Ex1:

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

class HashMap {
public:
    int m_size = 0;
    std::unordered_map<int, std::string> m_map;
    void insert(std::pair<int, std::string>p) {
        m_map[p.first] = p.second;
        m_size++;
        checkLoadFactor();
    }

    void remove(int key) {
        if (m_map.find(key) != m_map.end()) {
            m_map.erase(key);
            m_size--;
        }
    }

    std::string find(int key) {
        if (m_map.find(key) != m_map.end())
            return m_map[key];
        return " ";
    }

    void print() {
        for (std::pair<int, std::string> elm : m_map)
            std::cout << "<" << elm.first << "," << elm.second << ">" << std::endl;
    }

private:
    void checkLoadFactor() {
        float loadFactor = m_size / m_map.bucket_count();
        if (loadFactor > 1.0) {
            std::unordered_map<int, std::string> newmap(m_map);
            m_map.swap(newmap);
        }
    }
};

int main()
{
    HashMap mappy;
    std::ifstream fin("Text.txt");
    int n;
    fin >> n;
    int key;
    std::string value;
    for (int i = 0; i < n; i++) {
        fin >> key >> value;
        mappy.insert(std::make_pair(key, value));
    }
    fin.close();
    int option;
    std::cout << "1. Adaugare element" << std::endl;
    std::cout << "2. Stergere element" << std::endl;
    std::cout << "3. Cautare element" << std::endl;
    std::cout << "4. Afisare elemente" << std::endl;
    std::cout << "0. Iesire" << std::endl;

    do {
        std::cout << "Alege o optiune: ";
        std::cin >> option;
        switch (option) {
        case 1: {
            int key;
```

```
            std::string value;
            std::cout << "Introdu cheia: ";
            std::cin >> key;
            std::cout << "Introdu valoarea: ";
            std::cin >> value;
            mappy.insert(std::make_pair(key, value));
            break;
        }
        case 2: {
            int key;
            std::cout << "Introdu cheia elementului de sters: ";
            std::cin >> key;
            mappy.remove(key);
            break;
        }
        case 3: {
            int key;
            std::cout << "Introdu cheia elementului de cautat: ";
            std::cin >> key;
            std::string value = mappy.find(key);
            if (value.empty()) {
                std::cout << "Elementul nu exista" << std::endl;
            }
            else {
                std::cout << "Valoarea elementului este " << value << std::endl;
            }
            break;
        }
        case 4: {
            mappy.print();
            break;
        }
        case 0:
            break;
        default:
            std::cout << "Optiune invalida" << std::endl;
            break;
        }
    } while (option != 0);

    return 0;
}
```

Ex3:

```
#include <iostream>
#include <fstream>
#include <unordered_set>
#include <string>
void citire_fisier(std::string& sir1, std::string& sir2) {
                std::ifstream fin("Text.txt");
                std::string line;
                std::getline(fin, line);
                sir1 = line;
                std::cout << sir1 << std::endl;
                std::getline(fin, line);
                sir2 = line;
                std::cout << sir2 << std::endl;
}

bool permutare(std::string sir1, std::string sir2) {
                std::unordered_set<char> set1, set2;
                if (sir1.length() != sir2.length())
                                return false;
                for (char c : sir1)
                                set1.insert(c);
                for (char c : sir2) {
                                if (set1.find(c) == set1.end())
                                                return false;
                                set2.insert(c);
```

```
		}
		if (set1.size() != set2.size())
			return false;
		return true;
}

int main()
{
		std::string sir1, sir2;
		citire_fisier(sir1, sir2);
		std::cout << "Momentul adevarului: " << permutare(sir1, sir2);

		return 0;
}
```

Ex4:

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

void participantiCompetitii(std::unordered_map <std::string, int>& participanti, std::vector<std::string>& multiParticipanti) {
		std::ifstream fin("date.in");
		std::string line;
		while (!fin.eof()) {
			std::getline(fin, line);
			participanti[line]++;
			if (participanti[line] > 1)
				if (std::find(multiParticipanti.begin(), multiParticipanti.end(), line) == multiParticipanti.end())
					multiParticipanti.push_back(line);
		}
		fin.close();
}

int main()
{
		std::unordered_map <std::string, int> participanti;
		std::vector<std::string>multiParticipanti;
		participantiCompetitii(participanti, multiParticipanti);
		std::cout << "Participantii care se regasesc in lista mai multor competitii: "<<std::endl;
		for (std::string elm : multiParticipanti)
			std::cout << " -" << elm << std::endl;
		return 0;
}
```

Ex5:

```
#include<iostream>
#include<fstream>
#include<unordered_map>
#include<vector>

void citire(int& Nr, std::vector<int>& vector) {
		int x;
		std::ifstream fin("Text.txt");
		fin >> Nr;
		for (int i = 0; i < Nr; i++) {
			fin >> x;
			vector.push_back(x);
		}

		fin.close();
}

void sumePartiale(int Nr, std::unordered_map<int, std::vector<std::pair<int, int>>>& sume_part, std::vector<int>& vector) {
		int sum;
		for (int i = 0; i < Nr; i++) {
```

```
                        sum = 0;
                        for (int j = i; j < Nr; j++) {
                                sum += vector[j];
                                sume_part[sum].push_back({ i,j });
                        }
                }
}
void afisarePerechi(std::unordered_map<int, std::vector<std::pair<int, int>>> sume_part) {
                int k;
                do {
                        std::cout << "Introduceti suma pe care o cautati: ";
                        std::cin >> k;
                        std::unordered_map<int, std::vector<std::pair<int, int>>>::iterator it = sume_part.find(k); //auto
                        if (it != sume_part.end()) {
                                for (auto p : it->second) {
                                        std::cout << "Intervale cu suma " << k << " sunt: ";
                                        std::cout << "[" << p.first << "," << p.second << "] ";
                                }
                                std::cout << std::endl;
                        }
                        else
                                if (k > 0)
                                        std::cout << "Suma invalida.Introduceti alta." << std::endl;
                } while (k > 0);
}

int main()
{
                int Nr = 0;
                std::vector<int>vector;
                std::unordered_map<int, std::vector<std::pair<int, int>>> sume_part;
                citire(Nr, vector);
                sumePartiale(Nr, sume_part, vector);
                afisarePerechi(sume_part);
                return 0;
}
```

Ex7:

```
#include <iostream>
#include <fstream>
#include <unordered_set>
#include <vector>

void citire(int& n, std::vector<int>& vector) {
                int x;
                std::ifstream fin("duplicate.in");
                fin >> n;
                while (!fin.eof()) {
                        fin >> x;
                        vector.push_back(x);
                }
}

void duplicateApropiate(int n, int dist, std::vector<int> vector) {
                int gasit = 0;
                std::cout << "dist=";
                std::cin >> dist;
                std::unordered_set<int> duplicate;
                for (int elem : vector) {
                        if (duplicate.size() == 0)
                                duplicate.insert(elem);
                        else {
                                if (std::find(duplicate.begin(), duplicate.end(), elem) != duplicate.end()) {
                                        int ok = 0, poz1 = 0, poz2 = 0;
                                        for (int i = 0; i < vector.size(); i++) {
                                                if (vector[i] == elem && ok == 0) {
                                                        poz1 = i;
                                                        ok = 1;
                                                }
```

```
                                                else
                                                        if (vector[i] == elem && ok == 1) {
                                                                poz2 = i;
                                                                break;
                                                        }
                                        }
                                        if (poz2 - poz1 <= dist) {
                                                std::cout << "Au fost gasite doua elemente cu valoarea " << elem << " la dist de " <<
poz2 - poz1 << std::endl;
                                                gasit = 1;
                                                return;
                                        }
                                }
                                else
                                        duplicate.insert(elem);
                        }
                }
                if (!gasit)
                        std::cout << "Nu au fost gasite doua elemente cu aceeasi valoare la o distanta mai mica sau egala cu "<<dist<<std::endl;
}

int main()
{
                int n = 0, dist = 0;
                std::vector<int> vector;
                citire(n, vector);
                duplicateApropiate(n, dist, vector);
                return 0;
}
```

Ex8:

```
#include <iostream>
#include <fstream>
#include <unordered_map>
#include <vector>
#include <unordered_set>
void citire(std::unordered_map<std::string, std::vector<std::string>>& magazine) {
                std::ifstream fin("magazine.in");
                std::string mag, prod;
                while (fin >> mag >> prod) {
                        magazine[mag].push_back(prod);
                }

}

void produseExclusive(std::unordered_map<std::string, std::vector<std::string>> magazine) {
                std::string mag_max;
                int count_max = 0;
                std::unordered_set<std::string> lista_prod_exclusive;

                for (const std::pair<const std::string, std::vector<std::string>>& pair : magazine) {
                        std::string magazin_curent = pair.first;
                        std::vector<std::string>produse = pair.second;
                        std::unordered_set<std::string> exclusive;
                        for (const std::string& produs : produse) {
                                int count = 0;
                                for (const std::pair<const std::string, std::vector<std::string>>& pair : magazine) {
                                        std::string alt_magazin = pair.first;
                                        std::vector<std::string>alte_produse = pair.second;
                                        if (alt_magazin != magazin_curent)
                                                if (std::find(alte_produse.begin(), alte_produse.end(), produs) != alte_produse.end())
                                                        count++;
                                }
                                if (count == 0)
                                        exclusive.insert(produs);
                        }
                        if (exclusive.size() > count_max) {
```

```
                                        mag_max = magazin_curent;
                                        count_max = exclusive.size();
                                        lista_prod_exclusive = exclusive;
                                }
        }
        std::cout << "Magazinul cu cele mai multe produse exclusive: " << mag_max << std::endl;
        std::cout << "Produsele exclusive sunt: ";
        for (const auto& produs : magazine[mag_max]) {
                if (std::find(lista_prod_exclusive.begin(), lista_prod_exclusive.end(), produs) != lista_prod_exclusive.end()) {
                        std::cout << produs << " ";
                }
        }
        std::cout << std::endl;
}

int main()
{
        std::unordered_map<std::string, std::vector<std::string>> magazine;
        citire(magazine);
        produseExclusive(magazine);
        return 0;
}
```