Colegiul National "Stefan cel Mare", Suceava
Strada Vasile Alecsandri, nr. 3, 720001, tel. 0230551342, fax. 0230551343, cnstefancelmare.ro
cn_stefan@yahoo.com, cn.stefan.sv@gmail.com

Lucrare semestrială la INFORMATICĂ
An şcolar 2018-2019, semestrul al II-lea
19-aprilie-2019

Numărul I

1 (6p)

Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă. Dacă **p**, **q** şi **r** sunt adresele a trei elemente din listă astfel încât:

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

atunci ordinea logică a elementelor în listă (elementele fiind identificate prin adrese) este:

a. **q, r, p** b. **p, r, q** c. **r, q, p** d. **p, q, r**

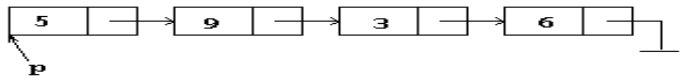
Într-o listă simplu înlănţuită, cu cel puţin două elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă, iar **q** memorează adresa penultimului element din listă. Dacă **p** reţine adresa unui element ce urmează a fi adăugat la sfârşitul listei şi **p->urm** are valoarea **NULL**, stabiliţi care dintre următoarele este o operaţie corectă de adăugare:

a. **p->urm=q;** b. **q->urm=p;** c. **q->urm->urm=p;** d. **p->urm->urm=q;**

Se consideră că variabila **prim** memorează adresa de început a unei liste liniare simplu înlănţuite nevide. Orice element al listei memorează în câmpul **urm** adresa elementului următor din listă. Dacă expresia **prim->urm** este diferită de **NULL** şi expresia **prim->urm->urm** are valoarea **NULL** atunci numărul de elemente din listă este egal cu:

a. **1** b. **0** c. **3** d. **2**

Într-o listă liniară simplu înlănţuită, fiecare element reţine în câmpul **urm** adresa următorului nod din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este reţinută în variabila **p**. Dacă în listă sunt memorate, în această ordine, numerele **5 9 3 6** ca în figura de mai jos



ce numere vor fi memorate în ordine în listă în urma executării următoarei secvenţe de instrucţiuni

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

a. **9,5,3,6** b. **5,9,6,3** c. **5,3,6,9** d. **5,3,9,6**

Se consideră lista simplu înlănţuită în care fiecare nod memorează în câmpul **nr** o valoare întreagă şi în câmpul **urm** adresa nodului următor. Un nod al listei este de tipul **nod**.

În listă sunt memorate, în această ordine, valorile **1,2,3,4,5,6,7**. Dacă variabila **p** reţine adresa primului element din listă, ce se va returna la apelul **f(p)** funcția?

```
int f(nod *p)
{if(p==NULL) return 0;
 else   if(p->nr%2==0)return 1+f(p->urm);
        else return f(p->urm);
}
```

a. **7** b. **3** c. **12** d. **0**

Lista simplu înlanţuită alocată dinamic din secvenţa alăturată are **10** noduri ce reţin in câmpul **urm** adresa nodului următor sau **NULL** dacă nu există un element următor, iar în câmpul **info** câte o valoare întreagă din intervalul **[1,10]**. Valorile sunt memorate în ordine crescătoare, astfel: primul nod conţine valoarea **1**, cel de-al doilea **2**, etc. Dacă **p** reţine iniţial adresa primului element al listei, ce valoare se afişează?

```
s=0;
while (p<>NULL)  {   s=s+p->info;   p=p->urm->urm; }
cout<<s;
```

a. **30** b. **55** c. **10** d. **25**

---

2 (3p)

Se consideră o listă liniară simplu înlănțuită, alocată dinamic, în care elementele sunt de tipul declarat mai jos:

```
struct nod{ int info;  nod * urm;  };
```

în care câmpul **info** memorează un număr întreg, iar câmpul **urm** memorează adresa următorului element al listei.

Să se scrie funcțiile C++

a) **int numarare(nod * p);** - care determină și returnează numărul de elemente memorate în lista pentru care primul element are adresa memorată în pointerul p.

b) **void inserare(nod * p, nod * q, int x);** care inserează valoarea **x** după elementul cu adresa memorată în **q** din lista pentru care primul element are adresa memorată în pointerul **p.**

c) **void sterge(nod * p, nod * q);** care elimină, dacă există, din lista pentru care primul element are adresa memorată în pointerul **p** elementul situat în listă după elementul cu adresa memorată în **q.**

Colegiul National "Stefan cel Mare", Suceava
Strada Vasile Alecsandri, nr. 3, 720001, tel. 0230551342, fax. 0230551343, cnstefancelmare.ro
cn_stefan@yahoo.com, cn.stefan.sv@gmail.com

Lucrare semestrială la INFORMATICĂ
An şcolar 2018-2019, semestrul al II-lea
19-aprilie-2019

Numărul II

Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **urm** adresa elementului următor din listă. Dacă **p**, **q** şi **r** sunt adresele a trei elemente din listă astfel încât:

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

atunci ordinea logică a elementelor în listă (elementele fiind identificate prin adrese) este:

a. **q, r, p** b. **r, q, p** c. **p, r, q** d. **p, q, r**

Într-o listă simplu înlănţuită, cu cel puţin patru elemente, fiecare element reţine în câmpul **adr** adresa elementului următor din listă, iar **q** este adresa ultimului element din listă. Atunci **p** este adresa antepenultimului element din listă dacă şi numai dacă este satisfăcută condiţia:

a. q->adr->adr==p **b.**p->adr==q **c.p->adr->adr==q d.q->adr==p->adr->adr**

Se consideră o listă simplu înlănţuită în care fiecare element memorează în câmpul **nr** un număr natural strict pozitiv şi în câmpul **urm** adresa elementului următor din listă. Lista memorează, în ordine, pornind de la primul element, valorile **1,2,2,3,3,3,4,4,4,4,5,5,5,5,5.** Ştiind că **prim** reţine adresa primului element al listei iar **p** este o variabilă de acelaşi tip cu **prim**, stabiliţi câte valori nule există în listă după executarea secvenţei

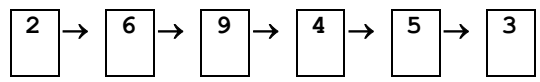
```
p=prim;
while(p->urm!=NULL)
 if(p->nr == p->urm->nr) {p->nr=0; p=p->urm->urm;}
 else p=p->urm;
```

a. **11** b. **5** c. **6** d. **7**

---

1 (6p)

Într-o listă simplu înlănţuită fiecare element reţine în câmpul **adr** adresa elementului următor din listă, iar în câmpul **inf** un număr întreg. Adresa primului element al listei este memorat în variabila **p**. Dacă în listă sunt memorate, în această ordine, numerele



Ştiind că variabila **c** este de acelaşi tip cu **inf,** în urma executării secvenţei de instrucţiuni:

```
c=p->adr->inf;
p->adr->inf=p->adr->adr-inf;
p->adr->adr->inf=c;
```

în listă vor fi memorate în ordine numerele:

a. **2 6 9 4 5 3** **b.6 9 4 5 3 2** **c. 2 9 6 4 5 3** **d. 6 2 9 4 5 3**

Într-o listă liniară simplu înlănţuită fiecare element reţine în câmpul **urm** adresa următorului nod din listă, iar în câmpul **info** un număr întreg. Adresa primului nod al listei este memorată în variabila **p**. Dacă în listă sunt memorate în această ordine numerele **7, 8, 9, 2, 0, 2, 9, 8, 7,** ce se va afişa în urma executării secvenţei de program alăturate?

```
int nr = 0;
while(p->urm->info!=0 && p) { p = p->urm;     nr++;    }
cout<<nr;
```

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

O listă liniară simplu înlănţuită alocată dinamic, în care fiecare element memorează în câmpul **nr** un număr întreg, iar în câmpul **urm** adresa elementului următor din listă, conţine exact trei elemente ale căror adrese sunt memorate în variabilele **p**, **q** şi **r**. Ştiind că: **q->nr=3**, **p->nr=5**, **r->nr=8**, **p->urm!=NULL** şi **r->urm=q**, care este ordinea numerelor din listă?

a. **8, 3, 5** b. **5, 8, 3** c. **3, 8, 5** d. **5, 3, 8**

Se consideră o listă liniară simplu înlănțuită, alocată dinamic, în care elementele sunt de tipul declarat mai jos:

```
struct nod{ int info;  nod * urm;  };
```

în care câmpul **info** memorează un număr întreg, iar câmpul **urm** memorează adresa următorului element al listei.
Să se scrie funcțiile C++

2 (3p)

a) **int suma(nod * p);** - care determina și returnează suma elementelor impare situate între două elemente pare din lista pentru care primul element are adresa memorată în pointerul **p**.

b) **void inserare(nod * p);** - care inserează după fiecare element par al unei liste pentru care primul element are adresa memorată în pointerul **p** dublul acelui element.

c) **void sterge(nod * & p, nod * q);** - care elimină, dacă există, din lista pentru care primul element are adresa memorată în pointerul **p** elementul cu adresa memorată în **q.**