ce valoare se va afişa, în urma executării secvenței alăturate, dacă lista memorează, în ordine, doar valorile `5, 4, 3, 2, 6` ? **(6p.)**

```
x=1;
while (p->leg!=NULL)
 {
  x=x * p->leg->info;
  p=p->leg;
 }
cout<<x;|printf("%d",x);
```

Fiecare nod al unei liste simplu înlănțuite, cu cel puțin `4` noduri, reține in câmpul `urm` adresa nodului următor din listă sau `NULL` dacă nu are un nod următor. Ştiind că variabila `p` reține adresa primului nod din listă, variabila `q` reține adresa celui de-al doilea nod din listă, iar variabila `r` reține adresa celui de-al treilea nod din listă, care este secvența prin care se interschimbă al doilea cu al treilea element din lista inițială? **(4p.)**

a.
```
p->urm=r;
q->urm=r->urm;
r->urm=q;
```

b.
```
p->urm=r;
r->urm=q->urm;
q->urm=r->urm;
```

c.
```
r->urm=q->urm;
q->urm=r->urm;
p->urm=r;
```

d.
```
q->urm=r->urm;
p->urm=r;
r->urm=q->urm;
```

Într-o listă simplu înlănțuită alocată dinamic, cu cel puțin două noduri, fiecare nod reține în câmpul `urm` adresa nodului următor din listă sau `NULL` dacă nu are un nod următor. Ştiind că variabila `p` reține adresa primului nod din listă, iar variabila `q` este de acelaşi tip cu `p`, care este secvența ce realizează eliminarea celui de-al doilea nod din listă? **(4p.)**

a.
```
q=p->urm;
p->urm=p->urm->urm;
delete q;  |  free(q);
```

b.
```
p->urm=p->urm->urm;
delete p;  |  free(p);
```

c.
```
q=p->urm;
q->urm=p->urm->urm;
delete q;  |  free(q);
```

d.
```
q=p->urm;
q->urm=p->urm->urm;
delete q;  |  free(q);
```

Într-o listă simplu înlănțuită, fiecare element reține în câmpul `urm` adresa elementului următor din listă sau `NULL` dacă nu are un element următor. Lista are cel puțin două elemente, variabila `p` reține adresa primului element, iar `q` reține adresa ultimului element din listă. Care este numărul de elemente din listă dacă este relația de mai jos are valoarea `1`?

```
p->urm->urm==q
```

(4p.)

a. 3 **b.** 2 **c.** 4 **d.** 5

Într-o listă simplu înlănțuită cu cel puțin `4` elemente, fiecare nod reține in câmpul `urm` adresa nodului următor din listă sau `NULL` dacă nu are un nod următor. Ştiind că inițial variabila `p` reține adresa primului nod din listă, după executarea cărei secvențe `p` va reține adresa ultimului nod din listă? **(4p.)**

a. `while(p->urm!=NULL) p=p->urm;`

b. `while(p!=NULL) p=p->urm;`

c. `p=p->urm;`

d. `p=p->p->urm;`

Într-o listă simplu înlănțuită nevidă, fiecare element reține în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu are un nod următor. Știind că variabila **p** reține adresa primului element din listă, variabila **q** reține adresa ultimului element din listă, iar lista este formată din exact 3 elemente, care dintre următoarele secvențe de instrucțiuni determină, în urma executării, eliminarea celui de-al doilea element din listă? **(4p.)**

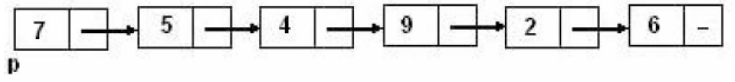
a.
```
delete p->urm;
p->urm=q;
```
|
```
free(p->urm);
```

b.
```
p->urm=q;
delete p->urm;
```
|
```
free(p->urm);
```

c.
```
p=q;
delete p->urm;
```
|
```
free(p->urm);
```

d.
```
p->urm=q;
delete q;
```
|
```
free(p->urm);
```

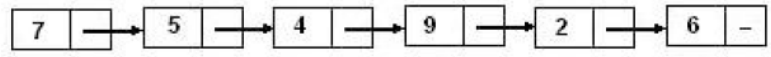
Într-o listă simplu înlănțuită cu cel puțin 3 elemente, fiecare element reține în câmpul **inf** un număr natural, iar în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **p** reține adresa primului element din listă. Dacă se prelucrează lista de mai jos, care este valoarea memorată de variabila întreagă **k**, la finalul executării următoarei secvențe de instrucțiuni?




```
k=0;
while(p->urm->urm && p->inf > p->urm->inf)
 { p = p->urm; k = k + p->urm->inf; }
```
(6p.)

Într-o listă simplu înlănțuită cu cel puțin 2 elemente, fiecare element reține în câmpul **inf** un număr natural, iar în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **p** reține adresa primului element din listă. Dacă se prelucrează lista de mai jos, care este valoarea memorată de variabila întreagă **k**, la finalul executării următoarei secvențe de instrucțiuni?

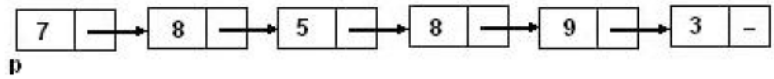
```
k=0;
while((p->urm!=NULL) && (p->inf*p->urm->inf%10!=0))
 { p = p->urm; k ++; }
```



(6p.)

Într-o listă simplu înlănțuită cu cel puțin 3 elemente, fiecare element reține în câmpul **inf** un număr natural, iar în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **p** reține adresa primului element din listă, iar variabilele **q** şi **aux** sunt de acelaşi tip cu **p**. Dacă se prelucrează lista de mai jos, care va fi conținutul listei după executarea următoarei secvențe de instrucțiuni?

```
q=p;
while(q->urm->urm !=NULL && q->inf >= p->inf)   q = q->urm;
aux=q->urm;
q->urm=aux->urm;
delete aux;
```

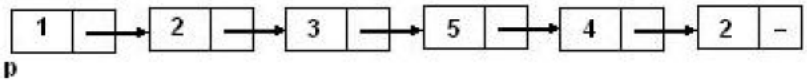


(4p.)

a. 8 5 8 9 3 b. 7 8 8 9 3 c. 7 8 5 8 9 3 d. 7 8 5 9 3

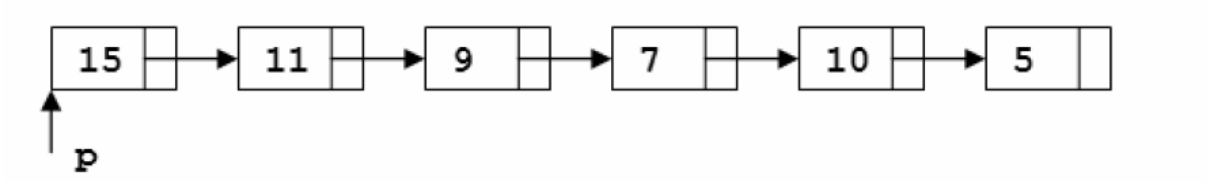
Într-o listă simplu înlănțuită cu cel puțin 2 elemente, fiecare element reține în câmpul **inf** un număr natural, iar în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **p** reține adresa primului element din listă, iar variabila **q** este de acelaşi tip cu **p**. Dacă se prelucrează lista de mai jos, care va fi conținutul listei după executarea următoarei secvențe de instrucțiuni?

```
q=p;
while(q->urm!=NULL && q->inf<=q->urm->inf) q=q->urm;
q->inf=q->urm->inf+1;
```



(6p.)

Într-o listă simplu înlănțuită cu cel puțin 2 elemente, fiecare element reține în câmpul **inf** un număr natural, iar în câmpul **urm** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Variabila **p** reține adresa primului element din listă. Dacă se prelucrează lista de mai jos, care este valoarea memorată de variabila întreagă **k**, la finalul executării următoarei secvențe de instrucțiuni?




```
k=3;
while(p->urm!=NULL && p->inf > p->urm->inf)   p = p->urm;
k = k + p->urm->inf;
```

(4p.)

a. 8 **b.** 10 **c.** 12 **d.** 13

Se consideră o listă liniară simplu înlănțuită cu cel puțin 5 noduri, în care fiecare nod al listei conține în câmpul **urm** adresa nodului următor din listă. Adresa primului nod este memorată în variabila **prim**, iar variabila **p** este de acelaşi tip cu **prim**.

Ce modificare se produce asupra listei prin executarea secvenței de instrucțiuni alăturate? **(6p.)**

```
p=prim;
prim=p->urm;
delete(p);   |   free(p);
```

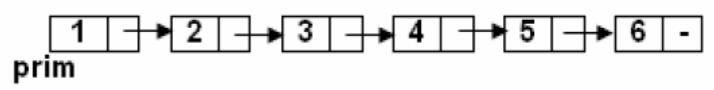
Se consideră o listă simplu înlănțuită alocată dinamic, în care fiecare nod memorează în câmpul **info** un număr întreg, iar în câmpul **urm** adresa elementului următor.

Scrieți un program C/C++ care citeşte de la tastatură 3 numere naturale nenule **n**, **a** şi **r** (**n≤20**, **a≤10**, **r≤10**) şi construieşte în memorie o listă simplu înlănțuită astfel încât parcurgând lista de la primul nod până la ultimul nod şi afişând pe ecran, separate prin câte un spațiu, numerele memorate în câmpul **info** al fiecărui nod, se obțin în ordine strict crescătoare toate elementele mulțimii **M={a, a+r, a+2·r,...,a+(n-1)·r}**.

Exemplu: dacă **n=4**, **a=10**, **r=2** atunci se vor afişa elementele alăturate. **(10p.)**

```
10 12 14 16
```

Se consideră lista liniară simplu înlănțuită cu 6 noduri, reprezentată mai jos, în care fiecare nod conține în câmpul **info** un număr natural, iar în câmpul **urm** adresa nodului următor din listă sau **NULL** dacă nu există un nod următor.



Dacă adresa primului nod este memorată în variabila `prim`, iar variabila `p` este de acelaşi tip cu `prim`, ce se afişează la executarea secvenței de mai jos?

```
for(p=prim->urm;p->urm!=NULL;p=p->urm->urm)
  cout<<p->info<<"  "; | printf("%d ",p->info);
```

(6p.)

Se consideră o listă liniară simplu înlănțuită cu cel puțin 5 noduri, în care fiecare nod al listei conține în câmpul `urm` adresa nodului următor din listă.

Dacă adresa primului nod este memorată în variabila `prim`, iar variabila `p` este de acelaşi tip cu `prim`, ce prelucrare realizează următoarea secvență de instrucțiuni? **(6p.)**

```
p=prim->urm;
prim->urm=p->urm;
delete(p); | free(p);
```

Într-o listă simplu înlănțuită circulară, alocată dinamic, fiecare element reține în câmpul `adr` adresa elementului următor din listă. Dacă `p` şi `q` sunt adresele a două elemente distincte din listă astfel încât să fie îndeplinite condițiile `p= q->adr` şi `q = p->adr` , atunci lista are: **(4p.)**

a. un numar impar de elemente
b. exact 2 elemente
c. cel putin 3 elemente
d. exact 1 element

Într-o listă simplu înlănțuită cu cel puțin `1000` de elemente identificate prin adrese, fiecare element reține în câmpul `adr` adresa elementului următor din listă. Dacă `q` este adresa unui element din listă şi `p` o variabilă de acelaşi tip cu `q`, ce reține adresa unui alt element, care nu face parte din listă, atunci inserarea elementului de la adresa `p`, în listă, imediat după elementul de la adresa `q` se realizează cu ajutorul secvenței de instrucțiuni: **(4p.)**

a. `p->adr=q->adr; q->adr=p;`
b. `p=q; q->adr= p->adr;`
c. `q->adr=p; p->adr=q;`
d. `q=p->adr; p->adr= q->adr;`

Într-o listă liniară dublu înlănțuită, alocată dinamic, fiecare element reține în câmpul `dr` adresa următorului nod din listă, în câmpul `st` adresa nodului precedent din listă, iar în câmpul `info` un număr întreg. Adresa primului element al listei este reținută în variabila `p`. Dacă în listă sunt memorate, începând cu elementul de la adresa `p`, toate numerele naturale de la `10000` la `1`, în ordine descrescătoare, care va fi numărul memorat în câmpul `info` al celui de-al `4`-lea element din listă după executarea secvenței alăturate? **(4p.)**

```
new(r);
r->info=0;
q= p->dr->dr->dr;
q->st=r;
r->dr=q;
r->st= p->dr->dr;
p->dr->dr->dr=r;
```

a. 9998
b. 9999
c. 9997
d. 0

Într-o listă simplu înlănțuită cu cel puțin 2 elemente, fiecare element memorează în câmpul `info` un număr întreg, iar în câmpul `urm` adresa elementului următor din listă sau `NULL` dacă nu există un element următor.

```
p=primul;
while (p->urm!=NULL) p=p->urm;
printf("%d",p->info); | cout<<p->info;
```

Ştiind că `primul` reprezintă adresa primului element din listă, iar variabila `p` este de acelaşi tip cu `primul`, ce realizează secvența alăturată de program? **(6p.)**

Se consideră o listă liniară simplu înlănțuită ale cărei noduri reţin în câmpul `data` o valoare numerică întreagă. Ştiind că `x, y` şi `z` reţin adresele unor elemente din listă, ce se va afişa după executarea secvenței alăturate de program? **(6p.)**

```
x->data=2;
z=x;
y->data=5;
y=z;
cout<<x->data<<y->data<<z->data;  |
printf("%d%d%d",x->data,y->data,z->data);
```

Fie o listă liniară simplu înlănțuită ale cărei noduri reţin in câmpul `next` adresa nodului următor sau `NULL` dacă nu există un element următor în listă. Lista are cel puțin două elemente. Ştiind că variabila `x` reține adresa primului nod din listă, iar variabila `y` reține adresa celui de al doilea nod din listă, scrieți o secvență de instrucțiuni în limbajul `C/C++`, care inserează între cele două noduri `x` şi `y` un nod a cărui adresă este memorată de variabila `z` ce are acelaşi tip cu `x` şi `y`. **(6p.)**

Se consideră o listă liniară simplu înlănțuită ale cărei noduri reţin în câmpul `next` adresa nodului următor sau `NULL` dacă nu există un element următor în listă. Lista are cel puțin un element. Ştiind că variabila `u` reține adresa ultimului nod din listă, care este secvența de instrucțiuni în limbajul `C/C++` prin care se inserează în listă după nodul `u` un nou nod a cărui adresă este reținută de variabila `v`, de acelaşi tip cu `u`? **(6p.)**

Se consideră o listă liniară simplu înlănțuită ale cărei noduri reţin în câmpul `next` adresa nodului următor sau `NULL` dacă nu există un element următor. Lista are cel puțin două elemente. Ştiind că `p1` reține adresa primului nod din listă iar `u1` adresa ultimului nod, care este atribuirea corectă, în limbajul `C/C++`, prin care lista liniară se transformă într-o listă circulară? **(6p.)**

Într-o listă liniară simplu înlănțuită cu cel puțin 3 noduri, fiecare element reține în câmpul `urm`, adresa următorului element din listă. Dacă în variabila `p` se reține adresa primului element din listă şi `q` este o variabilă de acelaşi tip cu `p`, atunci care dintre secvențele de mai jos elimină din listă al doilea nod ? **(4p.)**

a.
```
q=p->urm;
p->urm=q->urm;
delete q;  |  free(q);
```

b.
```
q=p->urm;
delete q;  |  free(q);
p->urm=q->urm;
```

c.
```
delete p->urm;  |  free(p->urm);
p->urm=p->urm->urm;
```

d.
```
q=p->urm->urm;
p->urm=q->urm;
delete q;  |  free(q);
```

Se consideră lista simplu înlănțuită în care fiecare nod memorează în câmpul **info** un număr întreg, iar în câmpul **urm** adresa nodului următor. În listă sunt memorate, în această ordine, numai valorile 1, 2, 3. Dacă variabila **p** reține adresa primului nod din listă, iar variabila **u** adresa ultimului nod din listă, scrieți instrucțiunile care pot înlocui zona punctată din secvența alăturată, astfel încât, în urma executării acesteia, să se afişeze 3 2 1. **(6p.)**

```
.....
.....
while(u != NULL)
{
   cout << u->info<<" ";
   u = u->urm;
}
```

Într-o listă liniară simplu înlănțuită, cu cel puțin 3 noduri, fiecare element reține în câmpul **urm** adresa următorului element din listă, iar în câmpul **info** informația utilă de tip întreg. Dacă variabila **p** reține adresa primului element din listă atunci care dintre secvențele de mai jos atribuie câmpului **info** al celui de al treilea nod informația utilă din primul nod al listei? **(4p.)**

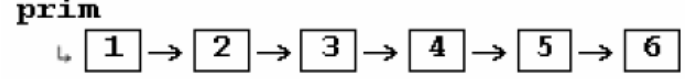
a. `p->urm->urm->info=p->info;`
b. `p->urm->urm->info=p->urm->info;`
c. `p->info->info->info = p->info;`
d. `p->urm->urm = p->info;`

Într-o listă liniară simplu înlănțuită cu cel puțin 3 noduri, fiecare element reține în câmpul **urm** adresa următorului element din listă. Dacă în variabila **prim** se reține adresa primului element din listă, iar **p, q** şi **r** sunt variabile de acelaşi tip cu **prim**, scrieți instrucțiunile cu care trebuie înlocuite punctele de suspensie din secvența de program alăturată, astfel încât, în urma executării ei, să se inverseze sensul legăturilor în listă. **(6p.)**

```
p = prim; q = prim->urm;
p->urm = NULL;
while(q != NULL)
{   r = q -> urm;
    .....
    .....
    q= r;
}
```

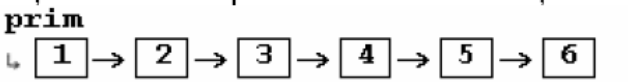
Într-o listă simplu înlănțuită fiecare element reține în câmpul **urm** adresa elementului următor din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este memorată în variabila **prim**, variabila **p** este de acelaşi tip cu **prim**, iar variabila **x** este de tip întreg. Inițial, în listă sunt memorate, în această ordine, numerele de mai jos,. Care este conținutul listei în urma executării secvenței de instrucțiuni scrise alăturat? **(4p.)**

```
p=prim;
while(p->urm!=NULL)
 {x=p->inf;
  p->inf=p->urm->inf;
  p->urm->inf=x;
  p=p->urm;
 }
```



a. 2 3 4 5 6 1 **b.** 6 5 4 3 2 1 **c.** 2 1 4 3 6 5 **d.** 1 2 3 4 5 6

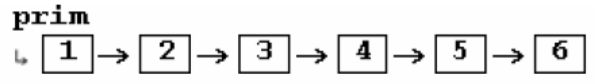
Într-o listă simplu înlănțuită fiecare element reține în câmpul **urm** adresa elementului următor din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este memorată în variabila **prim**, iar variabilele **p** şi **q** sunt de acelaşi tip cu **prim**. Inițial, în listă sunt memorate, în această ordine, numerele de mai jos. Care va fi conținutul listei după executarea secvenței alăturate? **(4p.)**




```
p=prim->urm;
q=p->urm;
p->urm=q->urm;
prim->urm=q;
q->urm=p;
```

a. 1 3 2 4 5 6 **b.** 6 5 4 3 2 1 **c.** 1 2 4 3 5 6 **d.** 2 1 3 4 5 6

Într-o listă simplu înlănțuită fiecare element reține în câmpul **urm** adresa elementului următor din listă,. Adresa primului element al listei este memorată în variabila **prim**, iar variabilele **p** şi **q** sunt de acelaşi tip cu **prim**. Care va fi conținutul listei de mai jos după executarea secvenței alăturate?



(4p.)

```
p=prim;
while(p->urm->urm->urm!=NULL)
p=p->urm;
q=p->urm;
p->urm=p->urm->urm;
delete q;   |   free(q);
```

a. 1 2 3 4 5 **b.** 1 2 3 4 6 **c.** 1 3 4 5 6 **d.** 2 3 4 5 6

Se consideră lista simplu înlănțuită în care fiecare nod memorează în câmpul **nr** o valoare întreagă şi în câmpul **urm** adresa nodului următor sau **NULL** dacă este ultimul nod din listă.
În listă sunt memorate, în această ordine, valorile **4, 3, 2, 5, 7, 9, 6, 1, 8**. Variabila **prim** reține adresa primului element din listă, variabila **p** este de acelaşi tip cu **prim**., iar variabila **k** este de tip întreg. Care este valoarea ce se va afişa în urma executării secvenței alăturate ? **(4p.)**

```
p=prim;
k=0;
while(p->urm!=NULL && k>=0)
{if(p->nr%2==0) k=k+1;
 else k=k-1;
 p=p->urm;
}
printf("%d",p->nr);  |  cout<<p->nr;
```

a. 9 **b.** 4 **c.** 3 **d.** 8

Se consideră o listă simplu înlănțuită în care fiecare nod memorează în câmpul **nr** o valoare întreagă şi în câmpul **urm** adresa nodului următor sau **NULL** dacă este ultimul nod din listă.
În listă sunt memorate, în această ordine, valorile **-1, -2, -4, -3, -5, -7, -6**.
Variabila **prim** reține adresa primului element din listă, variabila **p** este de acelaşi tip cu variabila **prim**, iar variabila **s** este de tip întreg. Care este valoarea ce se va afişa în urma executării secvenței alăturate? **(4p.)**

```
p=prim;
s=0;
while(p!=NULL && s>p->nr)
{if(p->nr%2!=0)
        s=s+p->nr;
 p=p->urm;
}
printf("%d",s);  |   cout<<s;
```

a. 0 **b.** -7 **c.** -16 **d.** -9

O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul **info** al fiecărui element câte un număr natural nenul cu cel mult **4** cifre, iar în câmpul **adr** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Considerând că adresa primului element al listei este reținută de variabila **prim**, şi că variabila **p** este de acelaşi tip cu variabila **prim**, să se completeze secvența C/C++ următoare, astfel încât ea să determine afişarea tuturor numerelor memorate în listă, care sunt divizibile cu 7.

```
p=prim;
while(p!=NULL)
    {. . .}
```

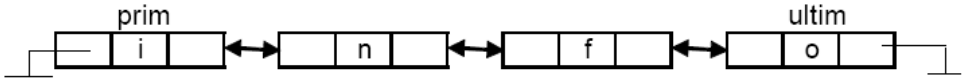
(6p.)

O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul **info** al fiecărui element câte un număr natural nenul cu cel mult **4** cifre, iar în câmpul **adr** adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Considerând că adresa primului element al listei este reținută de variabila **prim**, şi că variabila **p** este de acelaşi tip cu variabila **prim**, să se completeze secvența C/C++ următoare, astfel încât ea să determine afişarea numerelor memorate în listă, care au cifra unităților egală cu 0.

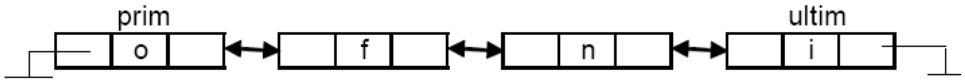
```
p=prim;
while(p!=NULL)
    {. . .}
```

(6p.)

O listă liniară dublu înlănțuită, alocată dinamic, reține în câmpul **info** al fiecărui element câte o literă din alfabetul englez. Considerând că lista este creată şi conține un număr par de elemente şi că adresa primului element este reținută în variabila **prim**, iar adresa ultimului element este reținută în variabila **ultim** să se scrie declarările de tipuri şi date necesare şi secvența de program C/C++ care inversează ordinea valorilor reținute în listă.
Exemplu: dacă lista conține inițial valorile



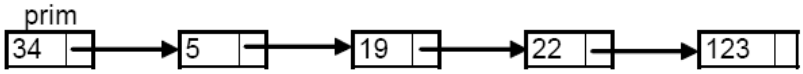
se va afişa următoarea listă:



(10p.)

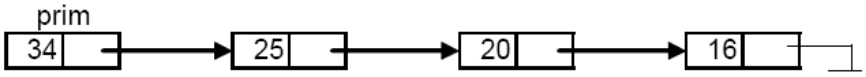
O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul **info** al fiecărui element câte un număr natural din intervalul **[1, 10000]**, iar în câmpul **adr**, adresa elementului următor din listă. Considerând că lista este creată şi că adresa primului element este reținută de variabila **prim** să se scrie declarările de tipuri şi date necesare şi secvența C/C++ care afişează pe ecran produsul numerelor memorate în primul şi ultimul element al listei.

Exemplu: pentru lista



se va afişa numărul **4182**. **(10p.)**

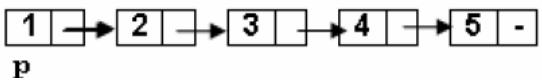
O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul **info** al fiecărui element câte un număr natural din intervalul **[1,10000]**, iar în câmpul **adr**, adresa elementului următor din listă. Considerand lista creată şi că adresa primului element este reținută în variabila **prim**, să se scrie declarările de tipuri şi date necesare şi secvența de program **C/C++** care afişează pe ecran numerele memorate în listă, care sunt pătrate perfecte.
Exemplu: pentru lista



se vor afişa numerele **25 şi 16**. **(10p.)**

Fiecare element al unei liste înlănțuite reține în câmpul **nr** un număr întreg, iar în câmpul **urm** adresa următorului element din listă sau **NULL** dacă nu există un element următor. Ce valori au variabilele întregi **a** şi **b** după executarea secvenței alăturate, dacă variabila **p** reține adresa primului element al listei de mai jos, iar variabila **q** este de acelaşi tip cu **p**? **(6p.)**

```
q=p;
a=p->urm->nr;
while (q->urm->urm!=NULL)
{
 q=q->urm;
 q->urm->nr=q->nr+q->urm->nr;
}
b=q->nr;
```



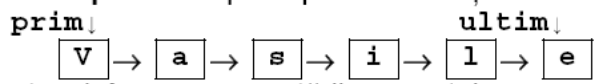
O listă dublu înlănțuită memorează în fiecare nod al său, în câmpul **info** un caracter, iar în câmpurile **prec** şi **urm** adresa nodului precedent, respectiv următor din listă.
Scrieți programul **C/C++** care citeşte de la tastatură un cuvânt (având maximum **20** de litere), creează lista dublu înlănțuită care va conține în ordine, de la stânga la dreapta, caracterele cuvântului citit (câte o literă în fiecare nod al listei) şi afişează pe ecran caracterele din listă, în ordinea inversă a memorării lor.
Exemplu: dacă s-a citit de la tastatură cuvântul **IRINA** se crează lista de mai jos, apoi se va afişa **ANIRI**.

(10p.)

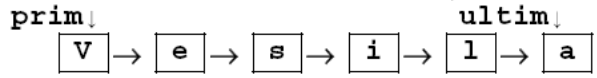
O listă liniară simplu înlănțuită cu **99** de elemente, alocate dinamic, reține în câmpul **nr** al fiecărui element câte un număr natural, iar în câmpul **urm**, adresa elementului următor din listă sau **NULL** dacă nu există un element următor. Ştiind că **prim** păstrează adresa primului element al listei şi că **p** şi **q** sunt două variabile de acelaşi tip cu **prim**, câte elemente are lista după executarea secvenței alăturate? **(4p.)**

```
p=prim;
while(p->urm!=NULL)
{q=p->urm;
 p->urm=q->urm;
 delete q;  |  free(q);
 p=p->urm;}
```

În memorie este construită o listă simplu înlănțuită, alocată dinamic, ale cărei elemente memorează în câmpul `lit` un caracter şi în câmpul `next` adresa elementului următor din listă sau valoarea `NULL` dacă nu există un element următor. Se ştie că această listă conține cel puțin trei elemente. Variabila `prim` reține adresa primului element al acestei liste, iar variabila `ultim` reține adresa ultimului element din această listă.
Scrieți secvența de program care interschimbă valorile reținute în câmpurile `lit` de la al doilea, respectiv de la ultimul element al acestei liste. În cazul în care veți folosi şi alte variabile decât cele date prin enunț, scrieți şi declarările necesare pentru aceste variabile.
Exemplu: dacă presupunem că inițial lista avea conținutul şi forma următoare:



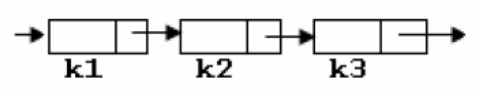
atunci, în urma executării secvenței, ea va avea următorul conținut:



(6p.)

O listă liniară simplu înlănțuită, alocată dinamic, reține în câmpul `info` al fiecărui element câte un număr întreg de cel mult `4` cifre, iar în câmpul `adr`, adresa elementului următor din listă. Lista are cel puțin trei noduri, iar variabila `p` reține adresa primului nod al listei. Scrieți, în limbajul `C/C++`, declarările ce definesc lista şi o singură instrucțiune prin a cărei executare se afişează pe ecran valoarea memorată în cel de-al treilea nod al listei. **(4p.)**

Fiecare element al unei liste simplu înlănțuite memorează în câmpul `urm` adresa următorului element din listă, iar variabilele `k1`, `k2` şi `k3`, reținadresele a `3` elemente succesive în listă, ca în figură. Cu ce instrucțiune se pot înlocui punctele de suspensie din secvența de mai jos astfel încât aceasta să determine interschimbarea corectă a pozițiilor în listă a elementelor de la adresele `k2` şi `k3`? **(4p.)**




```
k1->urm=k3; ... k3->urm=k2;
```

a. `k2->urm=k3->urm;`

b. `k2->urm=k3;`

c. `k2->urm=k1->urm;`

d. `k2=k3->urm;`