BAREM

OFICIU ..... 10 puncte

Partea A ..... 30 puncte

A. 1. Oare ce face? Răspunsul A ..... 5 puncte

A. 2. Calcul. Răspunsurile B, D..... 5 puncte

A. 3. Expresie logică. Răspunsurile A, B, D, E ..... 5 puncte

A. 4. Ce se afișează? Răspunsul D..... 5 puncte

A. 5. Număr norocos. Răspunsurile B, C..... 5 puncte

A. 6. Pune b. Răspunsurile A, B[1] ..... 5 puncte

Partea B ..... 60 puncte

B. 1. Calcul..... 10 puncte

- respectarea parametrilor de intrare și ieșire..... 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 în cazul unui caracter diferit de cifră ..... 2 puncte
- valoarea returnată în cazul unui caracter cifră..... 2 puncte

```
Subalgoritm calculCuCaractere(s, n, p, q, nr):
   Dacă p > q atunci
      returnează nr
    altfel
      Dacă s[p] < '0' sau s[p] > '9' atunci
         returnează nr + calculCuCaractere(s, n, p + 1, q, 0)
       altfel
         returnează calculCuCaractere(s, n, p + 1, q, 10 * nr + s[p] - '0')
      SfDacă
   SfDacă
SfSubalgoritm
```

B. 2. Perioadă ..... 25 puncte

- respectarea parametrilor de intrare și ieșire..... 2 puncte
- parcurgerea valorilor posibile ale perioadei ..... maxim 10 puncte

  Notă: punctajul acordat depinde și de următoarele aspecte:
  i. parcurgerea valorilor posibile ale perioadei
  ii. considerarea ca perioadă a divizorilor lui ***n***
- verificarea periodicității ..... maxim 13 puncte

  Notă: punctajul acordat depinde și de numărul de structuri repetitive folosite

B. 3. Robi grădină ..... 25 puncte

a. la un anumit moment de timp ***mt*** ($1 \leq mt \leq t$) se întâlnesc la izvor roboții care au valoarea ***q*** (egală cu suma dintre timpul necesar deplasării până la strat și înapoi, timpul necesar udării stratului și timpul necesar umplerii rezervorului) divizor al lui ***mt*** ..... 5 puncte

b. numărul minim de robinete suplimentare este egal cu maximul vectorului ***aux*** - 1, unde vectorul ***aux*** reține, pentru fiecare moment de timp, câți roboți se întâlnesc la izvor în momentul respectiv ..... 5 puncte

c. Dezvoltare subalgoritm

- V1: folosirea unui vector de frecvență pentru multiplii timpilor de lucru ai fiecărui robot..... 15 puncte
  a. respectarea parametrilor de intrare și ieșire ..... 2 puncte
  b. calcul timp de lucru (***q*** = 2 * deplasare + udare + încărcare)..... 2 puncte
  c. prelucrarea vectorului de frecvențe..... 5 puncte
  d. stabilirea frecvenței maximale ..... 4 puncte
  e. determinarea numărului de robinete suplimentare..... 2 puncte
- V2: simulare..... 10 puncte
  a. respectarea parametrilor de intrare și ieșire ..... 2 puncte
  b. calcul timp de lucru (***q*** = 2 * deplasare + udare + încărcare)..... 2 puncte
  c. structura repetitivă pentru timp..... 1 puncte
  d. structura repetitivă pentru roboți..... 1 puncte
  e. stabilirea numărului de robinete necesare la un anumit moment de timp..... 1 punct
  f. stabilirea numărului maxim de robinete ..... 1 punct
  g. determinarea numărului de robinete suplimentare..... 2 puncte

[1] in varianta subiectului în limba engleză, datorită traducerii ambigue a termenului *auto-apel*, a fost considerată corectă atât varianta cu răspunsurile A și B, cât și varianta cu răspunsul B.

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

```
Subalgoritm calculCuCaractere(s, n, p, q, nr):
   Dacă p > q atunci
      returnează nr
    altfel
      Dacă s[p] < '0' sau s[p] > '9' atunci
         returnează nr + calculCuCaractere(s, n, p + 1, q, 0)
       altfel
         returnează calculCuCaractere(s, n, p + 1, q, 10 * nr + s[p] - '0')
      SfDacă
   SfDacă
 SfSubalgoritm
```

B. 2. Perioadă maximală..... 25 puncte

```
bool verif(int n, char const x[], int perioada){
    for (int i = perioada; i < n; ++i) {
         if (x[i + 1] != x[i % perioada + 1])
                 return false;
    }
    return true;
}

int perioadaMax(int n, char x[]){
    int perioada = -1;
    for (int per = 2; per * per <= n; ++per){
         if (n % per == 0){ // perioada trebuie sa fie printre divizorii lui n
                 if (verif(n, x, n / per))
                         return n / per;

                 if ((per * per < n) && verif(n, x, per)) {
                         perioada = per;
                 }
         }
    }
    return perioada;
}
```

B. 3. Robi grădină ..... 25 puncte

```
int robiGradina(int n, int d[], int u[], int t){
    int aux[200000];      //aux(i) va retine cate robinete sunt necesare la momentul i
    int max = 1;
    for (int i = 1; i <= t; i++)
         aux[i] = 0;
    for (int j = 1; j <= n; j++){
         int q = d[j] * 2 + u[j] + 1;
                                        //se marchează multiplii lui q in vectorul aux
         for (int i = q; i <= t; i = i + q){
                 aux[i]++;
                 if (max < aux[i])                      //se determina maximul din aux
                         max = aux[i];
         }
    }
    return max - 1;
}
```