Екзамен з програмування (zалупа)

ТЕОРІЯ ЧИСЕЛ (пізда)

1. Написати функцію для знаходження натурального числа, що не перевищує N, з максимальною сумою власних дільників.
2. Натуральне число називають досконалим, якщо воно дорівнює сумі всіх своїх власних дільників, включаючи 1. Написати функцію для знаходження всіх досконалих чисел, що менші за N.
3. Написати функцію, яка для цілого числа знаходить кількість різних цифр у представленні цього числа за основою 16.
4. Написати функцію для знаходження цілих чисел-паліндромів, що не перевищують N та при підведені до квадрату також дають паліндроми (Наприклад: 22^2=484)
5. Написати функцію для знаходження всіх натуральних чисел , що не перевищують N,та які представляються у вигляді суми довільної кількості доданків, кожен з яких або A, або B.
6. Написати функцію для обчислення наближеного значення функції sin(x) з точністю ε за розкладом: $\sin(x) = x - x3/(3!) + x5/(5!) - \ldots + (-1)n \cdot x2 \cdot n+1/((2 \cdot n+1)!) + \ldots$. Враховувати лише доданки більші за модулем ε.

МАСИВИ ЧИСЕЛ (хуйня)

7. Написати функцію, яка для числового масиву A[n] та цілого число k ($1 \le k \le 4$, $k<n$) здійснює циклічний зсув праворуч на k позицій ($A1 \to Ak+1, \ldots, An \to Ak$). Дозволяється використовувати додатковий масив з 4 елементів.
8. Сума всіх простих чисел з масиву
9. Сума всіх різних чисел з масиву
10. Об’єднання впорядкованих множин
11. Максимальне число, яке зустрічається в матриці більше раза
12. Номери двох найближчих елементів
13. Мінімальний локальний максимум
14. Вектор з сум елементів діагоналей матриці, паралельних головній
15. Кількість елементів матриці, більших ніж сума їх стовпчика
16. Знайти кількість областей зі зростанням
17. Написати функцію, яка за матрицею A(n,n) будує матрицю B(n,n), де для всіх $1 \le i,j \le n$ bi,j – мінімальний елемент трикутника утвореного елементом ai,j та головною діагоналлю матриці A.

18. Комплексна матриця Z представлена парою дійсних матриць (X,Y) так, що Z = X + iY. Написати функцію для обчислення добутку двох комплексних матриць представлених (A,B) та (C,D) відповідно, тобто X + iY = (A + iB) * (C + iD).

19. Написати функцію, яка у заданого масиву цілих чисел A[N] знаходить найдовшу довжину та положення з найдовшої послідовності сусідніх нулів.

20. Написати функцію для обчислення суми всіх простих чисел, що зберігаються у масиві A[n]

21. Написати функцію, яка впорядковує масив дійсних чисел A[n], щоб спочатку йшли всі невід'ємні елементи, а потім від'ємні. Початковий взаємний порядок серед невід'ємних, а також серед від'ємних елементів повинен бути збережений.

22. Написати функцію, яка вилучає з масиву чисел A[n] всі однакові елементи, залишивши лише їх перші входження.

23. Сформувати масив A[N] перших 100 чисел (у зростаючому порядку) множини M, що визначаються наступним чином: - 1 належить M - якщо x належить M, то y = 2x+1 належить M, та z = 3x+1 належить M - ніякі інші числа не належать M (дописатиииИИИИ)

24. Написати функцію, яка у масиві з цілих чисел A[n] залишає лише елементи, що не є простим числом

25. Написати функцію, яка для масиву дійсних чисел A[n] створює «індексний» масив I[n], що містить порядкові номери елементів A[n] у відповідності до зростаючого порядку. Масив A[n] не змінюється.

26. Написати функцію, яка для матриці дійсних чисел A(n,n) обчислює вектор B(2n-1),елементи якого дорівнюють сумам елементів матриці, розташованих на відповідних діагоналях матриці паралельних до побічної діагоналі, включаючи її (починаючи з лівого верхнього кута матриці)

27. Написати функцію для знаходження кількості різних значень у масиві цілих чисел.

28. Для дійсної матриці розміром n*m визначити числа b1, b2, .., Ba , що є середніми арифметичними значень елементів рядків.

29. Написати функцію, яка для матриці дійсних чисел A(n,n) знаходить мінімальні значення елементів для кожної її діагоналі, що паралельна побічній (починаючи з правого нижнього кута матриці).

30. Написати функцію, яка перевіряє чи є квадратна матриця магічним квадратом

денис еблан 100% (туда його)заваліть пиздакиІДІ нахсамауй Садалбайобизакрийте пиздаки Фууу токсік рот нахуй.

РЯДКИ СІ (гавно)

31. Написати функцію, яка отримавши цілі додатні числа N1, N2 та рядки S1, S2 (рядки в «стилі С»), створює новий рядок (рядок в «стилі С»), що є злиттям перших N1 символів рядка S1 та останніх N2 символів рядка S2 .

32. Написати функцію, що вилучає з першого заданого символьного рядка V всі символи,які належать другому заданому рядку W (рядки в «стилі С»).

33. Написати функцію, яка для рядка (рядок в «стилі С») перевіряє чи співпадає його перше слово з останнім.

34. Довжина найкоротшого слова

35. Циклічне кодування Цезаря

36. Текстове представлення числа 100-999

37. Десяткове представлення дробу

38. Написати функцію, яка для рядків S1, S2 (рядки в стилі С) обчислює позицію першого входження рядка S1 до рядка S2, в якості підрядка. При цьому аналізуються лише перші n символів рядка S2)

39. Найбільш повторювана цифра в рядку

40. Порахувати різні слова в рядку

41. Десяткове у двійкове

42. Написати функцію, яка для заданого рядка (рядок в «стилі С»), що є записом виразу вигляду «<цифра>+-<цифра>+-…+-<цифра>», обчислює значення виразу (наприклад «4+7-2-8 -> 1»).

43. Написати функцію, яка для заданого рядка (рядок в «стилі С»), що містить повне ім’я файлу, повертає власне ім’я файлу (без розширення).

44. Написати функцію, яка для заданого рядка (рядок в «стилі С») здійснює шифрування, розташувавши спочатку символи з парних позицій заданого рядка (з збереженням порядку), а потім з непарних позицій (у оберненому порядку).

45. Написати функцію, яка для рядка (рядок в "в стилі С") перевіряє чи є він правильним ідентифікатором(містить лише латинські букви, цифри,"_", й не починаються з цифри, й не є порожнім). Повертати також інформацію про положення першого неприпустимого символу

46. Написати рекурсивну логічну функцію sim(s,i,j), для перевірки чи є симетричною частина рядка s, що починається i-м та закінчується j-м символами (рядок в "стилі С")

47. Написати функцію, яка для заданого рядка (рядок в "стилі С") підраховує кількість слів

48. Написати функцію, яка перевіряє чи можна перестановкою літер рядка X отримати рядок Y (рядки в «стилі С» складаються виключно з букв латинського алфавіту).

СТРУКТУРИ (жопа)

49. Час, менший даного на 25 сек
50. Різниця дат
51. Визначити потрібні типи даних й написати функцію, яка серед заданих n точок на площині обирає три з них, щоб периметр трикутника у обраних точках був найбільшим.
52. Максимально віддалені точки
53. Правильність ряду доміно
54. Перехід з полярних координат у декартові та навпаки.
55. Для представлення полів шахової дошки визначити відповідні типи даних й написати логічну функцію хідКоня(P,S), що перевіряє можливіть переходу коня з поля P до поля S.
56. Визначити потрібні типи даних й написати функцію sum(a, b), що здійснює додавання двох раціональних чисел a та b.
57. Для представлення полів шахової дошки визначити відповідні типи даних й написати логічну функцію хід ферзя (P, S), що перевіряє можливість переходу ферзя за один крок з поля P шахової дошки до поля S.

ФАЙЛИ (ААААААААА)

58. Прибрати останній рядок
59. Прибрати порожні рядки
60. Написати функцію, яка обчислює максимальну довжину рядків у текстовому файлі.
61. У вхідному тексті знайти цифру, що зустрічається найчастіше

== Теорія чисел ==

Написати функцію для знаходження натурального числа, що не перевищує N, з максимальною сумою власних дільників.

```
#include <iostream>
#include <cmath>

using namespace std;

int nDivMaxSum(int N)
{
   int maxSum = 0;
   int resN = 0;

   for (int n = 1; n <= N; ++n)
   {
       int divSum = 0;
       for (int div = 1; div <= sqrt(n); ++div)
       {
           if (n % div == 0)
           {
               divSum += div;
               if (n / div != div)
                   divSum += n / div;
           }
       }
       if (divSum > maxSum)
       {
           maxSum = divSum;
           resN = n;
       }
   }
   return resN;
}
int main()
{
   int N;
   cout << "Enter N: ";
   cin >> N;
   int res = nDivMaxSum(N);
   cout << res;
```

```
    return 0;
}
```

Натуральне число називають досконалим, якщо воно дорівнює сумі всіх своїх власних дільників, включаючи 1. Написати функцію для знаходження всіх досконалих чисел, що менші за N.

```
#include <iostream>

using namespace std;

bool isPerfectNumber(int num)
{
    int sum = 1;

    for (int i = 2; i <= num / 2; i++)
    {
        if (num % i == 0)
        {
            sum += i;
        }
    }

    return sum == num;
}

int main() {
    int n;
    cout << "Введіть n: ";
    cin >> n;
    cout << "Досконалі числа не більші за " << n << ":" << endl;

    for (int i = 2; i <= n; i++)
    {
        if (isPerfectNumber(i))
        {
            cout << i << " ";
        }
    }

    cout << endl;

    return 0;
}
```

Написати функцію, яка для цілого числа знаходить кількість різних цифр у представленні цього числа за основою 16.

```cpp
#include <iostream>

using namespace std;

int zalupa(int n)
{
   bool isNum [10] = {0};
   int temp, count=0;
   while (n != 0)
   {
       temp = n % 16;
       if (temp < 10 && !isNum [temp])
       {
           count++;
           isNum[temp] = true;
       }
       n /= 16;
   }
   return count;
}

int main()
{
   int n;
   cin >> n;
   cout << zalupa(n);
}
```

Написати функцію для знаходження цілих чисел-паліндромів, що не перевищують N та при підведені до квадрату також дають паліндроми (Наприклад: 22^2=484)

```cpp
#include <iostream>

using namespace std;

bool isPalindrome(int num) {
    int originalNum = num;
    int reversedNum = 0;
    while (num > 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
        num /= 10;
    }
    return originalNum == reversedNum;
}

void printPalindromes(int N) {
   cout << "Palindromes less than " << N << " are:\n";
    for (int i = 0; i < N; ++i) {
        if (isPalindrome(i))
            if (isPalindrome(i*i)) cout << i << " - " << i*i << "\n";
    }
}

int power(int base, int exponent) {
    if (exponent == 0) {
        return 1;
    }
    int halfPower = power(base, exponent / 2);
    int result = halfPower * halfPower;

    if (exponent % 2 == 1)
        result *= base;
    return result;
}
int main() {
    int N;

    cout << "Enter a number (N): ";
    cin >> N;

    printPalindromes(N);

    return 0;
}
```

Написати функцію для знаходження всіх натуральних чисел , що не перевищують N,та які представляються у вигляді суми довільної кількості доданків, кожен з яких або A, або B.

```
#include <iostream>
#include <cmath>
using namespace std;

void find(int n, int a, int b);

int main()
{
    int n, a, b;
    cin >> n >> a >> b;

    find(n, a, b);
}

void find(int n, int a, int b)
{
    int j_max = min(n / a, n);
    int k_max = min(n / b, n);

    for (int j = 1; j <= j_max; j++)
    {
        for (int k = 1; k <= k_max; k++)
        {
            int i = j * a + k * b;
            if (i <= n)
            {
                cout << i << " = " << a << " * " << j << " + " << b << "
* " << k << "\n";
            }
        }
    }
}
```

Написати функцію для обчислення наближеного значення функції sin(x) з точністю ε за розкладом: sin(x) = x – x3/(3!) + x5/(5!) – … + (–1)n·x2·n+1/((2·n+1)!) + …. Враховувати лише доданки більші за модулем ε.

```
#include <iostream>

using namespace std;

double res(double epsilon, double x);
int power(int n, int pow);
unsigned long long factorial(int n) {
    if (n < 0) {
        return 0; // Повертаємо 0 в разі некоректного вхідного значення
    }

    if (n == 0 || n == 1) {
        return 1; // Факторіал 0 і 1 дорівнює 1
    }

    unsigned long long result = 1;
    for (int i = 2; i <= n; ++i) {
        result *= i;
    }

    return result;
}

int main()
{
    double epsilon, x;
    cin >> x >> epsilon;

    cout << res(epsilon, x);
}

int power(int n, int pow)
{
    int res = 1;
    for (int i = 0; i < pow; i++)
    {
        res *= n;
    }

    return res;
}

double res(double epsilon, double x)
```

```
{
    double res = 0;
    double temp = 0;
    int n = 0;
    do
    {
        temp = res;
        res += pow(-1, n ) * pow (x, 2*n +1) / factorial(2 * n + 1);
        n++;
    } while (abs(res - temp) > epsilon);

    return res;
}
```

== Масиви чисел ==

Написати функцію, яка для числового масиву A[n] та цілого число k (1≤k≤4, k<n) здійснює циклічний зсув праворуч на k позицій ($A_1 \to A_{k+1}$, ..., $A_n \to A_k$). Дозволяється використовувати додатковий масив з 4 елементів.

```cpp
#include <iostream>
using namespace std;

void shift(double *A, int n, int k);

int main() {
    double A[] = {1,2,3,4,5,6,7,8,9,10};
    shift(A, 10, 4);

    for (int i = 0; i < 10; ++i) {
        cout << A[i] << " ";
    }

    return 0;
}

void shift(double *A, int n, int k){
    if(k >= n || k < 1 || k > 4) return; // перевірка правильності даних
    double memory[4];
    for (int i = n-k; i < n; ++i) {
        memory[(i-n+k)%k] = A[i]; // запис останніх k елементів масиву у додатковий масив memory
    }
    // для кожного елементу масиву
    for (int i = 0; i < n; ++i) {
        double t = A[i];

        // перезаписуємо на його місце число, що було k індексів раніше
        A[i] = memory[i%k];
        // записуємо оригінальне число у пам'ять
        memory[i%k] = t;
    }
}
```

Сума всіх простих чисел з масиву

```
#include <iostream>
using namespace std;
int sumOfPrimes(int A[], int n);

int main() {
    int n = 7;
    int A[] = {1, 2, 3, 40, 5, 6, 7};
    std::cout << sumOfPrimes(A, n) << std::endl;
    return 0;
}

int sumOfPrimes(int A[], int n){
    int sum = 0;

    for (int i = 0; i < n; ++i) {

        bool prime = true;

        for (int j = 2; j < A[i]; ++j) {
            if(A[i]%j == 0){
                prime = false;
            }
        }

        if(prime) sum += A[i];
    }

    return sum;
}
```

Сума всіх різних чисел з масиву

```cpp
#include <iostream>
using namespace std;

int sumOfDif(int A[], int n);
bool inArray(int el, int arr[], int n);

int main() {
    int n = 7;
    int A[] = {1, 3, 5, 1, 4, 3, 7};
    std::cout << sumOfDif(A, n) << std::endl;
    return 0;
}

int sumOfDif(int A[], int n){
    int summedNumbers[n];
    int c = 0;
    int sum = 0;
    for (int i = 0; i < n; ++i) {
        if(!inArray(A[i], summedNumbers, c)){
            summedNumbers[c] = A[i];
            sum += A[i];
            c++;
        }
    }
    return sum;
}

bool inArray(int el, int arr[], int n){
    for (int i = 0; i < n; ++i) {
        if(arr[i] == el) return true;
    }
    return false;
}
```

Об’єднання впорядкованих множин

```
#include <iostream>
using namespace std;

void cup(int *res, int A[], int n1, int B[], int n2);

int main() {
    int n1 = 7;
    int n2 = 5;
    int A[] = {1, 3, 6, 9, 13, 15, 21};t
    int B[] = {6, 8, 10, 24, 69};
    int C[n1+n2];
    cup(C, A, n1, B, n2);
    for (int i = 0; i < n1 + n2; ++i) {
        cout << C[i] << endl;
    }
    return 0;
}

void cup(int *res, int A[], int n1, int B[], int n2){
    int aCounter = 0;
    int bCounter = 0;

    int n = n1+n2;
    for (int i = 0; i < n; ++i) {
        if (aCounter != n1 && A[aCounter] < B[bCounter]){
            res[i] = A[aCounter];
            aCounter++;
        } else {
            res[i] = B[bCounter];
            bCounter++;
        }
    }
}
```

Максимальне число, яке зустрічається в матриці більше раза

```
#include <iostream>
using namespace std;

double maxRepeated(double **A, int n, int m);
bool inArr(double el, double arr[], int len);

int main() {
    int m = 3;
    int n = 3;
    double **arr;
    arr = new double *[m];
    for (int i = 0; i < m; ++i) {
        arr[i] = new double[n];
    }
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            arr[i][j] = i+j;
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
    cout << maxRepeated(arr, n, m) << endl;
    delete [] arr;
    return 0;
}

double maxRepeated(double **A, int n, int m){
    double best = INT_MIN;
    double candidates[n*m];
    int count = 0;
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            if(inArr(A[i][j], candidates, count)){
                if(A[i][j] > best) best = A[i][j];
            } else {
                candidates[count] = A[i][j];
                count++;
            }
        }
    }
    return best;
}
```

```
bool inArr(double el, double arr[], int len){
    for (int i = 0; i < len; ++i) {
        if(el == arr[i]) return true;
    }
    return false;
}
```

Номери двох найближчих елементів

```
#include <iostream>
#include <cmath>
using namespace std;

void nearest(double *res, double *A, int n);

int main() {
    double res[2];
    double a[] = {12, 15.9, 21, 2, 67, 14};
    nearest(res, a, 6);
    cout << res[0] << " " << res[1] << endl;
    return 0;
}

void nearest(double *res, double *A, int n){
    int n1 = 0;
    int n2 = 0;
    double lowestDistance = INT_MAX;

    for (int i = 0; i < n; ++i) {
        for (int j = i+1; j < n; ++j) {
            double dist = abs(A[i]-A[j]);
            if(dist < lowestDistance){
                n1 = i; n2 = j;
                lowestDistance = dist;
            }
        }
    }

    res[0] = n1;
    res[1] = n2;
}
```

Мінімальний локальний максимум

```cpp
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

struct fR{double min; int pos;};
void display(char *s, double *a, int n){cout <<s<<": ";for (int i = 0; i <
n; ++i) {cout << a[i] << " ";} cout << endl;}
fR minLocalMax(double *A, int n);

int main() {
    srand(time(0));
    int n = 10;
    double A[n];
    for (int i = 0; i < n; ++i) {
      A[i] = rand() % 10;
    }
    fR res = minLocalMax(A, n);
    cout << res.min << ", pos "<<res.pos<<endl;
    return 0;
}

fR minLocalMax(double *A, int n){
    display("Input", A, n);
    double min = INT_MAX;
    int pos = -1;

    double ass[n];
    for (int i = 0; i < n; ++i) {
        // local max
        double localMax = A[i];
        if(i != 0) localMax = max(localMax, A[i-1]);
        if(i != n-1) localMax = max(localMax, A[i+1]);
        //
        if(localMax < min) {
            min = localMax;
            pos = i;
        }
        ass[i] = localMax;
    }
    display("Local", ass, n);
    return fR{min, pos};
}
```

Вектор з сум елементів діагоналей матриці, паралельних головній

```
#include <iostream>
#include <stdlib.h>
#include <time.h>
using namespace std;

int n = 5;

void compute(double *S, double **A);

int main() {
    double S[59];
    // generating
    srand(time(0));
    double **A = new double *[n];
    for (int i = 0; i < n; ++i) {
        A[i] = new double [n];
    }
    cout << "Randomising...\n";
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            A[i][j] = rand() % 10;
        }
    }
    // input
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
    cout << "==================================\n";
    // f
    compute(S, A);
    // output
    for (int i = 0; i < 2*n-1; ++i) {
        cout << S[i] << " ";
    }

    return 0;
}

void compute(double *S, double **A){
    int sCounter = 0;
```

```
    // нижняя половина и главная диагональ
    for (int i = n-1; i >= 0; --i) {
        double s = 1;
        for (int p = i; p < n; ++p) {
            s *= A[p][p-i];
        }
        S[sCounter] = s; sCounter++;
    }
 // верхняя половина
    for (int i = 1; i < n; ++i) {
        double s = 1;
        for (int p = i; p < n; ++p) {
            s *= A[p-i][p];
        }
        S[sCounter] = s; sCounter++;
    }
}
```

Кількість елементів матриці, більших ніж сума їх стовпчика

```
#include <iostream>
using namespace std;
int biggerThanColumnCount(double **A, int m, int n){
    int count = 0;
    for (int i = 0; i < n; ++i) {
        double sum = 0;
        for (int k = 0; k < m; ++k) {
            sum += A[k][i];
        }
        for (int j = 0; j < m; ++j) {
            if(A[j][i] > sum-A[j][i]) {
                count++;
                break;
            }
        }
    }
    return count;
}

int main() {
    int m = 4;
    int n = 3;
    double **A = new double *[m];
    for (int i = 0; i < m; ++i) {
        A[i] = new double[n];
        for (int j = 0; j < n; ++j) {
            A[i][j] = (i-j)*i;
        }
    }
    for (int i = 0; i < m; ++i) {
        for (int j = 0; j < n; ++j) {
            cout << A[i][j] << " ";
        }
        cout << endl;
    }
    cout << biggerThanColumnCount(A, m, n) << endl;
    return 0;
}
```

Знайти кількість областей зі зростанням

```cpp
#include <iostream>
using namespace std;

int countGrowthZones(double *A, int n);

int main() {
    double A[] = {1,2,1,2};
    std::cout << countGrowthZones(A, 4) << std::endl;
    return 0;
}

int countGrowthZones(double *A, int n){
    int zones = 0;
    bool lastWasGrowing = false;
    for (int i = 0; i < n-1; i++) {
        if(A[i+1]-A[i] > 0){
            if(!lastWasGrowing){
                zones++;
                lastWasGrowing = true;
            }
        } else {
            lastWasGrowing = false;
        }
    }
    return zones;
}
```

Комплексна матриця Z представлена парою дійсних матриць (X,Y) так, що Z = X + iY. Написати функцію для обчислення добутку двох комплексних матриць представлених (A,B) та (C,D) відповідно, тобто X + iY = (A + iB) * (C + iD).

```
#include <iostream>

using namespace std;


void generateRandomMatrix(int rows, int cols, int** realmatrix, int**
imgmatrix) {
  for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
      realmatrix[i][j] = rand() % 100; // Generate random numbers between
0 and 99
      imgmatrix[i][j] = rand() % 100; // Generate random numbers between
0 and 99
    }
  }
}

void displayMatrix(int rows, int cols, int** realmatrix, int** imgmatrix)
{
  for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
      cout << realmatrix[i][j] << " + i" << imgmatrix[i][j] << "\t";
    }
    cout << "\n";
  }
  cout << "\n";
}

void deleteMatrix(int rows, int cols, int** realmatrix, int** imgmatrix)
{
  for (int i = 0; i < rows; ++i) {
    delete[] realmatrix[i];
    delete[] imgmatrix[i];
  }
  delete[] realmatrix;
  delete[] imgmatrix;
}

void product(int** real1, int** img1, int** real2, int** img2, int**
resreal, int** resimg, int row1, int row2, int col1, int col2);

int main()
{
```

```
  int row1 = 2, row2 = 2, col1 = 2, col2 = 2;

  int** real1 = new int* [row1];
  int** img1 = new int* [row1];
  for (int i = 0; i < row1; i++)
  {
    real1[i] = new int[col1];
    img1[i] = new int[col1];
  }

  int** real2 = new int* [row2];
  int** img2 = new int* [row2];
  for (int i = 0; i < row2; i++)
  {
    real2[i] = new int[col2];
    img2[i] = new int[col2];
  }

  int** resreal = new int* [row1];
  int** resimg = new int* [row1];
  for (int i = 0; i < row1; i++)
  {
    resreal[i] = new int[col2];
    resimg[i] = new int[col2];
  }

  // Generate and display the first random matrix
  cout << "Matrix 1:\n";
  generateRandomMatrix(row1, col1, real1, img1);
  displayMatrix(row1, col1, real1, img1);

  // Generate and display the second random matrix
  cout << "Matrix 2:\n";
  generateRandomMatrix(row2, col2, real2, img2);
  displayMatrix(row2, col2, real2, img2);

  product(real1, img1, real2, img2, resreal, resimg, row1, row2, col1,
col2);

  // Display the result matrix

  cout << "Result Matrix:\n";
  displayMatrix(row1, col2, resreal, resimg);

  deleteMatrix(row1, col1, real1, img1);
  deleteMatrix(row2, col2, real2, img2);
  deleteMatrix(row1, col2, resreal, resimg);
}
```

```
void product(int** real1, int** img1, int** real2, int** img2, int**
resreal, int** resimg, int row1, int row2, int col1, int col2)
{
  for (int i = 0; i < row1; ++i)
  {
    for (int j = 0; j < col2; ++j)
   {
      resreal[i][j] = 0;
      resimg[i][j] = 0;
      for (int k = 0; k < col1; ++k) {
        resreal[i][j] += real1[i][k] * real2[k][j] - img1[i][k] *
img2[k][j];
        resimg[i][j] += real1[i][k] * img2[k][j] + img1[i][k] *
real2[k][j];
      }
    }
  }
}
```

Написати функцію, яка у заданого масиву цілих чисел A[N] знаходить найдовшу довжину та положення з найдовшої послідовності сусідніх нулів

```
#include <iostream>

using namespace std;

void seq(int* arr, int size);

int main()
{
  int arr[] = { 0, 21, 324, 324, 0, 0, 0, 0, 0, 0, 23, 1, 0, 213, 43, 23,
0, 0, 0, 0, 1, 3, 234, 0, 0, 0, 0, 0};
  seq(arr, sizeof(arr) / sizeof(arr[0]));
}

void seq(int* arr, int size)
{
  int maxLenght = 0, maxStartElem = 0, maxEndElem = 0;;
  for (int i = 0; i <= size; i++)
  {
    if (arr[i] == 0)
    {
      int lenght = 1, startElem = i;
      for (int j = i+1; j <= size; j++)
      {
        if (arr[j] != 0)
        {
          if (lenght > maxLenght)
          {
            maxLenght = lenght;
            maxStartElem = startElem;
            maxEndElem = maxStartElem - 1 + maxLenght;
          }
          break;
        }
        lenght++;
        i = j-1;
      }
    }
  }

  cout << "Lenght: " << maxLenght << "\n";
  cout << "Diaposone: " << ++maxStartElem << " - " << ++maxEndElem <<
"\n";
}
```

Написати функцію для обчислення суми всіх простих чисел, що зберігаються у масиві A[n]

```cpp
#include <iostream>
#include <cmath>

using namespace std;

bool isPrime(int n)
{
    if(n % 2 == 0) return false;
    for(int i = 3; i < sqrt(n); i += 2)
        if(n % i == 0)
            return false;
    return true;
}
int main()
{
    int n, arr[1000];
    long long sumPrime = 0;
    cin >> n;
    for(int i = 0; i < n; i++)
    {
        cin >> arr[i];
        if(isPrime(arr[i])) sumPrime += arr[i];
    }

    cout << sumPrime;

    return 0;
}
```

Написати функцію, яка впорядковує масив дійсних чисел A[n], щоб спочатку йшли всі невід'ємні елементи, а потім від'ємні. Початковий взаємний порядок серед невід'ємних, а також серед від'ємних елементів повинен бути збережений.

```
#include <iostream>

using namespace std;

int main()
{
    int arr[] = { 10, 3, -2, -1, 3, 4, -1, 5, -4, 30, 94 };
    int size = sizeof(arr) / sizeof(arr[0]);
    for (int i = 0; i < size; i++)
    {
        if (arr[i] < 0)
        {
            int tempI = i;
            while (arr[tempI] < 0 && tempI != size)
            {
                tempI++;
            }
            if (tempI == size) break;
            for (int j = tempI; j > i; j--)
            {
                swap(arr[j], arr[j - 1]);
            }
        }
    }
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
}
```

Написати функцію, яка вилучає з масиву чисел A[n] всі однакові елементи, залишивши лише їх перші входження.

```
#include <iostream>

using namespace std;

void delSameElem(int* arr, int &size);

int main()
{
    int arr[] = { 1, 3, 4, 5, 2, 4, 13, 13, 3, 2, 1, 4, 0, 0, 0 };
    int size = sizeof(arr) / sizeof(arr[0]);
    delSameElem(arr, size);
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
}

void delSameElem(int* arr, int &size)
{
    for(int i = 0; i < size; i++)
    {
        for(int j = i+1; j < size;)
        {
            if(arr[j] == arr[i])
            {
                for(int k = j; k < size; k++)
                {
                    arr[k] = arr[k + 1];
                }
                size--;
            }
            else
            {
                j++;
            }
        }
    }
}
```

Сформувати масив A[N] перших 100 чисел (у зростаючому порядку) множини M, що визначаються наступним чином:

- 1 належить M
- якщо x належить M, то y = 2x+1 належить M, та z = 3x+1 належить M
- ніякі інші числа не належать M

Написати функцію, яка у масиві з цілих чисел A[n] залишає лише елементи, що не є простим числом

```
#include <iostream>
#include <cmath>

using namespace std;

bool isPrime(int n);
void delPrime(int* arr, int& size);

int main()
{
    int arr[] = { 1, 14, 16, 17};
    int size = sizeof(arr) / sizeof(arr[0]);

    delPrime(arr, size);
    for (int i = 0; i < size; i++)
    {
        cout << arr[i] << " ";
    }
}

bool isPrime(int n)
{
    if (n == 1) return false;
    if (n == 2) return true;

    if (n % 2 == 0) return false;

    for (int i = 3; i <= sqrt(n); i+=2)
    {
        if (n % i == 0) return false;
    }
    return true;
}

void delPrime(int* arr, int& size)
{
    for (int i = 0; i < size; i++)
    {
        if(isPrime(arr[i]))
        {
            for (int j = i; j < size - 1; j++)
            {
                swap(arr[j], arr[j + 1]);
            }
            size--;
            i--;
```

```
        }
    }

}
```

Написати функцію, яка для масиву дійсних чисел A[n] створює «індексний» масив I[n], що містить порядкові номери елементів A[n] у відповідності до зростаючого порядку. Масив A[n] не змінюється.

```cpp
#include <iostream>

using namespace std;

void build_index_array(const double arr[], int index[], int N) {
    // Ініціалізація індексованого масиву
    for (int i = 0; i < N; ++i) {
        index[i] = i; //0, 1, 2, 3, 4
    }

    // Сортування індексів за відповідними значеннями в масиві arr
(бульбашкове сортування)
    for (int i = 0; i < N - 1; ++i) {
        for (int j = 0; j < N - i - 1; ++j) {
            if (arr[index[j]] > arr[index[j + 1]]) {
                // Обмін індексів, якщо потрібно
                int temp = index[j];
                index[j] = index[j + 1];
                index[j + 1] = temp;
            }
        }
    }
}

int main() {
    // Приклад використання:
    const int N = 5;
    double A[N] = { 5.0, 2.0, 8.0, 3.0, 1.0};
    int I[N];

    // Виклик функції для побудови індексованого масиву
    build_index_array(A, I, N);

    cout << "Array A: ";
    for (int i = 0; i < N; ++i) {
        cout << A[i] << " ";
    }
    cout << "\n";

    cout << "Index array A: ";
    for (int i = 0; i < N; ++i) {
        cout << I[i] << " ";
    }
    return 0;
}
```

Написати функцію, яка для матриці дійсних чисел A(n,n) обчислює вектор B(2n-1),елементи якого дорівнюють сумам елементів матриці, розташованих на відповідних діагоналях матриці паралельних до побічної діагоналі, включаючи її (починаючи з лівого верхнього кута матриці)

```
#include <iostream>
#include <cstdlib>

using namespace std;

void sum(int** matrix, int* res, int sizem, int ressize);

int main()
{
  int size;
  cin >> size;
  int sizeVec = size * 2 - 1;
  int** matrix = new int*[size];

  for (int i =0; i < size; i++)
  {
    matrix[i] = new int[size];
    for (int j = 0; j < size; j++)
    {
      matrix[i][j] = rand() % 10 + 1;
      cout << matrix[i][j] << "\t";
    }
    cout << "\n\n";
  }

  int* vecMatrix = new int[sizeVec];

  sum(matrix, vecMatrix, size, sizeVec);

}

void sum(int** matrix, int* res, int sizem, int ressize)
{
  int sumUp = 0;
  int sumDown = 0;
  for (int i = 0; i < sizem; i++)
  {
    for (int j = 0; j <=i; j++)
    {
      sumUp += matrix[j][i - j];
      sumDown += matrix[sizem - 1 - i + j][sizem - 1 - j];
      //cout << matrix[j][i-j] << " ";
```

```
      //cout << matrix[sizem - 1 - i + j][sizem - 1 - j] << " ";
    }
    res[i] = sumUp;
    res[ressize - 1 - i] = sumDown;
    sumUp = 0;
    sumDown = 0;
    //cout << "\n";
  }


  for (int i = 0; i < ressize; i++)
  {
    cout << res[i] << " ";
  }
}
```

Написати функцію для знаходження кількості різних значень у масиві цілих чисел.

```
using namespace std;

int countDistinctValues(const int arr[], int size)
{
    int distinctCount = 0;

    for (int i = 0; i < size; ++i)
    {
        bool isDistinct = true;

        for (int j = 0; j < i; ++j)
        {
            if (arr[i] == arr[j])
            {
                isDistinct = false;
                break;
            }
        }

        if (isDistinct)
        {
            ++distinctCount;
        }
    }

    return distinctCount;
}

int main()
{
    const int size = 10;
    int arr[size] = {1, 2, 3, 4, 5, 2, 3, 6, 7, 8};

    int distinctCount = countDistinctValues(arr, size);

    cout << "Кількість різних значень в масиві: " << distinctCount <<
endl;

    return 0;
}
```

Для дійсної матриці розміром n*m визначити числа b1, b2, .., bn , що є середніми арифметичними значень елементів рядків.

```cpp
#include <iostream>
#include <locale>
// Функція для знаходження середнього арифметичного масиву
double average(int arr[], int size) {
    double sum = 0.0;
    for (int i = 0; i < size; ++i) {
        sum += arr[i];
    }
    return sum / size;
}

int main() {
    setlocale(LC_ALL, "ukr");
    using namespace std;
    const int N = 50, M = 50;
    int n, m;
    cout << "Введіть розміри матриці (n m): ";
    cin >> n >> m;

    int matrix[N][M];

    cout << "Введіть елементи матриці:" << endl;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cin >> matrix[i][j];
        }
    }

    double averages[N]; // Масив для збереження середніх значень

    cout << "Середні арифметичні значення елементів у рядках:" << endl;
    for (int i = 0; i < n; ++i) {
        averages[i] = average(matrix[i], m);
        cout << "Рядок " << i + 1 << ": " << averages[i] << endl;
    }

    return 0;
}
```

Написати функцію, яка для матриці дійсних чисел A(n,n) знаходить мінімальні значення елементів для кожної її діагоналі, що паралельна побічній (починаючи з правого нижнього кута матриці).

```
#include <iostream>

using namespace std;

void printDiagonalsParallelToSecondary(int** matrix, int sizem);

int main()
{
    int size;
    cout << "Enter the size of the matrix: ";
    cin >> size;

    int** matrix = new int* [size];

    for (int i = 0; i < size; i++)
    {
        matrix[i] = new int[size];
        for (int j = 0; j < size; j++)
        {
            matrix[i][j] = rand() % 10;
            cout << matrix[i][j] << "\t";
        }
        cout << "\n";
    }

    cout << "\nElements of diagonals parallel to the secondary
diagonal:\n";
    printDiagonalsParallelToSecondary(matrix, size);

    // Очищення пам'яті
    for (int i = 0; i < size; i++)
    {
        delete[] matrix[i];
    }
    delete[] matrix;

    return 0;
}

void printDiagonalsParallelToSecondary(int** matrix, int sizem)
{
    // Виведення елементів діагоналей, паралельних побічній
    for (int i = 0; i < sizem; i++)
    {
        double minVal = numeric_limits<double>::infinity();
```

```
        for (int j = 0; j <= i; j++)
        {
            double currentVal = matrix[sizem - 1 - i + j][sizem - 1 - j];
            if (currentVal < minVal)
            {
                minVal = currentVal;
            }
        }
        cout << "Min value on down diagonal " << i + 1 << ": " << minVal
<< endl;
    }
    cout << "====================================\n";
    for (int i = 0; i < sizem; i++)
    {
        double minVal = numeric_limits<double>::infinity();
        for (int j = 0; j <= i; j++)
        {
            double currentVal = matrix[j][i - j];
            if (currentVal < minVal)
            {
                minVal = currentVal;
            }
            //cout << matrix[sizem - 1 - i + j][sizem - 1 - j] << " ";
        }
        cout << "Min value on up diagonal " << i + 1 << ": " << minVal <<
endl;
    }
}
```

Написати функцію, яка перевіряє чи є квадратна матриця магічним квадратом

```
#include<iostream>

using namespace std;
const int N = 50;
bool magicSquare(int arr[][N], int n) {
  int rowSum = 0;
  //Перевірка суми за рядками
  for (int i = 0; i < n; i++) {
    int currentRowSum = 0;
    for (int j = 0; j < n; j++) {
      currentRowSum += arr[i][j];
    }
      if (i == 0) {
        rowSum = currentRowSum;
      }
      else if(currentRowSum!=rowSum){
        return false;

      }

  }
  //Перевірка суми по стовпцям
  for (int i = 0; i < n; i++) {
    int colSum = 0;
    for (int j = 0; j < n; j++) {
      colSum += arr[j][i];
    }
      if (colSum != rowSum) {
        return false;
      }

  }
  return true;
}

int main() {

  int n, arr[N][N];
  cin >> n;
  for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
      cin >> arr[i][j];
    }
  }
  if (magicSquare(arr, n)) {
    cout << "It`s a magic square!!!" << endl;
```

```
  }
  else {
    cout << "It is nota magic square. It`s ochko" << endl;
  }
}
```

Написати функцію, яка за матрицею A(n,n) будує матрицю B(n,n), де для всіх 1≤i,j≤n bi,j – мінімальний елемент трикутника утвореного елементом ai,j та головною діагоналлю матриці A.

/*Написати функцію, яка за матрицею A(n,n) будує матрицю B(n,n), де для всіх 1≤i,j ≤n bi,j – мінімальний елемент трикутника утвореного елементом ai,j та головною діагоналлю матриці A*/

```
#include<iostream>
using namespace std;

const int N = 25;

void arrB(int n, int arrayA[N][N], int arrayB[N][N]) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            int minVal = arrayA[i][j]; // Ініціалізуємо мінімальне
значення поточним елементом

            // Якщо елемент під головною діагоналлю
            if (j <= i) {
                // Перебираємо елементи у трикутнику
                for (int k = j; k <= i; k++) {
                    for (int l = k; l <= k; l++) {
                        if (arrayA[l][k] < minVal) {
                            minVal = arrayA[l][k];
                        }
                    }
                }
            }
            else { // Якщо елемент над головною діагоналлю
                // Перебираємо елементи у трикутнику
                for (int k = i; k <= j; k++) {
                    for (int l = i; l <= j; l++) {
                        if (arrayA[l][k] < minVal) {
                            minVal = arrayA[l][k];
                        }
                    }
                }
            }

            arrayB[i][j] = minVal; // Зберігаємо мінімальне значення у
arrayB
        }
    }
}

void printMatrix(int n, int matrix[N][N]) {
```

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

int main() {
    int n;
    int arrayA[N][N], arrayB[N][N];
    cout << "Enter size of array: ";
    cin >> n;
    cout << "Enter matrix elements:" << endl;

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> arrayA[i][j];
        }
    }

    arrB(n, arrayA, arrayB);

    cout << "\nMatrix B: \n";
    printMatrix(n, arrayB);

    return 0;
}
```

== Рядки Сі ==

Написати функцію, яка отримавши цілі додатні числа N1, N2 та рядки S1, S2 (рядки в «стилі С»), створює новий рядок (рядок в «стилі С»), що є злиттям перших N1 символів рядка S1 та останніх N2 символів рядка S2 .

```
#include <iostream>
#include <cstring>
using namespace std;

void mergeStrings(char *res, unsigned int N1, unsigned int N2, char
*S1, char *S2);

int main() {
    char S1[] = "Ame123";
    char S2[] = "12343643rica";
    char res[100];
    mergeStrings(res, 3, 4, S1, S2);
    cout << res << endl;
    return 0;
}

void mergeStrings(char *res, unsigned int N1, unsigned int N2, char
*S1, char *S2){
    // res - посилання на рядок, куди треба записати результат
    int counter = 0;
    int lenOfS1 = strlen(S1);
    int lenOfS2 = strlen(S2);
    if(lenOfS1 < N1) N1 = lenOfS1;
    if(lenOfS2 < N2) N2 = lenOfS2;

    // записуємо на перші N1 місць нового рядка перші N1 символи
першого даного рядка
    for (int i = 0; i < N1; ++i) {
        res[counter] = S1[i];
        counter++;
    }

    // записуємо на інші нового рядка останні N2 символи другого
даного рядка
    for (int i = lenOfS2-N2; i < lenOfS2; ++i) {
        res[counter] = S2[i];
        counter++;
    }
```

```
    // останім символом робимо знак закінчення рядка
    res[counter] = '\0';
}
```

Написати функцію, що вилучає з першого заданого символьного рядка V всі символи, які належать другому заданому рядку W (рядки в «стилі С»).

```
#include <iostream>
#include <cstring>

using namespace std;

void delSymb(char V[], const char W[])
{
   int lenV = strlen(V);
   int lenW = strlen(W);

   for (int i = 0; i < lenV; ++i)
   {
       for (int j = 0; j < lenW; ++j)
       {
           if (V[i] == W[j])
           {   // Якщо знайдено спільний символ, вилучаємо його
               for (int k = i; k < lenV - 1; ++k)
               {
                   V[k] = V[k + 1];
               }
               V[lenV - 1] = '\0';  // Закінчуємо рядок
               --i;  // Перевіряємо попередній символ
               break;
           }
       }
   }
}

int main() {
   const int Розмір_V = 100;
   char V[Розмір_V];
   char W[Розмір_V];

   // Введення рядків V та W
   cout << "Введіть рядок V: ";
   cin.getline(V, Розмір_V);

   cout << "Введіть рядок W: ";
   cin.getline(W, Розмір_V);

   // Вилучення символів
```

```
    delSymb(V, W);

    // Виведення результату
    cout << "Результат: " << V << endl;

    return 0;
}
```

Написати функцію, яка для рядка (рядок в «стилі С») перевіряє чи співпадає його перше слово з останнім.

```
#include <iostream>
#include <cstring>

using namespace std;

bool firstLastWord(const char penis[])
{
    int len = strlen (penis);
    int i = len;
    while (i >= 0 && penis[i] != ' ')
    {
        i--;
    }
    int j = 0;
    i++;
    while(j < i && penis[j] != ' ' && i < len)
    {
        if (penis[i] != penis[j])
            return false;
        i++;
        j++;
    }

    if(penis[j] == ' ' && i == len) return true;
    return false;

}

int main()
{
    char penis[255];
    cin.getline(penis, 255);

    if (firstLastWord(penis)) cout << "YESSSS";
    else cout << "ІДІ нахуй";
}
```

Довжина найкоротшого слова

```
#include <iostream>
using namespace std;

int shortestWordLength(char line[], int length);

int main() {
    cout << shortestWordLength("Hello   darkness   myn   friend",
30) <<endl;
    return 0;
}

int shortestWordLength(char line[], int length){
    int minLength = -1;
    int curLength = 0;
    bool lastCharWasSpace = false;
    for (int i = 0; i < length; ++i) {
        // пробел
        if(isspace(line[i])) {
            if(!lastCharWasSpace){
                if(curLength < minLength || minLength == -1)
minLength = curLength;
                lastCharWasSpace = true;
            }
            curLength = 0;
        }
        // символ
        else {
            curLength++;
            lastCharWasSpace = false;
        }
    }
    return minLength;
}
```

Циклічне кодування Цезаря

```
#include <iostream>
using namespace std;

void code(char res[], char str[], int n);

int main() {
    char str[] = "Helloworld";
    char res[10];
    code(res, str, 10);
    std::cout << str << " -> " << res << std::endl;
    return 0;
}


void code(char res[], char str[], int n){
    for (int i = 0; i < n; ++i) {
        int c = (int) str[i];
        if((c >= 65 && c <= 89) || (c >= 97 && c <= 121)) res[i] =
(char) (c + 1);
        else if (c == 90 || c == 121) res[i] = (char) (c - 25);
    }
}
```

Текстове представлення числа 100-999

```
#include <iostream>
#include <string.h>
using namespace std;

void toText(char res[], int n){
    int h = n/100;
    int t = (n-(h*100))/10;
    int u = n-(h*100)-(t*10);

    char hundreds[10][32] = {"", " сто", " двісті", " триста", "
чотириста", " п'ятсот", " шістсот", " сімсот", " вісімсот", "
дев'ятсот"};
    char tens[10][32] = {"", "", " двадцять", " тридцять", "
сорок", " п'ятдесят", " шістдесят", " сімдесят", " вісімдесят", "
дев'яносто"};
    char units[10][32] = {"", " один", " два", " три", " чотири", "
п'ять", " шість", " сім", " вісім", " дев'ять"};
    char teens[10][32] = {" десять", " одинадцять", " дванадцять",
" тринадцять", " чотирнадцять", " п'ятнадцять", " шістнадцять", "
сімнадцять", " вісімнадцять", " дев'ятнадцять"};

    strncat(res, hundreds[h], 32);

    if(t == 1) strncat(res, teens[u], 32);
    else {
        strncat(res, tens[t], 32);
        strncat(res, units[u], 32);
    }
}

int main() {
    system("chcp 65001");
    char bruh[100];
    toText(bruh, 999);
    cout << bruh;
    return 0;
}
```

Десяткове представлення дробу.Написати функцію, яка формує рядок (рядок в "стилі С") з точним десятковим представленням дробі 1/N (N-ціле число з інтервалом [2,50]), за правилами: послідовність закінчується з виявленням періоду; період виділяється *. Наприклад N=3 => ".*3*"; N=7 => ".*142857*"

```
#include <iostream>
using namespace std;

void decimalRepresentation(char *res, int N);

int main() {
        int N = 47;
        char res[N+3];
        decimalRepresentation(res, N);
        std::cout << res << std::endl;


    return 0;
}

void decimalRepresentation(char *res, int N){
    res[0] = '0';
    res[1] = '.';
    int c = 2;

    int remainders[N];
    int quotients[N];
    int r = 0;

    int dividend = 10;
    quotients[r] = 0;
    remainders[r] = 1;
    r++;
    // ---------
    while (true) {
        // добавляем нули к делимому если надо
        while (N > dividend && dividend != 0) {
            dividend *= 10;
            quotients[r] = 0;
            remainders[r] = -1;
            r++;
        }
```

```
        // делим
        int quotient = dividend / N;
        int remainder = dividend % N;
        dividend = remainder*10;
        quotients[r] = quotient;
        remainders[r] = remainder;
        r++;

        int found = -1; // ищем, был ли уже такой остаток
        for (int i = 0; i < r-1; ++i) {
            if (remainders[i] == remainder) found = i;
        }
        if (found != -1) {
            // если остаток был, перезаписываем все частные
            for (int i = 1; i < found+1; ++i) {
                res[c] = (char) (quotients[i] + 48);
                c++;
            }
            res[c] = '*';
            c++;
            for (int i = found+1; i < r; ++i) {
                res[c] = (char) (quotients[i] + 48);
                c++;
            }
            res[c] = '*';
            return;
        }
    }
}
```

АБО (варіант Кості)

```
#include <iostream>
#include <cstring>

using namespace std;

void period(char* per, int num);

int main()
{
    int n;
    cin >> n;
    char per[1000] = "\0";
    per[0] = '.';
```

```
    period(per, n);

    return 0;
}

void period(char* per, int num)
{
    int remainders[100] = { 0 };
    int rem = 1 % num;
    int index = 1;

    while (remainders[rem] == 0 && rem != 0)
    {
        remainders[rem] = index;
        per[index++] = rem * 10 / num + '0';
        rem = rem * 10 % num;
    }

    if (rem == 0)
    {
        per[index] = '*';
        per[index + 1] = '0';
        per[index + 2] = '*';
    }
    else
    {
        per[index] = '*';
        int size = strlen(per);
        char temp = per[remainders[rem]];
        per[remainders[rem]] = '*';

        for (int i = size; i > remainders[rem]; i--)
        {
            per[i + 1] = per[i];
        }

        per[remainders[rem] + 1] = temp;
    }

    cout << '0' << per << endl;
}
```

Пошук підрядка(написати функцію, яка для рядків S1, S2 (рядки в стилі C) обчислює позицію першого входження рядка S1 до рядка S2, в якості підрядка. При цьому аналізуються лише перші n символів рядка S2)

```
#include <iostream>
#include <cstring>

using namespace std;

bool findSubStr(char* str, char* subStr, int n)
{
    int len = strlen(str), lenSub = strlen(subStr);
    int end = min (n, len), idSub = 0, i = 0;
    while(i < end)
    {
        if(str[i] == subStr[idSub])
        {
            if(len - i < lenSub) return false;
            while(idSub < lenSub && str[i + idSub] ==
subStr[idSub])
                idSub++;
            if(idSub == lenSub) return true;
            idSub = 0;
        }
        i++;
    }
    return false;
}
int main()
{
    char str1[1024], str2[1024];
    cin >> str1 >> str2;
    int n;
    cin >> n;

    if(findSubStr(str1, str2, n)) cout << "Yeeee";
    else cout << "Nooo(";

    return 0;
}
```

Найбільш повторювана цифра в рядку

```
#include <iostream>
using namespace std;

int mostRepeatedDigit(char *s, int n){
    int numbers[10] = {0,0,0,0,0,0,0,0,0,0};
    for (int i = 0; i < n; ++i) {
        if(isdigit(s[i])) numbers[(int) s[i] - 48]++;
    }
    int maxN = 0, max = numbers[0];
    for (int i = 0; i < 10; ++i) {
        if(numbers[i] > max) {
            max = numbers[i]; maxN = i;
        }
    }
    return maxN;
}

int main() {
    std::cout << mostRepeatedDigit("2x5nb728y977727", 12) <<
std::endl;
    return 0;
}
```

АБО

```
#include <iostream>

using namespace std;

struct Tdomino{
    int start;
    int second;
    friend std::istream & operator>>(istream &is, Tdomino& p)
    {
        return is >> p.start >> p.second;
    }
    friend std::ostream & operator<<(ostream &os, Tdomino& p)
    {
        return os << p.start << " " << p.second << "\n";
```

```
    }
};

bool Corred (Tdomino *arr, int r)
{
    for(int i = 1; i < r-1; i++)
        if(arr[i].second != arr[i].start) return false;
    return true;
}

int main()
{
    int r;
    Tdomino arr[1000];
    cin >> r;
    for(int i = 0; i < r; i++)
        cin >> arr[i];
    if(Corred(arr, r)) cout << "Okay";
    else cout << "Noooooooooo ";
    return 0;
}
```

Порахувати різні слова в рядку

```
#include <iostream>
#include <string.h>
using namespace std;

int countDifferentWords(char *s, int n){
    char words[n][n];
    int wordCount = 0;
    int charCount = 0;
    bool wordInProgress = false;


    for (int i = 0; i < n; ++i) {
        char c = (char) tolower(s[i]);
        if(isalpha((c))) {
            words[wordCount][charCount] = c;
            charCount++;
        }
        if(!isalpha(c) || i==n-1) {
            if(charCount != 0){
                words[wordCount][charCount] = '\0';
                charCount = 0;
                bool found = false;
                for (int j = 0; j < wordCount; ++j) {
                    if(strcmp(words[j], words[wordCount]) == 0)
found = true;
                }
                if(!found){
                    wordCount++;
                }
            }
        }
    }

//    for (int i = 0; i < wordCount; ++i) {
//        cout << words[i] << endl;
//    }
    return wordCount;
}

int main() {
    char s[] = "This is the house that Jack built. This is the malt
that lay in the house that Jack built. This is the rat that ate the
```

```
malt. That lay in the house that Jack built.";
    std::cout << countDifferentWords(s, strlen(s)) << std::endl;
    return 0;
}
```

Десяткове в двійкове

```
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;

void decimalToBinary(char *res, char *dec);

int main() {
    char a[100];
    decimalToBinary(a, "3246");
    cout << "res: "<<a<<endl;
    return 0;
}

void decimalToBinary(char *res, char *dec){
    int digits = strlen(dec);
    long d = 0;
    for (int i = 0; i < digits; ++i) {
        d += ((int) dec[i]-48)*pow(10, digits-i-1);
    }
    int binDigits = 0;
    for (long i = 1; i <= d; i*=2) {
        binDigits++;
    }
    res[binDigits] = '\0';
    int c = binDigits-1;

    while(d >= 2){
        res[c] = (d%2)+'0';
        d /= 2;
        c--;
    }
    res[c] = d+'0';
}
```

Написати функцію, яка для заданого рядка (рядок в «стилі С»), що є записом виразу вигляду «<цифра>+-<цифра>+-...+-<цифра>», обчислює значення виразу (наприклад «4+7-2-8 -> 1»).

```
#include <iostream>

using namespace std;

int calc(char* str, int size);

int main()
{
    char str[] = "-32+0+10-24+23-342+2+4-353";
    cout << "\nsum: " << calc(str, strlen(str));
}


int calc(char* str, int size)
{
    int num = 0, sum = 0, sign = 1;
    for (int i = 0; i <= size; i++)
    {
        if(str[i] == '+' || str[i] == '-' || str[i] == '\0')
        {
            sign == 1 ? sum+= num : sum -= num;
            str[i] == '-' ? sign = -1 : sign = 1;
            num = 0;
        }
        else
        {
            num = num * 10 + (str[i] - '0');
        }
    }

    return sum;
}
```

Написати функцію, яка для заданого рядка (рядок в «стилі С»), що містить повне ім’я файлу, повертає власне ім’я файлу (без розширення).

```
#include <iostream>

using namespace std;

const char* extractFileName(const char* fullPath)
{
    const char* fileName = fullPath;

    // Find the last occurrence of the directory separator
    for (const char* p = fullPath; *p != '\0'; ++p)
    {
        if (*p == '\\' || *p == '/')
        {
            // Update the fileName pointer to the character after the
separator
            fileName = p + 1;
        }
        if (*p == '.')
        {
            *const_cast<char*>(p) = '\0';
        }
    }

    return fileName;
}

int main()
{
    const char path[] =
"C:\\Users\\Admin\\Desktop\\Programming\\Екзамен\\V15\\denysLox.xuy";
    const char* fileName = extractFileName(path);

    // Print the extracted file name
    std::cout << "File Name: " << fileName << std::endl;

    return 0;
}
```

Написати функцію, яка для заданого рядка (рядок в «стилі С») здійснює шифрування, розташувавши спочатку символи з парних позицій заданого рядка (з збереженням порядку), а потім з непарних позицій (у оберненому порядку).

```
#include <iostream>

using namespace std;

void shuffleCStyleString(char* str) {
    // Перевірка, чи рядок не є порожнім або не є nullptr
    if (str == nullptr || str[0] == '\0') {
        return;
    }

    // Знаходимо довжину рядка
    int size = strlen(str);
    for (int i = 0; i < size; i++)
    {
        if (i % 2 == 1)
        {
            for (int j = 0; j<=i/2; j++)
            {
                swap(str[i - j], str[i - 1 - j]);
            }
        }
        else if (i < size/2)
        {
            swap(str[i], str[size - i - 2 + size%2]);
        }
    }
}

int main() {
    // Приклад використання
    char cString[] = "123252729";
    std::cout << "Original: " << cString << std::endl;

    shuffleCStyleString(cString);

    std::cout << "Shuffled: " << cString << std::endl;

    return 0;
}
```

Написати функцію, яка для рядка (рядок в "в стилі С") перевіряє чи є він правильним ідентифікатором(містить лише латинські букви, цифри,"_", й не починаються з цифри, й не є порожнім). Повертати також інформацію про положення першого неприпустимого символу

```
#include <iostream>
#include <cctype>

using namespace std;

bool correct(char* str, int size);
bool isLetter(int number);

int main()
{
    char str[] = "ABd dsf  23 423 f sdsf gFsASFf"; // corect
    char str2[] = "ABd Пdsf  23 423 f sdsf gFsASFf"; // uncorect

    cout << static_cast<int>(str[0]) << "\n";
    if (correct(str, strlen(str))) cout << "true\n";
    if (!correct(str2, strlen(str2))) cout << "False\n";
}

bool correct(char* str, int size)
{
    for (int i = 0; i < size; i++)
    {
        if (isLetter((int)str[i]) || str[i] == ' ' ||
isdigit((int)(str[i])))
        {
            continue;
        }
        else
        {
            cout << "Uncorrect index: " << i << "\n";
            return false;
        }
    }
    return true;
}

bool isLetter(int number)
{
    if ((number <= 90 && number >= 65) || (number <= 122 && number >=
96)) return true;

    return false;
}
```

Написати рекурсивну логічну функцію sim(s,i,j), для перевірки чи є симетричною частина рядка s, що починається i-м та закінчується j-м символами (рядок в "стилі С")

```cpp
#include <iostream>

bool sim(const char* s, int i, int j) {
    // Базовий випадок: якщо i>=j, то це симетрична частина
    if (i >= j) {
        return true;
    }

    // Перевірка, чи поточні символи співпадають
    if (s[i] != s[j]) {
        return false;
    }

    // Рекурсивний виклик для наступної пари символів
    return sim(s, i + 1, j - 1);
}

int main() {
    const char* str = "fsdffsfabcbambvc";

    // Виклик рекурсивної функції для перевірки симетричності
    if (sim(str, 7, strlen(str) - 4)) {
        std::cout << "The substring is symmetric.\n";
    }
    else {
        std::cout << "The substring is not symmetric.\n";
    }

    return 0;
}
```

Написати функцію, яка для заданого рядка (рядок в "стилі С") підраховує кількість слів

```
#include <iostream>

using namespace std;

int countWord(char* str, int size);

int main()
{
    char str[] = "a        d sa       . ,     fd ssss. GF, fd ";

    cout << countWord(str, strlen(str));
}

int countWord(char* str, int size)
{
    int countt = 0;
    for (int i = 0; i < size; i++)
    {
        if ((i == 0 && str[i] != ' ') || (str[i] != ' ' && str[i-1] == '
' && str[i] != '.' && str[i] != ','))
        {
            countt++;
        }
    }
    return countt;
}
```

Написати функцію, яка перевіряє чи можна перестановкою літер рядка X отримати рядок Y (рядки в «стилі C» складаються виключно з букв латинського алфавіту).

```
#include <iostream>

using namespace std;

// Функція для перевірки можливості отримати рядок Y перестановкою літер рядка X
int canBePermutation(char X[], char Y[]) {
    // Перевірка довжини рядків
    int lenX = 0, lenY = 0;
    while (X[lenX] != '\0')
        lenX++;
    while (Y[lenY] != '\0')
        lenY++;
    if (lenX != lenY)
        return 0; // Рядки мають різну довжину, отже, неможливо отримати Y з X
    // Підрахунок кількості кожної літери в рядках X і Y
    int countX[26] = { 0 }; // Для літер a-z
    int countY[26] = { 0 };
    for (int i = 0; i < lenX; i++) {
        countX[X[i] - 'a']++;
        countY[Y[i] - 'a']++;
    }
    // Порівняння кількості кожної літери в рядках X і Y
    for (int i = 0; i < 26; i++) {
        if (countX[i] != countY[i]) {
            return 0; // Кількість літер не співпадає, отже, неможливо отримати Y з X
        }
    }

    return 1; // Можливо отримати Y з X перестановкою літер
}

int main() {
    char X[] = "listen";
    char Y[] = "silent";

    if (canBePermutation(X, Y))
        cout << "We can\n";
    else
        cout << "We cant`t\n";
    return 0;
}
```

== Структури ==

Час, менший даного на 25 секунд

```
#include <iostream>

struct time{
    int hours;
    int minutes;
    int seconds;
    int subtract25seconds(){
        seconds -= 25;
        if(seconds < 0) {
            seconds += 60;
            minutes -= 1;
            if(minutes < 0) {
                minutes += 60;
                hours -= 1;
                if(hours < 0) {
                    hours += 24;
                }
            }
        }
    }
};

int main() {
    time a = {23, 56, 56};
    a.subtract25seconds();

    std::cout << a.hours << ":" << a.minutes << ":" << a.seconds <<
std::endl;
    return 0;
}
```

Різниця дат

```
#include <iostream>
using namespace std;

int days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
struct date{
    int day;
    int month;
    int year;
};

long daysAnnoDomini(date d);
date daysToDate(long d);
date subtractDates(date a, date b);

int main() {
    date one{1, 1, 1970};
    date two{19, 12, 2020};
    date three = subtractDates(one, two);
    cout << abs(daysAnnoDomini(one) - daysAnnoDomini(two)) << endl;
    cout << three.year << " years, " << three.month << " months, "
<< three.day << " days." << endl;
    return 0;
}

long daysAnnoDomini(date d){
    long res = 0;
    res += d.year*365;
    // res += d.year/4;
    int month = d.month-1;
    for (int i = 0; i < month; ++i) {
        res += days[i];
    }
    res += d.day;
    return res;
}

date daysToDate(long d){
    date res;
    res.year = d/365;
    res.month = (d%365)/31;
    res.day = d-res.year*365-res.month*31;
    return res;
}
```

```
date subtractDates(date a, date b){
    return daysToDate(abs(daysAnnoDomini(a) - daysAnnoDomini(b)));
}
```

Визначити потрібні типи даних й написати функцію, яка серед заданих n точок на площині обирає три з них, щоб периметр трикутника у обраних точках був найбільшим.

```
#include <iostream>
#include <cmath>
#include <cstdlib>
using namespace std;
struct point
{
   int x;
   int y;
};
int perimetr (point a, point b, point c)
{
   int s1 = sqrt((a.x-b.x)*(a.x-b.x) - (a.y-b.y)*(a.y-b.y));
   int s2 = sqrt((b.x-c.x)*(b.x-c.x) - (b.y-c.y)*(b.y-c.y));
   int s3 = sqrt((a.x-c.x)*(a.x-c.x) - (a.y-c.y)*(a.y-c.y));
   return s1 + s2 + s3;
}
int hueta(int n, point * mas)
{
   int etalon = 0, temp = 0;
   for (int i = 0; i < n; i++)
   {
       for (int j = 0; j < n; j++)
       {
           for (int k = 0; k < n; k++)
           {
               temp = perimetr(mas[i], mas[j], mas[k]);
               if(etalon < temp) etalon = temp;
           }
       }
   }
   return etalon;
}

int main()
{
   int n;
   point mas[100];
   cin >> n;
   cout << "Point: ";
   for(int i = 0; i < n; i++)
   {
       mas[i].x = rand() % 100;
       mas[i].y = rand() % 100;
       cout << mas[i].x << " " << mas[i].y << "\n";
   }
   cout << "Preimetrs: ";
   cout << hueta(n, mas);
}
```

Максимально віддалені точки

```
#include <iostream>
#include <cmath>
using namespace std;

struct dot{
    double x;
    double y;
};

struct funcReturn{
    dot dots[2];
    double distance;
};

funcReturn biggestDistance(dot A[], int n1, dot B[], int n2);
double dist(dot a, dot b){ return sqrt(pow(a.x-b.x, 2)+pow(a.y-b.y,
2));}

int main() {
    dot A[] = {dot{2, 1}, dot{1, -1}, dot{-2, 0}, dot{-1, 2}};
    dot B[] = {dot{1, 12}, dot{24, -24}, dot{32, -42}, dot{-3, 10},
dot{-21, 21}};
    funcReturn a = biggestDistance(A, 4, B, 5);
    cout << a.dots[0].x << ":" << a.dots[0].y << ", " <<
a.dots[1].x << ":" << a.dots[1].y << " -> " << a.distance << endl;
    return 0;
}

funcReturn biggestDistance(dot A[], int n1, dot B[], int n2){
    funcReturn res{};
    res.distance = 0;
    for (int i = 0; i < n1; ++i) {
        for (int j = 0; j < n2; ++j) {
                double d = dist(A[i], B[j]);
                if (d > res.distance) {
                    res.dots[0] = A[i];
                    res.dots[1] = B[j];
                    res.distance = d;
                }
        }
    }
    return res;
}
```

Правильність ряда доміно

```
#include <iostream>
using namespace std;

struct domino{
    int left;
    int right;
};

bool Correct(domino *A, int n){
    int previous = 0;
    for (int i = 0; i < n; ++i) {
        if(A[i].left != previous && i != 0) return false;
        previous = A[i].right;
    }
    return true;
}

int main() {

    domino A[4] =
{domino{1,1},domino{2,1},domino{1,2},domino{2,2}};
    cout << Correct(A, 4);
    return 0;
}
```

АБО

```
#include <iostream>

using namespace std;

struct Tdomino{
    int start;
    int second;
    friend std::istream & operator>>(istream &is, Tdomino& p)
    {
        return is >> p.start >> p.second;
    }
    friend std::ostream & operator<<(ostream &os, Tdomino& p)
    {
        return os << p.start << " " << p.second << "\n";
```

```
    }
};

bool Corred (Tdomino *arr, int r)
{
    for(int i = 1; i < r-1; i++)
        if(arr[i].second != arr[i].start) return false;
    return true;
}

int main()
{
    int r;
    Tdomino arr[1000];
    cin >> r;
    for(int i = 0; i < r; i++)
        cin >> arr[i];
    if(Corred(arr, r)) cout << "Okay";
    else cout << "Nooooooooo ";
    return 0;
}
```

Для представлення полів шахової дошки визначити відповідні типи даних й написати логічну функцію хідКоня(P,S), що перевіряє можливіть переходу коня з поля P до поля S.

```
#include <iostream>
#include <cstring>

using namespace std;

struct Tpoint
{
    int x;
    int y;

    friend istream & operator >>(istream &is, Tpoint &p)
    {
        is >> p.x >> p.y;
    }

    friend ostream & operator <<(ostream &os, Tpoint &p)
    {
        os << p.x << " " << p.y << "\n";
    }
};

bool  checkMove(Tpoint Figure, Tpoint movePoint)
{
    if(abs(Figure.x - movePoint.x) == 2 && abs(Figure.y - movePoint.y) ==
1) return true;
    if(abs(Figure.x - movePoint.x) == 1 && abs(Figure.y - movePoint.y) ==
2) return true;
    return false;
}
int main()
{
    Tpoint  Horse, movePoint;
    cin >> Horse >> movePoint;

    if(checkMove(Horse, movePoint)) cout << "Yes\n";
    else cout << "No\n";
}
```

Визначити потрібні типи даних й написати функцію sum(a, b), що здійснює додавання двох раціональних чисел a та b.

```
#include <iostream>
using namespace std;

struct rational
{
    int numerator;      //chiselnyk
    int denominator;    //znamennyk
};
int findNSD (int a, int b) // алгоритм евкліда
{
    while (b != 0)
    {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
rational simplify (rational num)
{
    int nsd = findNSD(num.numerator, num.denominator);
    num.numerator /= nsd;
    num.denominator /= nsd;
    return num;
}
rational sum (rational a, rational b)
{
    rational res;
    res.numerator = a.numerator * b.denominator   +   b.numerator *
a.denominator;
    res.denominator = a.denominator * b.denominator;
    res = simplify (res);
    return res;
}

int main()
{
    rational a = {1, 2};     // 1/2
    rational b = {2, 3};     // 2/3
    rational result = sum(a, b);

    cout << "Сума: " << result.numerator << "/" << result.denominator <<
endl;

    return 0;
}
```

АБО

```
#include <iostream>

using namespace std;

struct rationalNum
{
    int num;
    int deNum;

    friend std::ostream & operator<<(std::ostream &os, const rationalNum&
p)
    {
        return os << p.num << '/' << p.deNum << "\n";
    }

    void cut()
    {
        int tempMin = this->num < this->deNum ? this->num : this->deNum;
        int tempMax = this->num > this->deNum ? this->num : this->deNum;
        while (tempMin != 0)
        {
            tempMax %= tempMin;
            swap(tempMax, tempMin);
        }

        this->num /= tempMax;
        this->deNum /= tempMax;
    }
    rationalNum operator+(const rationalNum& a)
    {
        rationalNum temp;
        temp.deNum = this->deNum * a.deNum;
        temp.num = this->num * a.deNum + this->deNum*a.num;
        temp.cut();
        return temp;
    }

};

int main()
{
    rationalNum a, b;
    cin >> a.num >> a.deNum >> b.num >> b.deNum;
    cout << a+b;
    return 0;
}
```

Для представлення полів шахової дошки визначити відповідні типи даних й написати логічну функцію хід ферзя (P, S), що перевіряє можливість переходу ферзя за один крок з поля P шахової дошки до поля S.

```
#include <iostream>
#include <cstring>

using namespace std;

struct Tpoint
{
    int x;
    int y;

    friend istream & operator >>(istream &is, Tpoint &p)
    {
        is >> p.x >> p.y;
    }

    friend ostream & operator <<(ostream &os, Tpoint &p)
    {
        os << p.x << " " << p.y << "\n";
    }
};

bool  checkMove(Tpoint Queen, Tpoint movePoint)
{
    if(Queen.x == movePoint.x || Queen.y == movePoint.y) return true;
    if(abs(Queen.x - movePoint.x) == abs(Queen.y - movePoint.y)) return
true;
    return false;
}
int main()
{
    Tpoint  Queen, movePoint;
    cin >> Queen >> movePoint;

    if(checkMove(Queen, movePoint)) cout << "Yes\n";
    else cout << "No\n";
}
```

Перехід з полярних координат у декартові та навпаки.

```
#include <cmath>

using namespace std;

struct CartesianPoint
{
    double x;
    double y;
};

struct PolarPoint
{
    double radius;
    double angle;
};

// перетворення полярних у декартові
CartesianPoint polarToCartesian(const PolarPoint& polar)
{
    double x = polar.radius * cos(polar.angle);
    double y = polar.radius * sin(polar.angle);

    return {x, y};
}

// перетворення декартових у полярні
PolarPoint cartesianToPolar(const CartesianPoint& cartesian)
{
    double radius = sqrt(cartesian.x * cartesian.x + cartesian.y *
cartesian.y);
    double angle = atan2(cartesian.y, cartesian.x);
//функція, яка обчислює обернене тангенсне значення двох
аргументів. Вона приймає два аргументи - y та x

    return {radius, angle};
}

void printCartesianPoint(const CartesianPoint& point)
{
    cout << "Декартові координати: (" << point.x << ", " << point.y <<
")" << endl;
}

void printPolarPoint(const PolarPoint& point)
{
    cout << "Полярні координати: (r = " << point.radius << ", ϕ = " <<
point.angle << " рад)" << endl;
```

```
}

int main()
{
    CartesianPoint cartesianPoint;
    PolarPoint polarPoint;

    cout << "Введіть координати точки у декартових координатах (x y): ";
    cin >> cartesianPoint.x >> cartesianPoint.y;

    printCartesianPoint(cartesianPoint);
    polarPoint = cartesianToPolar(cartesianPoint);
    printPolarPoint(polarPoint);

    cout << "Введіть координати точки у полярних координатах (r θ в
радіанах): ";
    cin >> polarPoint.radius >> polarPoint.angle;

    printPolarPoint(polarPoint);
    cartesianPoint = polarToCartesian(polarPoint);
    printCartesianPoint(cartesianPoint);

    return 0;
}
```

== Файли ==

Прибрати останній рядок

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

void deleteLastLine(const char *path);

int main() {
    char path[] = "t2.txt";
    deleteLastLine(path);
}

void deleteLastLine(const char *path){
    FILE *inFile = fopen(path, "r");
    FILE *outFile = fopen("deleteLastLineOutput.txt", "w");
    if(inFile == NULL){
        cout << "Couldn't open the file.\n";
        return;
    }

    char line[2048];
    line[0] = '\0';
    while (feof(inFile) == 0){
        int len = strlen(line);
        if(len > 0){
            if(ftell(outFile) != 0) fputc('\n', outFile);
            for (int i = 0; i < len-1; ++i) {
                fputc(line[i], outFile);
            }
        }
        fgets(line, 2048, inFile);
    }

    fclose(inFile);
    fclose(outFile);
    remove(path);
    cout << rename("deleteLastLineOutput.txt", path);
}
```

Прибрати порожні рядки

```
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

void deleteEmptyLines(const char *path);

int main() {
    char path[] = "t2.txt";
    deleteEmptyLines(path);
}

void deleteEmptyLines(const char *path){
    FILE *inFile = fopen(path, "r");
    FILE *outFile = fopen("deleteEmptyLines.txt", "w");
    if(inFile == NULL){
        cout << "Couldn't open the file.\n";
        return;
    }

    bool newLine = false;
    int c;
    while ((c=fgetc(inFile)) != EOF){
        if (c == '\n') {
            newLine = true;
            continue;
        }
        if(newLine) {
            fputc('\n', outFile);
            newLine = false;
        }
        fputc(c, outFile);
    }

    fclose(inFile);
    fclose(outFile);
    remove(path);
    rename("deleteEmptyLines.txt", path);
}
```

АБО

```cpp
#include <iostream>
#include <cstring>
#include <fstream>

using namespace std;

int main()
{
    ifstream read("test.txt");
    ofstream out("out.txt");
    char buffer[1024];
    while(read.getline(buffer,  sizeof(buffer)))
    {
        int len = strlen(buffer);
        if(len != 0)
            out << buffer << "\n";
    }

}
```

Написати функцію, яка обчислює максимальну довжину рядків у текстовому файлі.

```cpp
#include <iostream>

using namespace std;

int theMostLen(char* path);

int main()
{
    char path[] = "denysHuilo.txt";
    int len = theMostLen(path);
    if (len == -1) cerr << "File isn`t open\n";
    else cout << len;
}

int theMostLen(char* path)
{
    FILE* file; //FILE* file = fopen(path, "r")
    errno_t err = fopen_s(&file, path, "r");

    if (err != 0)
    {
        return -1;
    }

    char buffer[1024];
    int len = 0;

    while(fgets(buffer, sizeof(buffer), file))
    {
        int tempLen = strlen(buffer)- 1;
        cout << tempLen << endl;
        if (len < tempLen)
        {
            len = tempLen;
        }
    }

    fclose(file);
    return len;
}
```

У вхідному тексті знайти цифру, що зустрічається найчастіше

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

using namespace std;

int main()
{
    int isNum[10] = {0};
    fstream read("test.txt");
    char buffer[1024];
    while(read.getline(buffer,  sizeof(buffer)))
    {
        int len = strlen(buffer);
        for(int i = 0; i < len; i++)
        {
            if(buffer[i] >= '0' && buffer[i] <= '9')
                isNum[buffer[i] - '0']++;

        }
    }
    int maxCnt = isNum[0], findNum = 0;
    for(int i = 1; i < 10; i++)
    {
        if(isNum[i] > maxCnt)
        {
            maxCnt = isNum[i];
            findNum = i;
        }
    }
    cout << findNum << " " << maxCnt << "\n";
}
```