UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET

Zadaća 2

Tehnike programiranja

Student:	**Daris Mujkić**	Ocjena:	3.2
Grupa:	RI1-2b		
Broj indeksa:	19413		

Potpis:

Potpis tutora:

Zadatak . 1.cpp

```
//TP 2022/2023: Zada?a 2, Zadatak 1
#include <iostream>
#include <cmath>


int main ()
{


        return 0;
}
```

Zadatak . 2.cpp

```
// TP 2022/2023: Zada?a 2, Zadatak 2
#include <cmath>
#include <iostream>
#include <limits>

void RastavaBroja(int n, int &p, int &q) {
  long long int temp = n;
  n = std::abs(n);
  p = 1;
  q = 1;
  int i = 2;
  while (i <=std::sqrt(n)) // brojac broji koliko puta se neko broj ponavlja
  {                   // ponovi >1 = nije slobodan od kvadrata i nije u p
    int brojac = 0;
    while (n % i == 0) {
      n = n / i;
      brojac++;
    }
    if (brojac > 1) {
      q = q * std::pow(i, brojac / 2);
      p = p * std::pow(i, brojac % 2);
    } else if (brojac == 1)
      p = p * i;
    i++;
  }
  if (n > 1)
    p = p * n;
  // if (temp==std::numeric_limits<int>::min())
  // p=std::numeric_limits<int>::min();
  if (temp < 0)
    p = p * -1;
  if (temp == 0)
    p = 0;
}

int main() {
  std::cout << "Unesi broj: ";
  int n;
  std::cin >> n;
  if (n == 0) {
    std::cout << n << " = " << n << "*"
              << "1^2";
  }
  int p, q;
  RastavaBroja(n, p, q);

  if (n < 0) {
    std::cout << n << " = -" << std::abs(p) << "*" << q << "^2";
  }
  if (n > 0) {
    std::cout << n << " = " << std::abs(p) << "*" << q << "^2";
  }

  return 0;
}
```

Zadatak . 3.cpp

```
// TP 2022/2023: Zada?a 2, Zadatak 3
#include <cmath>
#include <iomanip>
#include <iostream>
#include <stdexcept>
#include <type_traits>
#include <vector>

template <typename T1, typename T2, typename f1, typename f2>
auto GeneraliziraniMatricniProizvod(std::vector<std::vector<T1>> A,
                                    std::vector<std::vector<T2>> B, f1 f,
                                    f2 g) {

  if /*(A.size()==0 || B.size()==0 || A.at(0).size()!=B.size()*/
      (A.at(0).size() != B.size()) {
    throw std::domain_error("Matrice nisu saglasne za mnozenje");
  }


  using tip =
      std::remove_reference_t<decltype(g(A.at(0).at(0), B.at(0).at(0)))>;
  std::vector<std::vector<tip>> C(A.size(), std::vector<tip>(0));

  //std::vector<std::vector<tip>> prazna;

  //PROBA
  //int redovi_druge=0;
  //for (int i=0;i<B.at(0).size();i++) redovi_druge++;
  //std::vector<std::vector<tip>> prazna(A.size(), std::vector<tip>(redovi_druge));
  //PROBA

  if (A.at(0).size()==0/* || B.size()==0*/) return C;
  for (int i=0;i<A.size();i++) C.at(i).resize(B.at(0).size());

  int m = A.size();       // broj redova matrice A
  int n = A.at(0).size(); // broj kolona matrice A
  int p = B.at(0).size(); // broj kolona matrice B

  for (int i=0;i<m;i++)
  {
      if (A.at(i).size() != B.size()) throw std::domain_error(
"Matrice nisu saglasne za mnozenje");
  }

  try {
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < p; j++) {
        tip sum = g(A.at(i).at(0), B.at(0).at(j));
        for (int k = 1; k < n; k++)
          sum = f(sum, g(A.at(i).at(k), B.at(k).at(j)));
        C.at(i).at(j) = sum;
      }
    }
  } catch (...) {
    throw std::runtime_error("Neocekivani problemi pri racunanju");
  }
  return C;
}

int main() {
  int m;
  int n;
  int p;
  std::cout << "Unesite broj redova prve matrice: ";
  std::cin >> m;
  std::cout
      << "Unesite broj kolona prve matrice, ujedno broj redova druge matrice: ";
  std::cin >> n;
  std::cout << "Unesite broj kolona druge matrice: ";
  std::cin >> p;
  std::vector<std::vector<std::string>> A(m, std::vector<std::string>(n));
  std::vector<std::vector<std::string>> B(n, std::vector<std::string>(p));
  std::cout << std::endl << "Unesite elemente prve matrice:";
  for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++)
      std::cin >> A.at(i).at(j);
  }
  std::cout << std::endl << "Unesite elemente druge matrice:";
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < p; j++)
      std::cin >> B.at(i).at(j);
  }
  auto f = [](const std::string &a, const std::string &b) {
    return a + "+" + b;
  };
  auto g = [](const std::string &a, const std::string &b) {
    return a + "*" + b;
  };

  try {
    auto C = GeneraliziraniMatricniProizvod(A, B, f, g);
    std::cout << std::endl << "Matricni proizvod:" << std::endl;
    for (const auto &red : C) {
      for (const auto &el : red)
        std::cout << el << "  ";
      std::cout << std::endl;
    }
  } catch (std::exception e) {
    std::cout << e.what() << std::endl;
  }

  return 0;
}
```

Zadatak . 4.cpp

```
// TP 2022/2023: Zada?a 2, Zadatak 4
#include <algorithm>
#include <cctype>
#include <cmath>
#include <iostream>
#include <iterator>
#include <vector>

template <typename tip>
bool Kriterij(std::vector<tip> a, std::vector<tip> b) {
  auto it1 = a.begin();
  auto it2 = a.end();
  auto it3 = b.begin();
  auto it4 = b.end();
  if (a.size()==0 || b.size()==0) return false;
  tip A=*a.begin(); tip B=*b.begin();

  for (auto i = it1 + 1; i != it2; i++) {
    A = A * (*i);
  }
  for (auto i = it3 + 1; i != it4; i++) {
    B = B * (*i);
  }
  if (A!=B) return A < B;
  return a<b;
}

template <typename tip>
void SortirajPoProizvoduRedova(std::vector<std::vector<tip>> &matrica) {
  std::sort(matrica.begin(), matrica.end(), Kriterij<tip>);
}

template <typename tip>
void IspisiMatricu(const std::vector<std::vector<tip>> &matrica) {
  for (auto &red : matrica) {
    for (auto &element : red) {
      std::cout << element << " ";
    }
    std::cout << std::endl;
  }
}

int main() {
  std::vector<std::vector<int>> matrica;
  std::cout
      << "Unesi elemente (* za kraj reda, * na pocetku reda za kraj unosa): "
      << std::endl;
  for (;;) {
    std::vector<int> red;
    int broj=0;
    for(;;) {
        std::cin>>broj;
        if(!std::cin){
            break;
```

```
          }
          red.push_back(broj);
      }
      std::cin.clear();
      std::cin.ignore(1000,'\n');
      if (red.size() == 0)
        break;
      matrica.push_back(red);
    }

    SortirajPoProizvoduRedova(matrica);
    std::cout << "Matrica nakon sortiranja: " << std::endl;
    IspisiMatricu(matrica);
    std::cout << "Unesite elemente sekvence koja se trazi (* za kraj reda): ";
    std::vector<int> sekvenca;
      for(;;) {
          int broj=0;
          std::cin>>broj;
          if(!std::cin)
          {
              std::cin.clear();
              std::cin.ignore(1000, '\n');
              break;
          }
          sekvenca.push_back((broj));
      }

    auto it =
        std::lower_bound(matrica.begin(), matrica.end(), sekvenca, Kriterij<int>);
    if (it == matrica.end() || *it != sekvenca)
      std::cout << "Trazena sekvenca se ne nalazi u matrici";
    else
      std::cout << "Trazena sekvenca se nalazi u " << (it - matrica.begin() + 1)
                << ". redu (nakon sortiranja)";
    return 0;
  }
```

Zadatak . 5.cpp

```
// TP 2022/2023: Zada?a 2, Zadatak 5
#include <array>
#include <cmath>
#include <iostream>
#include <iterator>
#include <list>
#include <stdexcept>
#include <type_traits>
#include <vector>
#include <deque>

template <typename iterator1, typename iterator2>
auto KreirajTablicuSabiranja(iterator1 pocetak1, iterator1 kraj1,
                             iterator2 pocetak2 /*, T** &matrica*/) {
  typename std::remove_reference<decltype(*pocetak1 + *pocetak2)>::type **matrica;
  int n = std::distance(pocetak1, kraj1);
  // auto matrica = new typename
  // std::remove_reference<decltype(*pocetak1+*pocetak2)>::type *[n]{};
try{
    matrica =
      new typename std::remove_reference<decltype(*pocetak1 + *pocetak2)>::type
          *[n] {}; // alociranje memorije za matricu
   }
   catch(...)
   {
       throw std::range_error("Nema dovoljno memorije!");
   }
 try{
     matrica[0] = new typename std::remove_reference<decltype(
      *pocetak1 + *pocetak2)>::type[(n * (n + 1)) / 2]{};
    }
    catch (...)
    {
        delete[] matrica;
        throw std::range_error("Nema dovoljno memorije!");
    }
  for (iterator1 it1 = pocetak1; it1 != kraj1;
       it1++) // provjera komutativnosti sabiranja
  {
    for (iterator2 it2 = pocetak2;
         it2 != pocetak2 + std::distance(pocetak1, it1); it2++) {
      if (*it2 + *it1 != *it1 + *it2)
      {
        delete[] matrica[0];
        delete[] matrica;
        throw std::logic_error("Nije ispunjena pretpostavka o komutativnosti");
      }
    }
  }
  int k = 1;
  for (int i = 1; i < n; i++) {
    matrica[i] = matrica[i - 1] + k;
    k++; // zavrseno alociranje
  }
```

```
    iterator2 temp = pocetak2;

    for (int i = 0; i < n; i++) {
      pocetak2 = temp;
      for (int j = 0; j <= i; j++) {
        matrica[i][j] = *pocetak1 + *pocetak2;
        if (j != i)
          pocetak2++;
      }
      if (i != n - 1)
        pocetak1++;
    }
    return matrica;
  }

  int main() {
    try {
        int duzina;
        std::cout<<"Duzina sekvenci: "<<std::endl;
        std::cin>>duzina;
        std::cout<<"Elementi prve sekvence: "<<std::endl;
        std::vector<double> el1(duzina);
        for (int i=0;i<duzina;i++) std::cin>>el1.at(i);
        std::cout<<"Elementi druge sekvence: "<<std::endl;
        std::deque<double> el2(duzina);
        for (int i=0;i<duzina;i++) std::cin>>el2.at(i);
      double **matrica (KreirajTablicuSabiranja(el1.begin(), el1.end(), el2.begin()));

      std::cout<<"Tablica sabiranja: "<<std::endl;
      for (int i = 0; i < el1.size(); i++) {
      for (int j = 0; j <= i; j++) {
        std::cout << matrica[i][j] << " ";
      }
      std::cout << std::endl;
    }

    delete[] matrica[0];
    delete[] matrica;
    } catch (std::logic_error &e) {
      std::cout << e.what() << std::endl;
      // return 1;
    } catch (std::range_error &e) {
      std::cout << e.what() << std::endl;
      // return 1;
    }
    /*for (int i = 0; i < a.size(); i++) {
      for (int j = 0; j <= i; j++) {
        std::cout << matrica[i][j] << " ";
      }
      std::cout << std::endl;
    }*/

    //for (int i = 0; i < a.size(); i++) // oslobadjanje memorije
    //{
      //delete[] matrica[0];
    //}
    //delete[] matrica;

    return 0;
  }
```