Varianta 2 de pe site-ul modinfo

Link varianta:

- https://www.modinfo.ro/bac/variante-test-2021/info/v2.pdf

Rezolvare

Subiectul I

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

Subiectul II

1.

- a: -1 21
- b: 10123, 11234, 12345, etc
- c:

```
#include <iostream>

using namespace std;
void f (int x);

int main()
{
    int n, s = 1, c1, c2;
    cin >> n;
    c1 = n % 10;
    n = n / 10;
    c2 = n % 10;

    if (c1 == c2) {
        s = 0;
    } else if (c1 < c2) {
        s = -1;
    }

    while ((c1 - c2) * s > 0 && n > 9) {
        c1 = n % 10;
        n = n/10;
        c2 = n % 10;
    }
```

```
        cout << s << " " << n;
    }
```

- d:

```
citeste n (numar natural, n > 9)
s <- 1
c1 <- n%10; n<-[n/10]; c2<-n%10
daca c1 == c2 atunci s<-0
altfel
daca c1 < c2 atunci s<- -1
daca ((c1-c2)* s) > 0 si n > 9 atunci:
repeta
    c1 <- n%10; n <- [n/10]; c2<-n%10;
pana cand ((c1-c2)* s) >= 0 si n <= 9
scrie s, ' ', n
```

2.
```
struct polinom {
    int grad;
    int coeficient[100];
} p
```

3.
```
#include <iostream>
#include <string.h>

using namespace std;

int main()
{
    char s[21];
    cin>> s;
    char sirVocale[]= "aeiouAEIOU";
    for(int i = 0; i < strlen(sirVocale);i++) {
        if (strchr(s, sirVocale[i]) == NULL) {
            cout << sirVocale[i];
        }
    }
}

    ```
```

Subiectul III

1. Solutie
```

```
#include <iostream>

using namespace std;
int factori(int n, int m);
int primeDivizorHasSamePower(int divizor, int n, int m);
int main()
{
    int n, m;
    cin >> n >> m;
    int result = factori(n, m);
    cout << result;
}

int factori(int n, int m) {
    int contorNumerePrime = 0;
    for (int i = 2; i <= n && i <=m; i++) {
        if (n%i == 0 && m % i == 0) {
            int estePrim = 1;
            for (int j = 2; j*j <=i;j++) {
                if (i % j == 0) {
                    estePrim = 0;
                    break;
                }
            }
            if (estePrim == 1 && primeDivizorHasSamePower(i, n, m)) {
                contorNumerePrime++;
            }
        }
    }
    return contorNumerePrime;
}

int primeDivizorHasSamePower(int divizor, int n, int m) {
    int counter1 = 0;
    int counter2 = 0;
    while(n % divizor == 0) {
        counter1++;
        n  = n / divizor;
    }
    while(m % divizor == 0) {
        counter2++;
        m  = m / divizor;
    }
    return counter1 == counter2;
}
```

2. Solutie:

```
#include <iostream>

using namespace std;
```

```
int main()
{
    int n;
    cin >> n;
    int matrice[n][n];

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cout << "Enter matrice["<<i<<"][" << j <<"]: ";
            cin >> matrice[i][j];
        }
        cout << endl;
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (j == (n-i-1)) {
                matrice[i][j] = n;
            } else if ( j < (n-i-1)) {
            matrice[i][j] = n - (n-i-1-j);
            } else {
                matrice[i][j] = n + (n-i-1-j);
            }
        }
    }

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

3.

- a. Limbaj natural
  - Un algoritm eficient pentru problema de fata ar fi urmatorul, numaram lungimea secventei pana la primul numar pozitiv si lungimea secventei care incepe dupa primul numar pozitiv (in ambele cazuri, si numarul pozitiv se va contoriza).
  - Eficienta algoritmului consta in faptul ca avem o singura parcurgere a fisierului.
- b. Solutie

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

    using namespace std;
```

```
int main()
{
    ifstream fin("bac.in");
    int amGasitPrimulNumarPozitiv = 0;
    int maxLen = 0;
    int currentLength = 0;
    while (!fin.eof()) {
        int currentNumber;
        fin >> currentNumber;
        if (currentNumber >= 0 && !amGasitPrimulNumarPozitiv) {
            amGasitPrimulNumarPozitiv  = 1;
            currentLength++;
            if (currentLength > maxLen) {
                maxLen = currentLength;
            }
            currentLength = 1;
        } else {
            currentLength++;
        }
    }
    if (currentLength > maxLen) {
        maxLen = currentLength;
    }

    cout << maxLen;
    fin.close();
}
```