BAREM
OFICIU ..... 10 puncte

Partea A ..... 30 puncte
A. 1. c, d..... **5 puncte**
A. 2. b ..... **5 puncte**
A. 3. a, d..... **5 puncte**
A. 4. a, c, d ..... **5 puncte**
A. 5. e..... **5 puncte**
A. 6. a, e ..... **5 puncte**

Partea B ..... 60 puncte
B. 1. Conversie..... 10 puncte
– respectarea antetului ..... 2 puncte
– condiția de oprire din recursivitate ..... 1 puncte
– valoarea returnată la oprirea recursivității ..... 1 puncte
– condiția pentru caracter diferit de cifră..... 2 puncte
– valoarea returnată atunci când condiția pentru caracter diferit de cifră este îndeplinită..... 2 puncte
– valoarea returnată atunci când condiția pentru caracter diferit de cifră nu este îndeplinită..... 2 puncte

```
Subalgoritm conversie (s, lung):
    Dacă lung > 0 atunci
       Dacă s[lung] ≥ 'A' atunci
          returnează conversie(s, lung - 1) * 16 + s[lung] - 'A' + 10
       altfel
          returnează conversie(s, lung - 1) * 16 + s[lung] - '0'
       SfDacă
    altfel
       returnează 0
    SfDacă
 SfSubalgoritm
```

B. 2. Cifre identice ..... 20 puncte
- V1: **generarea** numerelor având cifre identice **pe bază de calcule** ..... 20 puncte
  - respectarea parametrilor de intrare și ieșire ..... 2 puncte
  - generarea numerelor (cu o singură cifră și/sau ca multipli de 11, 111, 1111, ...)..... 16 puncte
  - salvarea numerelor cu cifre identice în vector ..... 2 puncte
- V2: **considerarea** și **verificarea** tuturor numerelor din intervalul [***a***, ***b***] ..... 10 puncte
  - respectarea parametrilor de intrare și ieșire ..... 2 puncte
  - verificarea dacă un număr are toate cifrele identice ..... 4 puncte
  - parcurgerea și verificarea tuturor numerelor din [***a***, ***b***]..... 2 puncte
  - salvarea numerelor cu cifre identice în vector ..... 2 puncte

B. 3. Roboțel plimbăreț..... 30 puncte
- V1: determinarea corectă a valorii prin calcule ***(n*n*n+n)/2*** ..... 30 puncte
  - respectarea parametrilor de intrare și ieșire ..... 2 puncte
  - calcul..... 14 puncte
  - prezentarea detaliată a metodei de obținere a formulei de calcul ..... 14 puncte
- V2: determinarea corectă a valorii prin simulare..... 25 puncte
  - respectarea parametrilor de intrare și ieșire ..... 2 puncte
  - parcurgerea celor *n* x *n* celule..... 4 puncte
  - deplasarea corectă (în cele 4 cazuri posibile) ..... 4x4 puncte
  - calculul sumei elementelor de pe diagonală ..... 3 puncte

Partea B (soluții) ..... 60 puncte
B. 1. Conversie..... 10 puncte

```
Subalgoritm conversie(s, lung):
    Dacă lung > 0 atunci
        Dacă s[lung] ≥ 'A' atunci
            returnează conversie(s, lung - 1) * 16 + s[lung] - 'A' + 10
        altfel
            returnează conversie(s, lung - 1) * 16 + s[lung] - '0'
        SfDacă
    altfel
        returnează 0
    SfDacă
SfSubalgoritm
```

B. 2. Cifre identice ..... 20 puncte

- generarea numerelor având cifre identice pe bază de calcule ..... 20 puncte

```
void cifreIdentice(int a, int b, int &k, int x[]){
        k = 0;
        for (int i = a; ((i < 10) && (i <= b)); i++){
                x[++k] = i;
        }
        int crt = 11;
        int ratie = 11;
        while (crt <= b){
                if (crt >= a)
                        x[++k] = crt;
                crt = crt + ratie;
                if (crt > 9 * ratie){
                        crt = ratie * 10 + 1;
                        ratie = crt;
                }
        }
}
```

B. 3. Robotel plimbăreț ..... 30 puncte

- determinarea corectă a valorii prin calcule (***n*n*n+n)/2*** ..... 30 puncte

9	3			
2				8
		13	7	1
	12	6	**5**	
11	10	4		

17	23			
24				18
		13	19	25
	14	20	21	
15	16	22		

		22	16	15
	21	20	14	
25	19	13		
18				24
			23	17

9	3	22	16	15
2	21	20	14	8
25	19	13	7	1
18	12	6	5	
11	10	4		17

Traseul robotelului este simetric față de centrul pătratului. Altfel spus, centrul pătratului este întotdeauna vizitat exact la mijlocul drumului robotelului (după (n*n-1)/2 mutări), iar a doua jumătate a drumului este obținută luând prima jumătate, inversând sensul de parcurgere și rotindu-l 180 grade față de centrul pătratului (de exemplu, ultimul pătrățel este mijlocul laturii din stânga). De aici rezultă că, pentru orice pereche de pătrățele simetrice față de centru, suma valorilor lor este n*n+1. Aplicând observația anterioara asupra elementelor de pe diagonală și grupându-le două câte două, rezultă formula sumă=n*(n*n+1)/2

Dacă *n* = 5, suma va fi 65

```
int obiecte(int n){
        return (n * n * n + n) / 2;
}
```