Rezolvare varianta 7

Subiectul I

1. a (x = -9)
2. a

```
* f(315) = return f (313)
      return f(31) + 1
          return f(3) + 1
              return f(1)
                  return f(0) + 1
                     return 0 =>
```

3. c

```
1+2+3+4
1+2+7
1+3+6
1+4+5
1+9
2+3+5
2+8
...
```

4. d Dupa desen se observa faptul ca ultimul graf are 2 componente conexe si NU poate fi un arbore.
5. c

Subiectul II

1.
   - a: Se afiseaza valoarea 24. Algoritmul formeaza un numar din ultima cifra a fiecarui prefix divizibil cu b al lui a
   - b: 8999 deoarece niciun prefix nu este divizibil cu 9.
   - c

```
#include <iostream>

using namespace  std;

int main() {
int a, b, p = 1, c = 0;
cin >> a >> b;
while (a > 0) {
    if (a % b == 0){
        c = c + a % 10 * p;
```

```
            p = p * 10;
        }
        a = a / 10;
    }
    cout << c;
    }
```

- d:

```
    citeste a,b (numere natural nenule)
    p <- 1
    c <- 0
    repeta
        daca a%b = 0 atunci
            c <- c + a%10 * p
            p <- p * 10

        a <- [a/10]
    pana cand a = 0
    scrie c
```

2.
```
    dn.z == e.d_nt.z && dn.l == e.d_nt.l && dn.a == e.d_nt.a
```

3. Se va afisa 10

```
        char s[]="bac2021";
        s[3]= s[3]-1; //bac1021
        strcpy(t,s+7); //t = ''
        strcpy(s+5,t); //bac10
        strcpy(t,s+3); // t = 10
        strcpy(s,t); //10
```

Subiectul III

1. Solutie:

```
#include <iostream>

using namespace  std;

int pare (int n, int a, int b, int v[]);
int main() {
    int n = 7, a = 14, b = 30;
    int v[] = {15, 20, 3, 12, 5, 45, 24};
```

```
    cout << pare(n, a, b, v);
}

int pare (int n, int a, int b, int v[]) {
    int contor = 0;
    for (int i = 0; i < n; i++) {
        if(v[i]% 2 == 0 && (v[i]>= a && v[i] <= b)) {
            contor++;
        }
    }
    return contor;
}
```

2. Solutie:

```
#include <iostream>

using namespace  std;

int main() {
    int m, n;
    cin >> m >> n;
    int matrice[m][n];
    int impare[]= {1,3,5,7,9};
    int pozitie = 0;

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            matrice[i][j] = impare[pozitie % 5];
            pozitie++;
        }
    }

    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            cout << matrice[i][j] << " ";
        }
        cout << endl;
    }
}
```

3. ◦ a:

```
#include <iostream>
#include <fstream>

using namespace  std;
```

```
int main() {
    ifstream fin("bac.in");
    int n;
    fin >> n;
    int numar;
    int aparitii = 0;
    int contor = 0;
    int contorTemporar = 0;
    while (fin >> numar) {
        if (numar == n) {
            aparitii++;
            contor += contorTemporar;
            contorTemporar = 0;
            continue;
        }
        if (aparitii >= 1 && numar % 2 == 1) {
            contorTemporar++;
        }
    }
    if (aparitii < 2) {
        cout << "nu exista";
    } else {
        cout << contor;
    }
}
```

- b:

```
- Eficienta programului din punct de vedere al memoriei este
datorata faptului ca in orice moment, noi tinem  doar 2 numere
din fisier in memorie, anume primul numar si dupa acceea fiecare
numar pe rand. Din punct de vedere al timpului de executie,
progrmaul este eficient deoarece efectuam o singura citire a
fisierului.
```