REPUBLIKA E SHQIPËRISË
MINISTRIA E ARSIMIT
QENDRA E SHËRBIMEVE ARSIMORE

OLIMPIADA KOMBËTARE E INFORMATIKËS

NË ARSIMIN E MESËM TË LARTË

Faza e dytë

Klasa 12 **17 janar 2026**

Udhëzime për nxënësin:

- Olimpiada fillon në orën 10:00 dhe mbaron në orën 13:00.
- Testi përmban 5 pyetje.
- Për të zgjidhur secilin ushtrim nxënësi mund të përdorë gjuhën e programimit C/C++.

Për përdorim nga komisioni i vlerësimit

Pyetja	**1**	**2**	**3**	**4**	**5**
	12 pikë	**10 pikë**	**10 pikë**	**9 pikë**	**9 pikë**
Pikët e fituara					

Totali i pikëve të fituara ☐

KOMISIONI I VLERËSIMIT

1.....

2.....

1. Jepet një varg me n numra natyrorë, ku n ≤ 5000. Lejohet vetëm veprimi: zgjidh i<j dhe shkëmbe ai me aj maksimalisht C herë, ku C është një numër pozitiv. Shkruani një program që kontrollon nëse vargu mund të bëhet i renditur në rendin rritës. **12 pikë**

Zgjidhje:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
// Struktura per te ruajtur vleren dhe indeksin origjinal
struct Elementi {
    int vlera;
    int indeksi;
};
// Funksion ndihmes per te krahasuar dy elemente gjate renditjes
bool krahasoElementet(Elementi a, Elementi b) {
    if (a.vlera != b.vlera) {
        return a.vlera < b.vlera;
    }
    return a.indeksi < b.indeksi;
}
int main() {
    int n, C;
    cout << "Shkruani numrin e elementeve (n): ";
    cin >> n;
    cout << "Shkruani numrin maksimal te shkembimeve (C): ";
    cin >> C;
    // Perdorim nje array dinamik per te qene brenda standarteve
    vector<Elementi> vargu(n);
    cout << "Shkruani " << n << " numrat e vargut:" << endl;
```

```
for (int i = 0; i < n; i++) {
    cin >> vargu[i].vlera;
    vargu[i].indeksi = i;
}
// Rendisim vargun sipas vlerave
sort(vargu.begin(), vargu.end(), krahasoElementet);
// Vektori per te mbajtur shenim elementet e vizituara gjate cikleve
vector<bool> vizituar(n, false);
int shkembime_totale = 0;
for (int i = 0; i < n; i++) {
    // Nese elementi eshte ne vendin e duhur ose eshte vizituar, e anashkalojme
    if (vizituar[i] || vargu[i].indeksi == i) {
        continue;
    }
    // Gjejme madhesine e ciklit te permutacionit
    int madhesia_ciklit = 0;
    int j = i;
    while (!vizituar[j]) {
        vizituar[j] = true;
        j = vargu[j].indeksi; // Levizim te pozicioni ku duhet te shkoje ky element
        madhesia_ciklit++;
    }
    // Per nje cikel me gjatesi L, duhen L-1 shkembime
    if (madhesia_ciklit > 1) {
        shkembime_totale += (madhesia_ciklit - 1);
    }
}
cout << "\n----------------------------------" << endl;
cout << "Numri minimal i shkembimeve: " << shkembime_totale << endl;
if (shkembime_totale <= C) {
```

```
        cout << "REZULTATI: PO, vargu mund te renditet." << endl;
    } else {
        cout << "REZULTATI: JO, nuk mund te renditet (tejkalon C)." << endl;
    }
    cout << "-----------------------------------" << endl;
    // Ne Dev-C++ kjo ndihmon qe dritarja te mos mbyllet menjehere
    system("pause");
    return 0;
}
```

2. Në një listë me numra të plotë (pozitivë, negativë dhe zero), kërkohet të gjendet segmenti i pandërprerë me gjatësinë më të madhe, i tillë që produkti i të gjithë elementëve të tij të jetë pozitiv. Gjeni dhe afishoni këtë gjatësi, duke ditur se lista përmban deri në 20 elementë. **10 pikë**

Zgjidhje:

```
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n;
    cout << "Jepni numrin e elementeve (maksimumi 20): ";
    cin >> n;
    if (n > 20) n = 20; // Sigurohemi qe nuk kalon limitin
    long long vargu[20];
    cout << "Jepni elementet e vargut:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> vargu[i];
    }
    int gjatesiaMaksimale = 0;
    // Kontrollojme cdo segment te mundshem (i = fillimi, j = fundi)
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
```

```
            long long produkti = 1;
            bool kaZero = false;
            // Llogarisim produktin e segmentit nga i ne j
            for (int k = i; k <= j; k++) {
                produkti *= vargu[k];
                if (vargu[k] == 0) {
                    kaZero = true;
                    break;
                }
            }
            // Kontrollojme nese produkti eshte pozitiv
            if (!kaZero && produkti > 0) {
                int gjatesiaAktuale = j - i + 1;
                if (gjatesiaAktuale > gjatesiaMaksimale) {
                    gjatesiaMaksimale = gjatesiaAktuale;
                }
            }
        }
    }
    cout << "\n-----------------------------------------" << endl;
    cout << "Gjatesia me e madhe e segmentit me produkt pozitiv: " << gjatesiaMaksimale << endl;
    cout << "-----------------------------------------" << endl;
    system("pause");
    return 0;
}
```

3. Jepet një varg me n elementë natyrorë, ku n ≤ 500. Shkruani një program që gjen gjatësinë më të madhe të një nën-vargu ku numri i elementëve çift është sa dyfishi i numrit të elementëve tek. **10 pikë**

Zgjidhje:

```
#include <iostream>
```

```
#include <vector>
using namespace std;
int main() {
    int n;
    cout << "Jepni numrin e elementeve (n <= 500): ";
    cin >> n;
    // Kontrolli per limitin e n
    if (n > 500) {
        cout << "Nuk lejohet n me e madhe se 500." << endl;
        return 0;
    }
    int vargu[500];
    cout << "Jepni elementet e vargut:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> vargu[i];
    }
    int gjatesiaMaksimale = 0;
    // Kontrollojme te gjitha segmentet e mundshme
    // i eshte fillimi i segmentit
    for (int i = 0; i < n; i++) {
        int countCift = 0;
        int countTek = 0;
        // j eshte fundi i segmentit
        for (int j = i; j < n; j++) {
            // Kontrollojme nese elementi aktual eshte cift apo tek
            if (vargu[j] % 2 == 0) {
                countCift++;
            } else {
                countTek++;
            }
```

```
            // Kushti: Numri i cifteve sa dyfishi i numrit te tekeve
            if (countCift == 2 * countTek) {
                int gjatesiaAktuale = j - i + 1;
                if (gjatesiaAktuale > gjatesiaMaksimale) {
                    gjatesiaMaksimale = gjatesiaAktuale;
                }
            }
        }
    }
    cout << "\n----------" << endl;
    cout << "Gjatesia me e madhe qe ploteson kushtin: " << gjatesiaMaksimale << endl;
    cout << "----------" << endl;
    system("pause");
    return 0;
}
```

4. Të ndërtohet programi i cili analizon të dhënat e performancës sportive për 4 atletë, për 7 ditë stërvitje. Përdoruesi duhet të ketë mundësi të shtojë për çdo ditë numrin e kilometrave të vrapuara dhe numrin e minutave të stërvitjes. **9 pikë**

Programi duhet të afishojë:

a) Totalin e kilometrave të vrapuara nga të gjithë atletët për çdo ditë;

b) Ditën me minutat më të larta të stërvitjes për secilin atlet;

c) Atletin me kilometrat mesatare ditore më të larta gjatë 7 ditëve.

Zgjidhje:

```
#include <iostream>
#include <string>
#include <iomanip>
using namespace std;
int main() {
    const int ATLETET = 3;
```

```
const int DITET = 3;
double kilometrat[ATLETET][DITET];
int minutat[ATLETET][DITET];
// Futja e te dhenave
for (int i = 0; i < ATLETET; i++) {
    cout << "\n--- Shenimi i te dhenave per Atletin " << i + 1 << " ---" << endl;
    for (int j = 0; j < DITET; j++) {
        cout << "Dita " << j + 1 << " (Kilometra): ";
        cin >> kilometrat[i][j];
        cout << "Dita " << j + 1 << " (Minuta): ";
        cin >> minutat[i][j];
    }
}
// a) Totali i kilometrave per cdo dite
cout << "\n========================================" << endl;
cout << "a) Totali i kilometrave per cdo dite:" << endl;
for (int j = 0; j < DITET; j++) {
    double totaliDite = 0;
    for (int i = 0; i < ATLETET; i++) {
        totaliDite += kilometrat[i][j];
    }
    cout << "Dita " << j + 1 << ": " << totaliDite << " km" << endl;
}
// b) Dita me minutat me te larta per secilin atlet
cout << "\n========================================" << endl;
cout << "b) Dita me stervitjen me intensive (minuta):" << endl;
for (int i = 0; i < ATLETET; i++) {
    int maxMinuta = minutat[i][0];
    int ditaMax = 1;
    for (int j = 1; j < DITET; j++) {
```

```
            if (minutat[i][j] > maxMinuta) {
                maxMinuta = minutat[i][j];
                ditaMax = j + 1;
            }
        }
        cout << "Atleti " << i + 1 << ": Dita " << ditaMax << " (" << maxMinuta << " min)" << endl;
    }
    // c) Atleti me kilometrat mesatare ditore me te larta
    double maxMesatarja = -1;
    int atletiMeIPmiri = 0;
    for (int i = 0; i < ATLETET; i++) {
        double shumaKm = 0;
        for (int j = 0; j < DITET; j++) {
            shumaKm += kilometrat[i][j];
        }
        double mesatarjaAktuale = shumaKm / DITET;

        if (mesatarjaAktuale > maxMesatarja) {
            maxMesatarja = mesatarjaAktuale;
            atletiMeIPmiri = i + 1;
        }
    }
    cout << "\n========================================" << endl;
    cout << "c) Atleti me mesataren me te larte te kilometrave:" << endl;
    cout << "Atleti " << atletiMeIPmiri << " me mesatare: " << fixed << setprecision(2) << maxMesatarja << "
km/dite" << endl;
    cout << "========================================" << endl;
    system("pause");
    return 0;
}
```

5. Një numër quhet i veçantë nëse nuk ka dy shifra të njëpasnjëshme të barabarta dhe shuma e shifrave është e plotpjesëtueshme me 3. Shkruani një program që gjen sa numra të veçantë ka në [B, K], ku $10 \le B \le K \le 10^3$ dhe afishon shumën e tyre. **9 pikë**

Zgjidhje:

```
#include <iostream>

using namespace std;

// Funksion per te kontrolluar nese numri eshte "i vecante"

bool eshteIVecante(int n) {

   int shuma = 0;

   int temp = n;

   int shifraAktuale;

   int shifraEFundit = -1; // Ruajme shifren e meparshme per krahasim

   while (temp > 0) {

      shifraAktuale = temp % 10; // Marrim shifren e fundit

      shuma += shifraAktuale;    // E shtojme te shuma

      // Kontrollojme nese dy shifra te njejtat jane njera pas tjetres

      if (shifraAktuale == shifraEFundit) {

         return false;

      }

      shifraEFundit = shifraAktuale; // Azhornojme shifren e meparshme

      temp = temp / 10;              // Kalojme te shifra tjeter

   }

   // Kontrollojme nese shuma e shifrave plotpjesetohet me 3

   return (shuma % 3 == 0);

}

int main() {

   int B, K;

   cout << "Jepni vleren e B (B >= 10): ";

   cin >> B;
```

```
    cout << "Jepni vleren e K (K <= 1000): ";
    cin >> K;
    if (B < 10 || K < B) {
        cout << "Inpute te pasakta!" << endl;
        system("pause");
        return 0;
    }
    int count = 0;
    long long shumaTotale = 0;
    cout << "\nNumrat e vecante te gjetur: " << endl;
    for (int i = B; i <= K; i++) {
        if (eshteIVecante(i)) {
            cout << i << " ";
            count++;
            shumaTotale += i;
        }
    }
    cout << "\n\n-----------------------------------" << endl;
    cout << "Sasia e numrave te vecante: " << count << endl;
    cout << "Shuma e ketyre numrave: " << shumaTotale << endl;
    cout << "-----------------------------------" << endl;
    system("pause");
    return 0;
}
```