KUJDES! MOS DËMTO BARKODIN

BARKODI

REPUBLIKA E SHQIPËRISË

MINISTRIA E ARSIMIT
QENDRA E SHËRBIMEVE ARSIMORE

OLIMPIADA KOMBËTARE E INFORMATIKËS

NË ARSIMIN E MESËM TË LARTË

Faza e tretë **21 mars 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++.
- Zgjidhjet do të bëhen në kompjuter.

Për përdorim nga komisioni i vlerësimit

Pyetja	1	2	3	4	5
	11 pikë	10 pikë	9 pikë	11 pikë	9 pikë
Pikët e fituara					

Totali i pikëve të fituara ☐

KOMISIONI I VLERËSIMIT

1……....

2……....

1. Një kompani ka N satelitë në orbitë. Dy satelitë mund të komunikojnë nëse distanca midis tyre është ≤ D.
Në çdo minutë një satelit mund të dërgojë mesazh vetëm te një satelit tjetër. Një sinjal nis nga sateliti 1. Ndërtoni një program që gjen numrin minimal të minutave të nevojshme në mënyrë që të gjithë satelitët të marrin sinjalin. **11 pikë**

```
#include <iostream>
#include <vector>
#include <cmath>
#include <queue>

using namespace std;

struct Satelit {
    double x, y, z;
};

// Funksioni per llogaritjen e distances 3D
double llogaritDistancen(Satelit s1, Satelit s2) {
    return sqrt(pow(s1.x - s2.x, 2) + pow(s1.y - s2.y, 2) + pow(s1.z - s2.z, 2));
}

int main() {
    int N;
    double D;

    cout << "Jepni numrin e sateliteve (N): ";
    cin >> N;
    cout << "Jepni distancen maksimale te komunikimit (D): ";
    cin >> D;

    vector<Satelit> satelitet(N);
    for (int i = 0; i < N; i++) {
        cout << "Koordinatat per satelitin " << i + 1 << " (x y z): ";
        cin >> satelitet[i].x >> satelitet[i].y >> satelitet[i].z;
    }

    // Perdorim BFS per te gjetur kohen minimale te perhapjes
    vector<int> koha(N, -1); // -1 do te thote i paarsyetshem
    queue<int> q;
    // Nisja nga sateliti i pare (index 0)
    koha[0] = 0;
    q.push(0);
    int max_koha = 0;
    int te_vizituar = 0;

    while (!q.empty()) {
        int u = q.front();
        q.pop();
        te_vizituar++;

        if (koha[u] > max_koha) max_koha = koha[u];

        for (int v = 0; v < N; v++) {
            if (koha[v] == -1) { // nese nuk eshte vizituar
```

```
                if (llogaritDistancen(satelitet[u], satelitet[v]) <= D) {
                    koha[v] = koha[u] + 1;
                    q.push(v);
                }
            }
        }
    }

    // Kontrolli nese te gjithe e kane marre sinjalin
    if (te_vizituar < N) {
        cout << "\nSinjali nuk mund te arrije te te gjithe satelitet." << endl;
    } else {
        cout << "\nNumri minimal i minutave: " << max_koha << endl;
    }

    return 0;
}
```

2. Ndërtoni një program që lexon një skedar tekst mesazhi.txt dhe e kompreson atë duke përdorur kodimin Huffman. Programi duhet të kryejë veprimet e mëposhtme: **10 pikë**

a) të lexojë përmbajtjen e skedarit mesazhi.txt.
b) të llogarisë frekuencën e çdo karakteri.
c) të ndërtojë Pemën Huffman.
d) të gjenerojë kodin binar për çdo karakter, ku karakteret më të shpeshta marrin kodet më të shkurtra.
e) të shkruajë në skedarin kodimi.txt tabelën e kodeve dhe mesazhin e koduar në fornën e një vargu bitesh (shembull “0110..”).

```
#include <iostream>
#include <fstream>
#include <map>
#include <queue>
#include <string>

using namespace std;

// Nyja e pemes
struct Node {
    char ch;
    int freq;
    Node* left;
    Node* right;
    Node(char c, int f) {
        ch = c;
        freq = f;
        left = NULL;right = NULL;
    }
};

// Krahasimi per priority_queue
```

```
struct Compare {
    bool operator()(Node* a, Node* b) {
        return a->freq > b->freq;
    }
};

// Gjenerimi i kodeve
void generateCodes(Node* root,string code,map<char,string> &codes) {
    if (root == NULL)
        return;

    // Nyje flete
    if (root->left == NULL && root->right == NULL) {
        codes[root->ch] = code;
    }
    generateCodes(root->left, code + "0", codes);
    generateCodes(root->right, code + "1", codes);
}

int main() {
    ifstream in("mesazhi.txt");
    if (!in.is_open()) {
        cout << "Nuk u hap mesazhi.txt\n";
        return 0;
    }
    string text = "";
    char c;
    // a) Leximi i skedarit
    while (in.get(c)) {
        text += c;
    }
    in.close();
    if (text.length() == 0) {
        cout << "Skedari bosh!\n";
        return 0;
    }

    // b) Frekuenca
    map<char,int> freq;
    for (int i = 0; i < text.length(); i++) {
        freq[text[i]]++;
    }

    // c) Ndertimi i pemes Huffman
    priority_queue<Node*,
                   vector<Node*>,
                   Compare> pq;
```

```
for (map<char,int>::iterator it = freq.begin();
     it != freq.end();
     it++) {
    pq.push(new Node(it->first,
                     it->second));
}

while (pq.size() > 1) {
    Node* left = pq.top();
    pq.pop();
    Node* right = pq.top();
    pq.pop();
    Node* parent =  new Node('\0', left->freq + right->freq);
    parent->left = left;
    parent->right = right;
    pq.push(parent);
}
Node* root = pq.top();

// d) Gjenerimi i kodeve
map<char,string> codes;
generateCodes(root,  "", codes);

// e) Shkrimi ne kodimi.txt
ofstream out("kodimi.txt");
out << "Tabela e kodeve:\n";
for (map<char,string>::iterator it =
     codes.begin();
     it != codes.end();
     it++) {
    if (it->first == ' ')
        out << "' ' : " << it->second << endl;
    else if (it->first == '\n')
        out << "\\n : " << it->second << endl;
    else
        out << it->first  << " : " << it->second << endl;
}

out << "\nMesazhi i koduar:\n";

for (int i = 0; i < text.length(); i++) {
    out << codes[text[i]];
}
out.close();
cout << "Programi perfundoi me sukses.\n";
cout << "Shiko skedarin kodimi.txt\n";
return 0;
```

}

3. Një kompani e shpërndarjes së energjisë elektrike llogarit faturën mujore të klientëve në bazë të sasisë së energjisë së konsumuar, duke përdorur tarifa të ndryshme sipas intervaleve të konsumit. Nëse fatura nuk paguhet brenda 30 ditëve nga data e gjenerimit, për çdo ditë vonesë shtohet interes ditor. Duke marrë në konsideratë rregullat e faturimit, ndërtoni një program që llogarit: **9 pikë**

a) vlerën bazë të faturës.
b) shumën e kamatës për vonesë.
c) shumën totale për t'u paguar.

Rregullat e faturimit janë:

- *për 200 kWh e para, çmimi është 9 lekë/kWh;*
- *për 300 kWh pasuese, çmimi është 11 lekë/kWh;*
- *për çdo kWh mbi 500 kWh, çmimi është 14 lekë/kWh;*
- *nëse pagesa kryhet pas 30 ditëve, atëherë për çdo ditë vonesë shtohet 0.15% e faturës bazë si interes.*

```
#include <iostream>

using namespace std;

int main() {
   double kwh;
   int dite;
   double fatura_baze = 0;
   double kamate = 0;
   double total = 0;

   // Leximi i te dhenave
   cout << "Jep konsumin (kWh): ";
   cin >> kwh;
   cout << "Jep numrin e diteve pas gjenerimit: ";
   cin >> dite;

   // a) Llogaritja e fatures baze
   if (kwh <= 200) {
      fatura_baze = kwh * 9;
   }
   else if (kwh <= 500) {
      fatura_baze = 200 * 9 + (kwh - 200) * 11;
   }
   else {
      fatura_baze = 200 * 9 + 300 * 11 + (kwh - 500) * 14;
   }

   // b) Llogaritja e kamates
```

```
    if (dite > 30) {
        int dite_vonesa = dite - 30;
        kamate = fatura_baze * 0.0015 * dite_vonesa;
    }

    // c) Shuma totale
    total = fatura_baze + kamate;
    // Afishimi i rezultateve
    cout << "\nFatura baze: " << fatura_baze << " leke";
    cout << "\nKamata per vonese: " << kamate << " leke";
    cout << "\nShuma totale: " << total << " leke";
    return 0;
}
```

4. Ndërtoni një program që gjen nënstringun më të gjatë të përbashkët (Longest Common Substring) midis dy dokumenteve tekst shumë të mëdha. Programi duhet të injorojë hapësirat e tepërta. **11 pikë**

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

using namespace std;

// Heq hapësirat e tepërta
string pastroTekstin(string s) {

    string rezultat = "";
    bool hap = false;

    for (int i = 0; i < s.length(); i++) {

        if (s[i] != ' ') {
            rezultat += s[i];
            hap = false;
        }
        else {
            if (!hap) {
                rezultat += ' ';
                hap = true;
            }
        }
    }

    return rezultat;
}

// Longest Common Substring
```

```
string LCS(string X, string Y) {

   int m = X.length();
   int n = Y.length();

   // krijo matricë dinamike
   int **dp = new int*[m+1];

   for (int i = 0; i <= m; i++) {
      dp[i] = new int[n+1];
   }

   // inicializo
   for (int i = 0; i <= m; i++)
      for (int j = 0; j <= n; j++)
         dp[i][j] = 0;

   int maxLength = 0;
   int endPos = 0;

   for (int i = 1; i <= m; i++) {

      for (int j = 1; j <= n; j++) {

         if (X[i-1] == Y[j-1]) {

            dp[i][j] = dp[i-1][j-1] + 1;

            if (dp[i][j] > maxLength) {

               maxLength = dp[i][j];
               endPos = i;
            }
         }
         else {
            dp[i][j] = 0;
         }
      }
   }

   string rezultat = X.substr(endPos - maxLength, maxLength);

   // liro memorien
   for (int i = 0; i <= m; i++) {
      delete[] dp[i];
   }
   delete[] dp;
```

```
    return rezultat;
}

int main() {

    ifstream f1("doc1.txt");
    ifstream f2("doc2.txt");

    if (!f1.is_open() || !f2.is_open()) {

        cout << "Gabim ne hapjen e skedareve!\n";
        return 0;
    }

    string text1 = "";
    string text2 = "";
    string line;

    // Lexo dokumentin 1
    while (getline(f1, line)) {
        text1 += line + " ";
    }

    // Lexo dokumentin 2
    while (getline(f2, line)) {
        text2 += line + " ";
    }

    f1.close();
    f2.close();

    // Pastro hapësirat
    text1 = pastroTekstin(text1);
    text2 = pastroTekstin(text2);

    // Gjej substring
    string rezultat = LCS(text1, text2);

    cout << "Substringu me i gjate:\n";
    cout << rezultat << endl;

    cout << "Gjatesia: "
         << rezultat.length() << endl;

    return 0;
}
```

5. Një restorant dëshiron të monitorojë përdorimin e tavolinave dhe shpejtësinë e shërbimit, me qëllim që të analizojë fluksin e klientëve dhe të parashikojë sa shpejt lirohen vendet për vizitorët e rinj. Për çdo klient ose grup klientësh regjistrohen: numri i tavolinës, koha kur merret porosia dhe koha kur kryhet pagesa. Ndërtoni një program që llogarit: **9 pikë**

a) statusin e çdo tavoline.
b) kohën mesatare të shërbimit për orën e fundit.
c) kohën mesatare pas së cilës pritet të lirohet një tavolinë për vizitorët e rinj.

```
#include <iostream>

using namespace std;

int main() {

    int n;

    cout << "Jep numrin e regjistrimeve: ";
    cin >> n;

    int tavolina[100];
    int porosia[100];
    int pagesa[100];

    int status[100] = {0};

    int kohaTotale = 0;
    int numerOraFundit = 0;

    int kohaAktuale;

    cout << "Jep kohen aktuale (ne minuta): ";
    cin >> kohaAktuale;

    // Leximi i te dhenave
    for(int i = 0; i < n; i++) {

        cout << "\nRegjistrimi " << i+1 << endl;

        cout << "Numri tavolines: ";
        cin >> tavolina[i];

        cout << "Koha e porosise (min): ";
        cin >> porosia[i];

        cout << "Koha e pageses (min): ";
```

```
        cin >> pagesa[i];
        int kohaSherbimit = pagesa[i] - porosia[i];
        kohaTotale += kohaSherbimit;

        // kontroll per oren e fundit
        if(pagesa[i] >= kohaAktuale - 60) {
            numerOraFundit++;
        }

        // statusi tavolines
        if(pagesa[i] > kohaAktuale) {
            status[tavolina[i]] = 1; // e zene
        }
        else {
            status[tavolina[i]] = 0; // e lire
        }
    }
    // a) Statusi i tavolinave
    cout << "\nStatusi i tavolinave:\n";
    for(int i = 1; i <= 10; i++) {
        if(status[i] == 1)
            cout << "Tavolina " << i << " : E zene\n";
        else
            cout << "Tavolina " << i  << " : E lire\n";
    }

    // b) Mesatarja e ores se fundit

    double mesatareOraFundit = 0;

    if(numerOraFundit > 0) {

        mesatareOraFundit = (double)kohaTotale /  numerOraFundit;
    }

    cout << "\nKoha mesatare e sherbimit (ora e fundit): " << mesatareOraFundit << " minuta";

    // c) Koha mesatare e lirimit

    double mesatareLirim = (double)kohaTotale / n;

    cout << "\nKoha mesatare e lirimit te tavolines: "  << mesatareLirim << " minuta";

    return 0;
}
```