**50 Uzdevumi**

Autors: Gabriels Rosvalds

Izmantotās vietnes: CleverCode, CodeWars

Lietotājvārds abās vietnēs: GabrielsRosvalds

Contents

[Uzdevums: Reizināšana ar 2 4](#_Toc135572261)

[Uzdevums: a burtu skaits vārdā 5](#_Toc135572262)

[Uzdevums: Cikls1 iesācējiem 6](#_Toc135572263)

[Uzdevums: Cikls2 iesācējiem 7](#_Toc135572264)

[Uzdevums: cikls3 iesācējiem 8](#_Toc135572265)

[Uzdevums: cikls4 iesācējiem 9](#_Toc135572266)

[Uzdevums: Cikls5 10](#_Toc135572267)

[Uzdevums: Cikls6 11](#_Toc135572268)

[Uzdevums: Vārds no burtiem (burti2) 12](#_Toc135572269)

[Uzdevums: Pēdējais burts vārdā (burti3) 13](#_Toc135572270)

[Uzdevums: Dotā burta skaits vārdā (burti4) 14](#_Toc135572271)

[Uzdevums: Kurš burts pirmais? (burti5) 15](#_Toc135572272)

[Uzdevums: Vārda veidošana (burti6) 16](#_Toc135572273)

[Uzdevums: Trīs Skaitļi 17](#_Toc135572274)

[Uzdevums: Summa bez lielākā un mazākā cipara 18](#_Toc135572275)

[Uzdevums: pārveide uz uppercase 19](#_Toc135572276)

[Uzdevums: Samazinas, bet aug 20](#_Toc135572277)

[Uzdevums: Pāra vai nepāra 21](#_Toc135572278)

[Uzdevums: Atgriez negatīvu 22](#_Toc135572279)

[Uzdevums: Tev vajag tikai vienu 23](#_Toc135572280)

[Uzdevums: Punkti 24](#_Toc135572281)

[Uzdevums: Grasshopper saskaitīšana 25](#_Toc135572282)

[Uzdevums: Vai mans skaitlis šeit izskatās liels? 26](#_Toc135572283)

[Uzdevums: Complementary DNA 27](#_Toc135572284)

[Uzdevums: dabūt summu 28](#_Toc135572285)

[Uzdevums: Pietiekami mazs 30](#_Toc135572286)

[Uzdevums: Dilstoša secība 31](#_Toc135572287)

[Uzdevums: Mazākais un lielākais 32](#_Toc135572288)

[Uzdevums: Sveiciens! 33](#_Toc135572289)

[Uzdevums: Izlabojiet rakstzīmju atpazīšanas programmatūras kļūdas 34](#_Toc135572290)

[Uzdevums: Reizināšana 35](#_Toc135572291)

[Uzdevums: Apgriezti vārdi 36](#_Toc135572292)

[Uzdevums: Pretējais skaitlis 37](#_Toc135572293)

[Uzdevums: Testēšana 1 2 3 38](#_Toc135572294)

[Uzdevums : Cilvēki autobusā 39](#_Toc135572295)

[Uzdevums: Atrast mazāko un lielāko skaitli no saraksta 40](#_Toc135572296)

[Uzdevums: Disemvowel Trolls 41](#_Toc135572297)

[Uzdevums: Vowel Count 42](#_Toc135572298)

[Uzdevums: Calculate BMI 43](#_Toc135572299)

[Uzdevums: Sum The Strings 44](#_Toc135572300)

[Uzdevums Grasshopper-Grade book 45](#_Toc135572301)

[Uzdevums: Fibonači virknes elements (fibonaci) 46](#_Toc135572302)

[Uzdevums: Tomāti 47](#_Toc135572303)

[Uzdevums: Get the Middle Character 48](#_Toc135572304)

[Uzdevums: Odd or Even? 49](#_Toc135572305)

[Uzdevums: Tansportation on vacation 50](#_Toc135572306)

[Uzdevums: Two to One 51](#_Toc135572307)

[Uzdevums: altERnaTIng cAsE <=> ALTerNAtiNG CaSe 52](#_Toc135572308)

[Uzdevums: Sort array by string length 53](#_Toc135572309)

[Uzdevums: Find the stray number 54](#_Toc135572310)

[Uzdevums: Sum of sequence 55](#_Toc135572311)

# Uzdevums: Reizināšana ar 2

Ievaddati satur 1 skaitli N (1 <= N <= 32000)

Jaizvada skaitlis 2 \* N

**Cods:**

**#include <iostream>**

**int main() {**

**int N;**

**std::cin >> N;**

**std::cout << N\*2;**

**}**

**Testi 5/5**

# Uzdevums: a burtu skaits vārdā

Uzdevums sameklēt burtu 'a' skaitu dotajā vārdā.

**Ievaddatu** vienīgajā rindā dots vārds. Vārda maksimālais burtu skaits ir 100.

**Izvaddatu** vienīgajā rindā jāizvada viens skaitlis, kas norāda cik reižu šajā vārdā ir atkārtojies burts 'a'.

**Cods:**

**#include <iostream>**

**int main() {**

**std::string name;**

**int i;**

**int count = 0;**

**std::cin>>name;**

**for(i = 0; i<name.length(); i++){**

**if(name[i]=='a'){**

**count++;**

**}**

**}**

**std::cout<<count;**

**} Testi 5/5**

# Uzdevums: Cikls1 iesācējiem

Jāizdrukā vārds CIKLS noteiktu reižu skaitu!

**Ievaddatu** pirmajā rindā ierakstīts vesels skaitlis a ( 0<a<32000).

**Izvaddati** Stabiņā viens zem otra jāizvada vārds CIKLS a reizes.

**Cods:**

**#include <iostream>**

**int main() {**

**std::string name = "CIKLS";**

**int i;**

**int loop;**

**std::cin>>loop;**

**for(i = 0; i<loop; i++){**

**std::cout<<name<<"\n";**

**}**

**}**

**Testi 3/3**

# Uzdevums: Cikls2 iesācējiem

Ievadi skaitli un vārdu. Izvadi nolasīto vārdu tik reizes, cik liels ir nolasītais skaitlis.

**Ievaddatu** pirmajā rindā ierakstīts vesels skaitlis a (1<a<32000), bet otrajā rindā vārds.

**Izvaddati** Stabiņā viens zem otra jāizvada nolasītais vārds a reizes.

**Cods:**

**#include <iostream>**

**int main() {**

**std::string name;**

**int loop, i;**

**std::cin>>loop;**

**std::cin>>name;**

**for(i = 0; i<loop; i++){**

**std::cout<<name<<"\n";**

**}**

**} Testi 4/4**

# Uzdevums: cikls3 iesācējiem

Atrodi cik reižu meklējamais skaitlis sakrīt ar kādu no dotajiem skaitļiem!

**Ievaddatu** pirmajā rindā ierakstīts vesels skaitlis a (1<a<32000), kas norāda skaitļu skaitu un meklējamais skaitlis b (1<b<32000). Sākot ar otro rindiņu, doti vēl a skaitļi.

**Izvaddati** Jāizvada cik reižu meklējamais skaitlis b atrodas skaitļos no otrās rindiņas līdz beigām.

**Cods:**

**#include <iostream>**

**int main() {**

**int a, a1, b, i, count = 0;**

**std::cin>>a>>b;**

**for(i = 0; i<a; i++){**

**std::cin>>a1;**

**if(b==a1){**

**count++;**

**}**

**}**

**std::cout<<count;**

**}**

**Testi 3/3**

# Uzdevums: cikls4 iesācējiem

Izdrukāt visus skaitļus no 1 lidz dotajam skaitlim.

**Ievaddatu** vienīgajā rindā dots vesels skaitlis a (1<a<32000).

**Izvaddati** satur a rindas un tajās atrodas pēc kārtas visi veselie skaitļi no 1 līdz a.

**Code**:

**#include <iostream>**

**int main() {**

**int a, i;**

**std::cin>>a;**

**for(i=1; i<=a; i++){**

**std::cout<<i<<"\n";**

**}**

**}**

**Testi 5/5**

# Uzdevums: Cikls5

Jāatrod visu pēc kārtas ņemtu veselo skaitļu summa no viena skaitļa līdz otram.

**Ievaddati** vienīgajā rindā doti divi veseli skaitļi a un b (0<a<b<100).

**Izvaddati** vienīgajā rindā jāizvada visu pēc kārtas ņemtu veselu skaitļu summa no a līdz b, ieskaitot skaitļus a un b

**Cods:**

**#include <iostream>**

**int main() {**

**int a, b, sum = 0;**

**std::cin>>a>>b;**

**for(int i = a; i<=b; i++){**

**sum=sum+i;**

**}**

**std::cout<<"\n"<<sum;**

**}**

**Tests: 5/5**

# Uzdevums: Cikls6

Doti divi skaitļi. Izdrukāt mazāko skaitli tik reižu, cik liels ir lielākais skaitlis.

**Ievaddati** Doti 2 veseli skaitļi a un b (1<=a,b<=100).

**Izvadati** mazāko skaitli vienu zem otra tik reizes, cik liels ir lielākais skaitlis.

**Cods**:

**#include <iostream>**

**int main() {**

**int a, b, maxNum, minNum;**

**std::cin>>a>>b;**

**if(a<b){ maxNum = b; minNum = a;**

**}**

**else{ maxNum = a;**

**minNum = b;**

**}**

**for(int i =0; i<maxNum; i++){**

**std::cout<<"\n"<<minNum;**

**}**

**}**

**Tests 5/5**

# Uzdevums: Vārds no burtiem (burti2)

Izdrukāt vārdu, kas veidojas no citu vārdu pirmajiem burtiem.

Katrā no **ievaddatu** piecām rindām dots viens vārds.

**Izvaddatu** vienīgajā rindā ir vārds, kas sastāv no piecu doto vārdu pirmajiem burtiem.

**Cods**:

**#include <iostream>**

**int main() {**

**std::string vards, rez;**

**for(int i = 0; i<5; i++){**

**std::cin>>vards;**

**rez+=(vards[0]);**

**}**

**std::cout<<rez;**

**}**

**Tests 5/5**

# Uzdevums: Pēdējais burts vārdā (burti3)

Atrast vārda pēdējo burtu.

**Ievaddatu** vienīgajā rindā dots vārds.

**Izvaddatu** vienīgajā rindā ir dotā vārda pēdējais burts.

**Cods**:

**#include <iostream>**

**int main() {**

**std::string vards;**

**std::cin>>vards;**

**std::cout<<vards[vards.length()-1];**

**}**

**Tests** **5/5**

# Uzdevums: Dotā burta skaits vārdā (burti4)

Atrast cik reižu dotais burts atkārtojas dotajā vārdā.

**Ievaddatu** pirmajā rindā ir dots vārds, kura garums nepārsniedz 20 simbolus, bet otrajā rindā burts.

**Izvaddatu** vienīgajā rindā ir skaitlis, kas norāda cik reižu dotais burts atkārtojas dotajā vārdā.

**Cods**:

**#include <iostream>**

**int main() {**

**std::string name;**

**char a;**

**int i;**

**int count = 0;**

**std::cin>>name>>a;**

**for(i = 0; i<name.length(); i++){**

**if(name[i]==a){**

**count++;**

**}**

**}**

**std::cout<<count;**

**}**

**Tests 5/5**

# Uzdevums: Kurš burts pirmais? (burti5)

Dots vārds, kura garums nepārsniedz 20 burtus un 2 burti. Atrast un izvadīt to burtu no diviem dotajiem, kurš, lasot doto vārdu, parādās pirmais. Zināms, ka katrs no burtiem vārdā sastopams tieši vienu reizi.

**Ievaddatu** pirmajā rindā dots vārds, bet otrajā un trešajā rindā katrā pa burtam.

**Izvaddatu** vienīgajā rindā ir atrastais burts.

**Cods:**

**#include <iostream>**

**int main() {**

**std::string vards;**

**char a, b, c = 0;**

**std::cin>>vards>>a>>b;**

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

**if(vards[i]==a){ c=a; break;**

**}**

**else if(vards[i]==b){**

**c=b; break;**

**}**

**}**

**std::cout<<c;**

**}**

**Tests 5/5**

# Uzdevums: Vārda veidošana (burti6)

Doti 2 vārdi, kuru garums nepārsniedz 20 burtus. Pārbaudīt vai no pirmā vārda burtiem var izveidot otro vārdu. Zināms, ka otrais vārds ir īsāks par pirmo un tajā neviens burts neatkārtojas. Atbildes ir **Var** un  **Nevar**.

**Ievaddatu** pirmajās divās rindās doti 2 vārdi.

**Izvaddatu** vienīgajā rindā ir vārds **Var** vai **Nevar**.

**Cods**:

**#include <iostream>**

**int main() {**

**std::string vards, vards1, name;**

**std::cin>>vards>>vards1;**

**for(int i = 0; i<vards1.length(); i++){**

**for(int j = 0; j<vards.length(); j++){**

**if(vards[j]==vards1[i]){**

**name+=vards1[i];**

**vards[j]='#';**

**break;**

**}**

**}**

**}**

**if(name==vards1) std::cout<<"Var";**

**else std::cout<<"Nevar";**

**}**

**Tests 5/5**

# Uzdevums: Trīs Skaitļi

Uzrakstīt programmu, kas nosaka, vai starp ievadītajiem trijiem veseliem skaitļiem var izvēlēties divus tā, lai to summa būtu

* pozitīvs skaitlis
* nulle
* negatīvs skaitlis

Ievaddatu vienīgajā rindā dotas trīs veselu skaitļu vērtības. Zināms, ka visi skaitļi ir robežās no -2147483648 līdz 2147483647.

Izvaddatiem jāsatur tieši trīs rindas: pirmajā rindā jāizvada vārds "VAR", ja no trim ievadītajiem skaitļiem iespējams izvēlēties divus tā, lai to summa būtu pozitīvs skaitlis un "NEVAR", ja tas nav iespējams. Faila otrajā rindā jāizvada vārds "VAR", ja no trim ievadītajiem skaitļiem iespējams izvēlēties divus tā, lai to summa būtu nulle un "NEVAR", ja tas nav iespējams. Faila trešajā rindā jāizvada vārds "VAR", ja no trim ievadītajiem skaitļiem iespējams izvēlēties divus tā, lai to summa būtu negatīvs skaitlis un "NEVAR", ja tas nav iespējams.

**Cods:**

**#include <iostream>**

**int main() {**

**double a, b, c;**

**std::cin>>a>>b>>c;**

**if(a+b>0 || b+c>0 || a+c>0) std::cout<<"VAR"<<"\n";**

**else std::cout<<"NEVAR"<<"\n";**

**if(a+b==0 || b+c==0 || a+c==0)**

**std::cout<<"VAR"<<"\n";**

**else std::cout<<"NEVAR"<<"\n";**

**if(a+b<0 || b+c<0 || a+c<0) std::cout<<"VAR"<<"\n";**

**else std::cout<<"NEVAR"<<"\n";**

**}**

**Tests 12/12**

# Uzdevums: Summa bez lielākā un mazākā cipara

Summējiet visus dotā masīva skaitļus ( cq. list ), izņemot augstāko un zemāko elementu ( pēc vērtības, nevis pēc indeksa! ).

Attiecīgi augstākais vai zemākais elements ir viens elements katrā malā, pat ja ir vairāk nekā viens ar vienādu vērtību.

**Cods:**

**#include<vector>**

**using namespace std;**

**int sum(vector<int> numbers)**

**{**

**if(numbers.empty()) return 0;**

**int Min=0, Max=0;**

**int summa = 0;**

**Min = \*min\_element(numbers.begin(), numbers.end());**

**Max = \*max\_element(numbers.begin(), numbers.end());**

**cout<<Min<<" mazakais ";**

**cout<<Max<<" lielakais ";**

**for(int i = 0; i<numbers.size(); i++){**

**summa +=numbers[i];**

**}**

**if(numbers.size()<3){**

**summa = 0;**

**}**

**else summa=summa-(Max+Min);**

**return summa;**

**}**

# Uzdevums: pārveide uz uppercase

Uzrakstiet funkciju, kas ievades virkni pārvērš par lielajiem burtiem.

**Cods:**

**#include <string>**

**#include <iostream>**

**using namespace std;**

**std::string makeUpperCase (const std::string& input\_str)**

**{**

**string upper = input\_str;**

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

**upper[i] = toupper(upper[i]);**

**}**

**return upper;**

**}**

# Uzdevums: Samazinas, bet aug

Ņemot vērā veselu skaitļu masīvu, kas nav tukšs, atgriež rezultātu, reizinot vērtības kopā.

**Cods:**

**#include <vector>**

**int grow(std::vector<int> nums) {**

**int rez = 1;**

**for(int i = 0; i<nums.size(); i++){**

**rez \*=nums[i];**

**}**

**return rez;**

**}**

# Uzdevums: Pāra vai nepāra

Izveidojiet funkciju, kas ņem veselu skaitli kā argumentu un atgriež "Pāri" pāra skaitļiem vai "Nepāra" nepāra skaitļiem.

**Cods:**

**#include <string>**

**std::string even\_or\_odd(int number)**

**{**

**if(number%2==0){**

**return "Even";**

**}**

**else{**

**return "Odd";**

**}**

**}**

# Uzdevums: Atgriez negatīvu

Šajā vienkāršajā uzdevumā jums tiek dots skaitlis, un tas ir jāpadara negatīvs. Bet varbūt skaitlis jau ir negatīvs?

**Code:**

**int makeNegative(int num)**

**{**

**if(num>0){**

**num= num-(num\*2);**

**}else{**

**num = num;**

**}**

**return num;**

**}**

# Uzdevums: Tev vajag tikai vienu

Jums tiks piešķirts masīvs a un vērtība x. Viss, kas jums jādara, ir pārbaudīt, vai sniegtais masīvs satur vērtību.

Masīvā var būt skaitļi vai virknes. X var būt vai nu.

Atgriež true, ja masīvā ir vērtība, un false, ja tā nav.

**Code:**

**#include <vector>**

**#include <string>**

**bool check(const std::vector<std::string>& seq, const std::string& elem) {**

**bool res = false;**

**for(int i = 0; i<seq.size(); i++){**

**if(seq[i]==elem){**

**res = true;**

**}**

**}**

**return res;**

**}**

# Uzdevums: Punkti

Mūsu futbola komanda čempionātu ir noslēgusi.

Mūsu komandas spēļu rezultāti tiek ierakstīti String kolekcijā. Katrs mačs ir attēlots String formātā "x:y", kur x ir mūsu komandas rezultāts un y ir mūsu pretinieku rezultāts.

Piemēram: ["3:1", "2:2", "0:1", ...]

Par katru spēli punkti tiek piešķirti šādi:

ja x > y: 3 punkti (uzvara)

ja x < y: 0 punkti (zaudējums)

ja x = y: 1 punkti (neizšķirts)

Mums ir jāuzraksta funkcija, kas ņem šo kolekciju un atgriež punktu skaitu, ko mūsu komanda (x) ieguva čempionātā saskaņā ar iepriekš norādītajiem noteikumiem.

Piezīmes:

mūsu komanda čempionātā vienmēr aizvada 10 spēles

0 <= x <= 4

0 <= y <= 4

**Cods:**

**int points(const std::array<std::string, 10>& games) {**

**int p = 0;**

**for(int i = 0; i < 10; i++){**

**int a = games[i] [0] - games[i] [2];**

**if(a>0)**

**p+=3;**

**else if(a == 0)**

**p+=1;**

**}**

**return p;**

**}**

# Uzdevums: Grasshopper saskaitīšana

Uzrakstiet programmu, kas atrod katra skaitļa summēšanu no 1 līdz skaitlim. Skaitlis vienmēr būs pozitīvs vesels skaitlis, kas lielāks par 0.

Piemēram (Ievade -> Izvade):

2 -> 3 (1 + 2)

8 -> 36 (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8)

**Cods:**

**int summation(int num){**

**int sum = 0;**

**for(int i = 1; i<=num; i++){**

**sum+=i;**

**}**

**return sum;**

**}**

# Uzdevums: Vai mans skaitlis šeit izskatās liels?

Narcistiskais skaitlis (vai Ārmstronga skaitlis) ir pozitīvs skaitlis, kas ir savu ciparu summa, katrs palielināts līdz ciparu skaitam noteiktā bāzē. Šajā Kata mēs aprobežosimies ar decimāldaļu (10. bāze).

Piemēram, ņemiet 153 (3 cipari), kas ir narcistisks:

1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153

un 1652 (4 cipari), kas nav:

1^4 + 6^4 + 5^4 + 2^4 = 1 + 1296 + 625 + 16 = 1938

**#include <cmath>**

**using namespace std;**

**bool narcissistic( int value ){**

**int num = value;**

**if (num == 0) return 1;**

**int count = 0;**

**while (num != 0) {**

**num = num / 10;**

**++count;**

**}**

**int sum =0, digits[count], num1 = value;**

**for(int i = 0; i<count; i++){**

**digits[i]=num1%10;**

**num1 = num1/10;**

**}**

**for(int i = 0; i<count; i++){**

**digits[i]=pow(digits[i], count);**

**sum = sum+digits[i];**

**}**

**if(sum==value) return true;**

**else return false;**

**}**

# Uzdevums: Complementary DNA

Dezoksiribonukleīnskābe (DNS) ir ķīmiska viela, kas atrodas šūnu kodolā un nes "norādījumus" dzīvo organismu attīstībai un funkcionēšanai.

DNS virknēs simboli "A" un "T" ir viens otru papildinājumi, piemēram, "C" un "G". Jūsu funkcija saņem vienu DNS pusi (virkni, izņemot Haskell); jums ir jāatgriež otra papildinošā puse. DNS virkne nekad nav tukša vai DNS vispār nav (atkal, izņemot Haskellu).

**Cods:**

**#include <string>**

**using namespace std;**

**std::string DNAStrand(const std::string& dna)**

**{**

**string dna1 =dna;**

**for(int i = 0; i<dna1.length(); i++){**

**if(dna1[i]=='A') dna1[i]='T';**

**else if(dna1[i]=='T') dna1[i]='A';**

**else if(dna1[i]=='C') dna1[i]='G';**

**else if(dna1[i]=='G') dna1[i]='C';**

**}**

**return dna1;**

**}**

# Uzdevums: dabūt summu

Doti divi veseli skaitļi a un b, kas var būt pozitīvi vai negatīvi, atrodiet visu veselo skaitļu summu starp tiem un to ieskaitot, un atgrieziet to. Ja abi skaitļi ir vienādi, atgriež a vai b.

Piezīme: a un b nav pēc kārtas!

Piemēri (a, b) --> izvade (skaidrojums)

(1, 0) --> 1 (1 + 0 = 1)

(1, 2) --> 3 (1 + 2 = 3)

(0, 1) --> 1 (0 + 1 = 1)

(1, 1) --> 1 (1, jo abi ir vienādi)

(-1, 0) --> -1 (-1 + 0 = -1)

(-1, 2) --> 2 (-1 + 0 + 1 + 2 = 2)

**Cods:**

**int get\_sum(int a, int b)**

**{**

**int sum=0;**

**int min=0, max=0;**

**if(a>b){**

**max=a; min=b;**

**} else{**

**max=b; min=a;**

**}**

**for(int i = min; i<=max; i++){**

**sum+=i;**

**}**

**return sum;**

**}**

# Uzdevums: Pietiekami mazs

Jums tiks piešķirts masīvs un robežvērtība. Jums jāpārbauda, ​​vai visas masīva vērtības ir mazākas vai vienādas ar robežvērtību. Ja tie ir, return true. Citādi return false.

Varat pieņemt, ka visas masīvā esošās vērtības ir skaitļi.

**Cods:**

**#include <vector>**

**bool small\_enough(std::vector<int> arr, int limit) {**

**bool res = true;**

**for(int i =0; i<arr.size(); i++){**

**if(arr[i]>limit) res = false;**

**}**

**return res;**

**}**

# Uzdevums: Dilstoša secība

Tavs uzdevums ir izveidot funkciju, kas var pieņemt jebkuru nenegatīvu veselu skaitli kā argumentu un atgriezt to ar cipariem dilstošā secībā. Būtībā pārkārtojiet ciparus, lai izveidotu pēc iespējas lielāku skaitli.

Piemēri:

Ievade: 42145 Izvade: 54421

Ievade: 145263 Izvade: 654321

Ievade: 123456789 Izvade: 987654321

**Cods:**

**#include <cinttypes>**

**uint64\_t descendingOrder(uint64\_t a)**

**{**

**uint64\_t mas[10] = { 0 };**

**while(a!=0){**

**mas[a%10]++;**

**a = a/10;**

**}**

**uint64\_t n=0;**

**for(int i=9; i>=0; i--){**

**while(mas[i]>0){**

**n = n\*10 + i;**

**mas[i]--;**

**}**

**}**

**return n;**

**}**

# Uzdevums: Mazākais un lielākais

Šajā mazajā uzdevumā jums tiek dota ar atstarpi atdalītu skaitļu virkne, un jums ir jāatgriež lielākais un mazākais skaitlis.

highAndLow("1 2 3 4 5"); // return "5 1"

highAndLow("1 2 -3 4 5"); // return "5 -3"

highAndLow("1 9 3 4 -5"); // return "9 -5"

**Cods:**

**#include <string>**

**std::string highAndLow(const std::string& numbers){**

**std::stringstream ss(numbers);**

**int low, high, temp;**

**ss >> temp;**

**low = high = temp;**

**while(ss >> temp) {**

**if (temp < low) low = temp;**

**if (temp > high) high = temp;**

**}**

**return std::to\_string(high) + " " + std::to\_string(low);**

**}**

# Uzdevums: Sveiciens!

Izveidojiet funkciju, kas atgriezīs sveiciena paziņojumu, kurā tiek izmantota ievade; jūsu programmai vajadzētu atgriezties,

"Hello, <name> how are you doing today?".

**Cods:**

**#include <string>**

**std::string greet(const std::string& n){**

**return "Hello, "+n+" how are you doing today?";**

**}**

# Uzdevums: Izlabojiet rakstzīmju atpazīšanas programmatūras kļūdas

Rakstzīmju atpazīšanas programmatūra tiek plaši izmantota drukāto tekstu digitalizēšanai. Tādējādi tekstus var rediģēt, meklēt un saglabāt datorā.

Kad dokumenti (īpaši diezgan veci, kas rakstīti ar rakstāmmašīnu), tiek digitalizēti, rakstzīmju atpazīšanas programmatūra bieži pieļauj kļūdas.

Jūsu uzdevums ir izlabot kļūdas digitalizētajā tekstā. Jums ir jārisina tikai šādas kļūdas:

S tiek nepareizi interpretēts kā 5

O ir nepareizi interpretēts kā 0

Mani nepareizi interpretē kā 1

Pārbaudes gadījumi satur skaitļus tikai kļūdas pēc.

**Cods:**

**#include <string>**

**std::string correct(std::string str){**

**for(auto &i: str){**

**if(i == '5') i = 'S';**

**else if(i == '1') i = 'I';**

**else if(i == '0') i = 'O';**

**}**

**return str;**

**}**

# Uzdevums: Reizināšana

Šī kata ir par dotā skaitļa reizināšanu ar astoņi, ja tas ir pāra skaitlis, un ar deviņi pretējā gadījumā.

**Cods:**

**int simpleMultiplication(int a){**

**if(a%2==0){**

**return a\*8;**

**}**

**else{**

**return a\*9;**

**}**

**}**

# Uzdevums: Apgriezti vārdi

Pabeidziet funkciju, kas pieņem virknes parametru, un apgriež katru vārdu virknē. Visas atstarpes virknē ir jāsaglabā.

"Šis ir piemērs!" ==> "sihT si na !elpmaxe"

"double spaces" ==> "elbuod secaps"

**Cods:**

**#include <string>**

**#include <algorithm>**

**std::string reverse\_words(std::string str)**

**{**

**int i = 0, j = 0;**

**while (j <= str.size()) {**

**if (j == str.size()|| str[j] == ' ' ) {**

**std::reverse(str.begin() + i, str.begin() + j);**

**i = j+1;**

**}**

**j++;**

**}**

**return str;**

**}**

# Uzdevums: Pretējais skaitlis

Ļoti vienkārši, ņemot vērā veselu skaitli, atrodiet tā pretstatu.

1: -1

14: -14

-34: 34

**Cods:**

**int opposite(int number)**

**{**

**if(number<0){**

**return abs(number);**

**}**

**else{**

**return number \* -1;**

**}**

**}**

# Uzdevums: Testēšana 1 2 3

Jūsu komanda raksta jaunu izdomātu teksta redaktoru, un jums ir uzdots ieviest rindu numerāciju.

Uzrakstiet funkciju, kas ņem virkņu sarakstu un atgriež katru rindiņu, pievienojot pareizo skaitli.

Numerācija sākas ar 1. Formāts ir n: virkne. Ievērojiet kolu un atstarpi starp tām.

Piemēri: (Ievade --> Izvade)

[] --> []

["a", "b", "c"] --> ["1: a", "2: b", "3: c"]

**Cods:**

**#include <string>**

**#include <vector>**

**using namespace std;**

**vector<string> number(const vector<string> &lines)**

**{**

**vector<string> mas = lines;**

**for (int i = 1; i < mas.size()+1; i++)**

**mas[i-1].insert(0, to\_string(i)+": ");**

**return mas;**

**}**

# Uzdevums : Cilvēki autobusā

Pilsētā kursē autobuss, kas ņem un izlaiž dažus cilvēkus katrā autobusa pieturā.

Jums tiek nodrošināts veselu skaitļu pāru saraksts (vai masīvs). Katra pāra elementi apzīmē cilvēku skaitu, kas iekāpj autobusā (pirmais vienums) un cilvēku skaitu, kas izkāpj no autobusa (otrais vienums) autobusu pieturā.

Jūsu uzdevums ir atgriezt to cilvēku skaitu, kuri joprojām atrodas autobusā pēc pēdējās autobusa pieturas (pēc pēdējā masīva). Lai gan tā ir pēdējā autobusa pietura, autobuss varētu nebūt tukšs un daži cilvēki vēl varētu būt iekšā, iespējams, ka viņi tur guļ :D

Apskatiet testa gadījumus.

Lūdzu, ņemiet vērā, ka testa gadījumi nodrošina, ka cilvēku skaits autobusā vienmēr ir >= 0. Tātad atgrieztais veselais skaitlis nevar būt negatīvs.

Otrā vērtība masīva pirmajā pārī ir 0, jo autobuss pirmajā pieturā ir tukšs.

**Cods:**

**#include <utility>**

**#include <vector>**

**unsigned int number(const std::vector<std::pair<int, int>>& busStops){**

**int x = 0;**

**for (int i = 0; i < busStops.size(); ++i)**

**x += busStops[i].first - busStops[i].second;**

**return x;**

**}**

# Uzdevums: Atrast mazāko un lielāko skaitli no saraksta

Jūsu uzdevums ir izveidot divas funkcijas (maksimums un min. vai maksimums un minimums utt., atkarībā no valodas), kas saņem veselu skaitļu sarakstu kā ievadi un attiecīgi atgriež lielāko un mazāko skaitli šajā sarakstā.

Piemēri (ievade —> izvade)

\* [4,6,2,1,9,63,-134,566] -> max = 566, min = -134

\* [-52, 56, 30, 29, -54, 0, -110] -> min = -110, maks. = 56

\* [42, 54, 65, 87, 0] -> min = 0, maks = 87

\* [5] -> min = 5, maks = 5

**Code:**

**#include <vector>**

**#include <algorithm>**

**using namespace std;**

**int min(vector<int> list){**

**std::sort(list.begin(), list.end());**

**return list[0];**

**}**

**int max(vector<int> list){**

**std::sort(list.begin(), list.end());**

**int res = list.back();**

**return res;**

**}**

# Uzdevums: Disemvowel Trolls

Troļļi uzbrūk tavai komentāru sadaļai!

Izplatīts veids, kā risināt šo situāciju, ir noņemt visus patskaņus no troļļu komentāriem, neitralizējot draudus.

Jūsu uzdevums ir uzrakstīt funkciju, kas aizņem virkni, un atgriezt jaunu virkni, kurā ir noņemti visi patskaņi.

Piemēram, virkne "This website is for losers LOL!!" kļūtu par "Ths wbst s fr lsrs LL!".

**Cods:**

**# include <string>**

**#include<algorithm>**

**#include<iostream>**

**using namespace std;**

**std::string disemvowel(const std::string& str) {**

**string str1 = str;**

**char mas[] = {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'};**

**cout<<str1;**

**for(int i = 0; i<str1.length(); i++){**

**for(int j = 0; j<sizeof(mas); j++){**

**str1.erase(remove(str1.begin(), str1.end(), mas[j]),str1.end());**

**}**

**}**

**cout<<str1;**

**return str1;**

**}**

# Uzdevums: Vowel Count

Return the number of vowels in the given string.

a, e, i, o, u as vowels for this Kata (but not y).

**Cods:**

**#include <string>**

**using namespace std;**

**int getCount(const string& inputStr){**

**int num\_vowels = 0;**

**char mas[] = {'A', 'a', 'E', 'e', 'I', 'i', 'O', 'o', 'U', 'u'};**

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

**for(int j = 0; j<sizeof(mas); j++){**

**if(inputStr[i]==mas[j]) num\_vowels++;**

**}**

**}**

**return num\_vowels;**

**}**

# Uzdevums: Calculate BMI

Ierakstiet funkciju bmi, kas aprēķina ķermeņa masas indeksu (bmi = svars / augums kvadrātā).

if bmi <= 18.5 return "Underweight"

if bmi <= 25.0 return "Normal"

if bmi <= 30.0 return "Overweight"

if bmi > 30 return "Obese"

**Cods:**

**#include<iostream>**

**#include<algorithm>**

**#include <math.h>**

**using namespace std;**

**std::string bmi(double w, double h)**

**{**

**float bmi = w/(h\*h);**

**if (bmi <= 18.5) return "Underweight";**

**else if(bmi <= 25.0) return "Normal";**

**else if(bmi <= 30.0) return "Overweight";**

**else if (bmi > 30.0) return "Obese";**

**}**

# Uzdevums: Sum The Strings

Izveidojiet funkciju, kas ņem 2 veselus skaitļus String formā kā ievadi un izvada summu (arī kā String):

Piemērs: (input1, Input2 --> Output):

"4", "5" --> "9"

"34", "5" --> "39"

"", "" --> "0"

"2", "" --> "2"

"-5", "3" --> "-2"

**Cods:**

**#include <string>**

**using namespace std;**

**std::string sum\_str(const std::string& a, const std::string& b)**

**{**

**string sum = " ";**

**if (a.empty() && b.empty()) return "0";**

**if (a.empty()) return b;**

**if (b.empty()) return a;**

**else**

**{**

**sum = to\_string(stoi(a)+stoi(b));**

**return sum;**

**}**

**}**

# Uzdevums Grasshopper-Grade book

Pabeidziet funkciju, lai tā atrastu vidējo no trīs tai piešķirtajiem punktiem un atgrieztu ar šo atzīmi saistīto burta vērtību.

Numerical Score Letter Grade

90 <= score <= 100 'A'

80 <= score < 90 'B'

70 <= score < 80 'C'

60 <= score < 70 'D'

0 <= score < 60 'F'

**Cods:**

**char getGrade(int a, int b, int c) {**

**int avg = (a+b+c)/3;**

**if(avg>=0 && avg<60) return 'F';**

**else if(avg>=60 && avg<70) return 'D';**

**else if(avg>=70 && avg<80) return 'C';**

**else if(avg>=80 && avg<90) return 'B';**

**else if(avg>=90 && avg<=100) return 'A';**

**}**

# Uzdevums: Fibonači virknes elements (fibonaci)

Definīcija

Matemātikā par Fibonači skaitļiem sauc virknes 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 … elementus. Tās pirmie divi locekļi ir vienādi ar 1, bet katru nākamo locekli iegūst saskaitot divus iepriekšējos. Uzraksti programmu, kas atrod šīs virknes n-to elementu.

Ievaddatu raksturojums

Ievadatu pirmajā rindā ir dots virknes elementa kārtas numurs n (1<=n<=45).

Izvaddatu raksturojums

Izvaddatu pirmajā rindā jāizvada atbilstošais Fibonači virknes skaitlis.

**Cods:**

**#include <iostream>**

**int main() {**

**unsigned long i, a, b1 = 0, b2 = 1, bK;**

**std::cin>>a;**

**for (i = 1; i<=a; i++) {**

**bK = b1 + b2;**

**b1 = b2;**

**b2 = bK;**

**}**

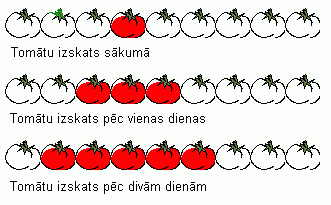
**std::cout<<b1<<" ";**

**}**

# Uzdevums: Tomāti

**Definīcija**

Erna nodarbojas ar tomātu audzēšanu un ir salikusi uz palodzes rindā n jaunās ražas tomātus. Viens no tomātiem ir sarkans, bet pārējie - zaļi. Ir novērots, ka katrs zaļais tomāts, kas stāvējis blakus sarkanajam, nākošajā dienā kļūst sarkans. Citi tomāti savu krāsu nemaina. Piemēram, ja pavisam ir deviņi tomāti un ceturtais no kreisās puses ir sarkans, tad pēc divām dienām būs pieci sarkani tomāti.



Uzrakstiet programmu, kas dotam tomātu skaitam, sarkanā tomāta novietojumam un dienu skaitam nosaka, cik sarkano tomātu būs pēc norādīto dienu skaita! Tomātus nogatavošanās laikā pārkārtot nedrīkst.

**Cods:**

**#include <iostream>**

**int main() {**

**int n, s, d, tomati= 1;**

**std::cin>>n>>s>>d;**

**int kp = s-1;**

**int lp = n-s;**

**tomati =(d>kp) ? tomati = tomati+kp:**

**tomati = tomati+d;**

**tomati = (d>lp) ? tomati =**

**tomati+lp:tomati = tomati +d;**

**std::cout<<tomati;**

**}**

# Uzdevums: Get the Middle Character

Jums tiks dots vārds. Jūsu uzdevums ir atgriezt vārda vidējo rakstzīmi. Ja vārda garums ir nepāra, atgrieziet vidējo rakstzīmi. Ja vārda garums ir pāra garums, atgrieziet vidējās 2 rakstzīmes.

#Piemēri:

Kata.getMiddle("test") should return "es"

Kata.getMiddle("testing") should return "t"

Kata.getMiddle("middle") should return "dd"

Kata.getMiddle("A") should return "A"

**Cods:**

**using namespace std;**

**std::string get\_middle(std::string input)**

**{**

**int length;**

**length = input.length();**

**int mid;**

**mid = length/2;**

**string middle;**

**string middle2;**

**if(input.length()%2==0){**

**middle = input[mid-1];**

**middle2 = input[mid];**

**return middle+middle2;**

**}**

**else{**

**middle = input[mid];**

**return middle;**

**}**

**}**

# Uzdevums: Odd or Even?

Dots veselu skaitļu saraksts, nosakiet, vai tā elementu summa ir nepāra vai pāra.

Norādiet savu atbildi kā virkni, kas atbilst "nepāra" vai "pāra".

Piemēri:

Input: [0]

Output: "even"

Input: [0, 1, 4]

Output: "odd"

Input: [0, -1, -5]

Output: "even"

**Cods:**

**#include <string>**

**#include <vector>**

**std::string odd\_or\_even(const std::vector<int> &arr)**

**{**

**int sum = 0;**

**for(int i = 0; i<arr.size(); i++){**

**sum += arr[i];**

**}**

**if(sum%2==0) return "even";**

**else return "odd";**

**}**

# Uzdevums: Tansportation on vacation

Pēc smaga ceturkšņa birojā jūs nolemjat mazliet atpūsties atvaļinājumā. Tātad jūs rezervēsiet lidojumu sev un savai draudzenei un mēģināsiet atstāt visu nekārtību aiz sevis.

Lai varētu pārvietoties atvaļinājumā, jums būs nepieciešama nomas automašīna. Auto nomas menedžeris jums izsaka dažus labus piedāvājumus.

Katra diena, kad jūs nomājat automašīnu, maksā 40 USD. Ja iznomājat automašīnu uz 7 vai vairāk dienām, jūs saņemat 50 $ atlaidi no kopējās summas. Alternatīvi, ja iznomājat automašīnu uz 3 vai vairāk dienām, jūs saņemat 20 ASV dolārus no kopējās summas.

Uzrakstiet kodu, kas parāda kopējo summu dažādām dienām(d).

**Cods:**

**int rental\_car\_cost(int d){**

**if(d>=3&&d<7){**

**return (40\*d)-20;**

**}**

**else if(d>=7){**

**return (40\*d)-50;**

**}**

**else return 40\*d;**

**}**

# Uzdevums: Two to One

Paņemiet 2 string s1 un s2, kas ietver tikai burtus no a līdz z. Atgrieziet jaunu sakārtotu virkni, visgarāko iespējamo, kas satur atšķirīgus burtus (katrs ņemts tikai vienu reizi), kas nāk no s1 vai s2.

Piemērs:

a = "xyaabbbccccdefww"

b = "xxxxyyyyabklmopq"

longest(a, b) -> "abcdefklmopqwxy"

a = "abcdefghijklmnopqrstuvwxyz"

longest(a, a) -> "abcdefghijklmnopqrstuvwxyz"

**Cods:**

**#include <algorithm>**

**using namespace std;**

**class TwoToOne**

**{**

**public:**

**static std::string longest(const std::string &s1, const std::string &s2)**

**{**

**string inOrder = s1 + s2;**

**sort(inOrder.begin(), inOrder.end());**

**auto last = unique(inOrder.begin(), inOrder.end());**

**inOrder.erase(last, inOrder.end());**

**return inOrder;**

**}**

**};**

# Uzdevums: altERnaTIng cAsE <=> ALTerNAtiNG CaSe

katrs mazais burts kļūst par lielo un katrs lielais burts kļūst par mazo. Piemēram:

"hello world".toAlternatingCase() === "HELLO WORLD"

"HELLO WORLD".toAlternatingCase() === "hello world"

"hello WORLD".toAlternatingCase() === "HELLO world"

"HeLLo WoRLD".toAlternatingCase() === "hEllO wOrld"

"12345".toAlternatingCase() === "12345" // Non-alphabetical characters are unaffected

"1a2b3c4d5e".toAlternatingCase() === "1A2B3C4D5E"

"String.prototype.toAlternatingCase".toAlternatingCase() === "sTRING.PROTOTYPE.TOaLTERNATINGcASE"

**Code:**

**#include <iostream>**

**using namespace std;**

**std::string to\_alternating\_case(const std::string& str)**

**{**

**string word = str;**

**int ln = word.length();**

**for (int i = 0; i < ln; i++) {**

**if (word[i] >= 'a' && word[i] <= 'z'){**

**word[i] = word[i] - 32;**

**}**

**else if (word[i] >= 'A' && word[i] <= 'Z'){**

**word[i] = word[i] + 32;**

**}**

**}**

**return word;**

**}**

# Uzdevums: Sort array by string length

Uzrakstiet funkciju, kas izmanto virkņu masīvu kā argumentu un atgriež sakārtotu masīvu, kurā ir tās pašas virknes, sakārtotas no īsākā uz garāko.

Masīvs sākumā:

["Telescopes", "Glasses", "Eyes", "Monocles"]

Sakārtots masīvs

["Eyes", "Glasses", "Monocles", "Telescopes"]

**Cods:**

**using namespace std;**

**class Kata**

**{**

**public:**

**static bool comparetor(const string& a, const string& b)**

**{**

**return (a.size() < b.size());**

**}**

**vector<string> sortByLength(vector<string> array)**

**{**

**sort(array.begin(), array.end(), comparetor);**

**return array;**

**}**

**};**

# Uzdevums: Find the stray number

Jums tiek dots nepāra garuma veselu skaitļu masīvs, kurā tie visi ir vienādi, izņemot vienu skaitli.

Pabeidziet metodi, kas pieņem šādu masīvu, un atgriež šo atšķirīgo skaitli.

Piemēri:

[1, 1, 2] ==> 2

[17, 17, 3, 17, 17, 17, 17] ==> 3

**Cods:**

**int stray(std::vector<int> numbers) {**

**int n = numbers.size();**

**if (n == 1)**

**return -1;**

**if (n == 2)**

**return 0;**

**if (numbers[0] == numbers[1] && numbers[0] != numbers[2])**

**return numbers[2];**

**if (numbers[0] == numbers[2] && numbers[0] != numbers[1])**

**return numbers[1];**

**if (numbers[1] == numbers[2] && numbers[0] != numbers[1])**

**return numbers[0];**

**for (int i = 3; i < n; i++){**

**if (numbers[i] != numbers[i - 1])**

**return numbers[i];**

**}**

**return -1;**

**};**

# Uzdevums: Sum of sequence

Tavs uzdevums ir uzrakstīt funkciju, kas atgriež veselu skaitļu virknes summu.

Secību nosaka 3 nenegatīvas vērtības: sākums, beigas, solis.

Ja sākuma vērtība ir lielāka par beigu, jūsu funkcijai ir jāatgriež 0. Ja beigas nav vesela soļu skaita rezultāts, nepievienojiet to summai. Skatiet tālāk 4. piemēru.

Piemēri:

2,2,2 --> 2

2,6,2 --> 12 (2 + 4 + 6)

1,5,1 --> 15 (1 + 2 + 3 + 4 + 5)

1,5,3 --> 5 (1 + 4)

**Cods:**

**int sequenceSum(int start, int end, int step)**

**{**

**int sum = 0;**

**if(start>end) return 0;**

**if(start<=end){**

**for(int i = start; i<=end; i+=step){**

**if(end<(i+step)){**

**}**

**sum+=i;**

**}**

**return sum;**

**}**

**}**