



#include <stdio.h>

int main()

{

int d1, d2;

do

{

printf ("Input number 1: ");

scanf ("%d", &d1);

printf ("Input number 2: ");

scanf ("%d", &d2);

if (d1==0 || d2==0 || d1==d2 || ((d1%d2)>3) || ((d1%d2)==0)) printf ("Incorrect data, try again.\n");

int remainder = d1%d2;

int i, fraction = d1 / d2;

if (remainder == 1)

{

for (i=0; i<d2; i++)

{

printf ("%d ", fraction);

}

printf ("%d ", remainder);

}

i = 0;

if (remainder == 2)

{

while (i < d2)

{

printf ("%d ", fraction);

i++;

}

printf ("%d ", remainder);

}

i = 0;

if (remainder == 3)

{

do

{

printf ("%d ", fraction);

i++;

} while (i != d2);

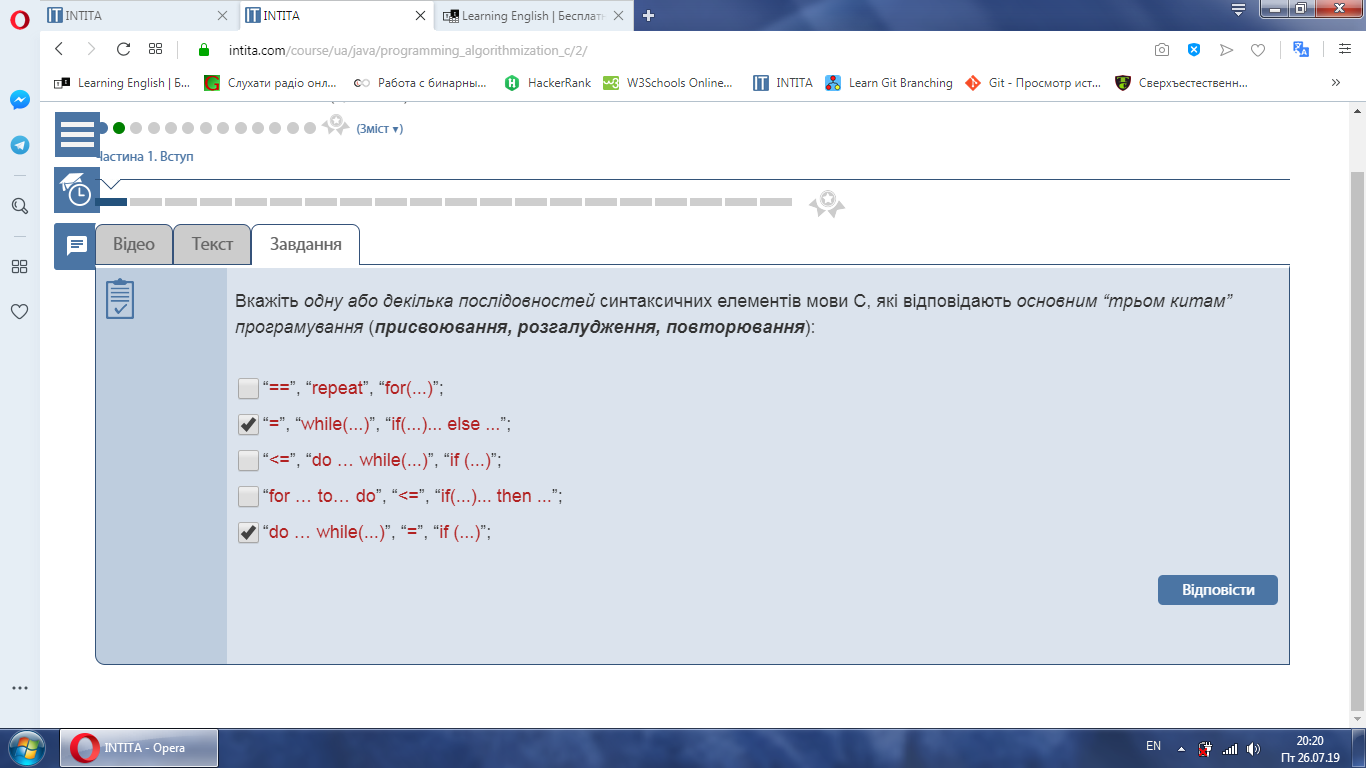
printf ("%d ", remainder);

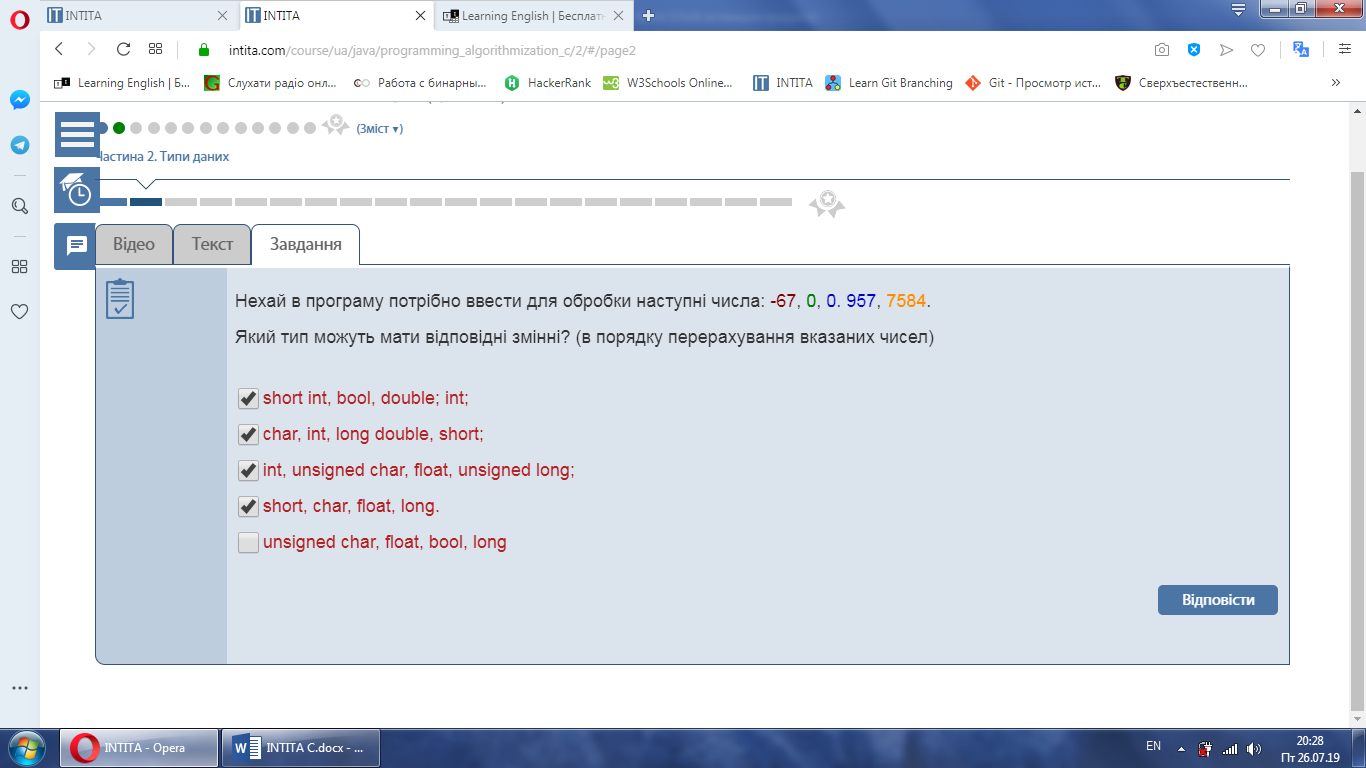
}

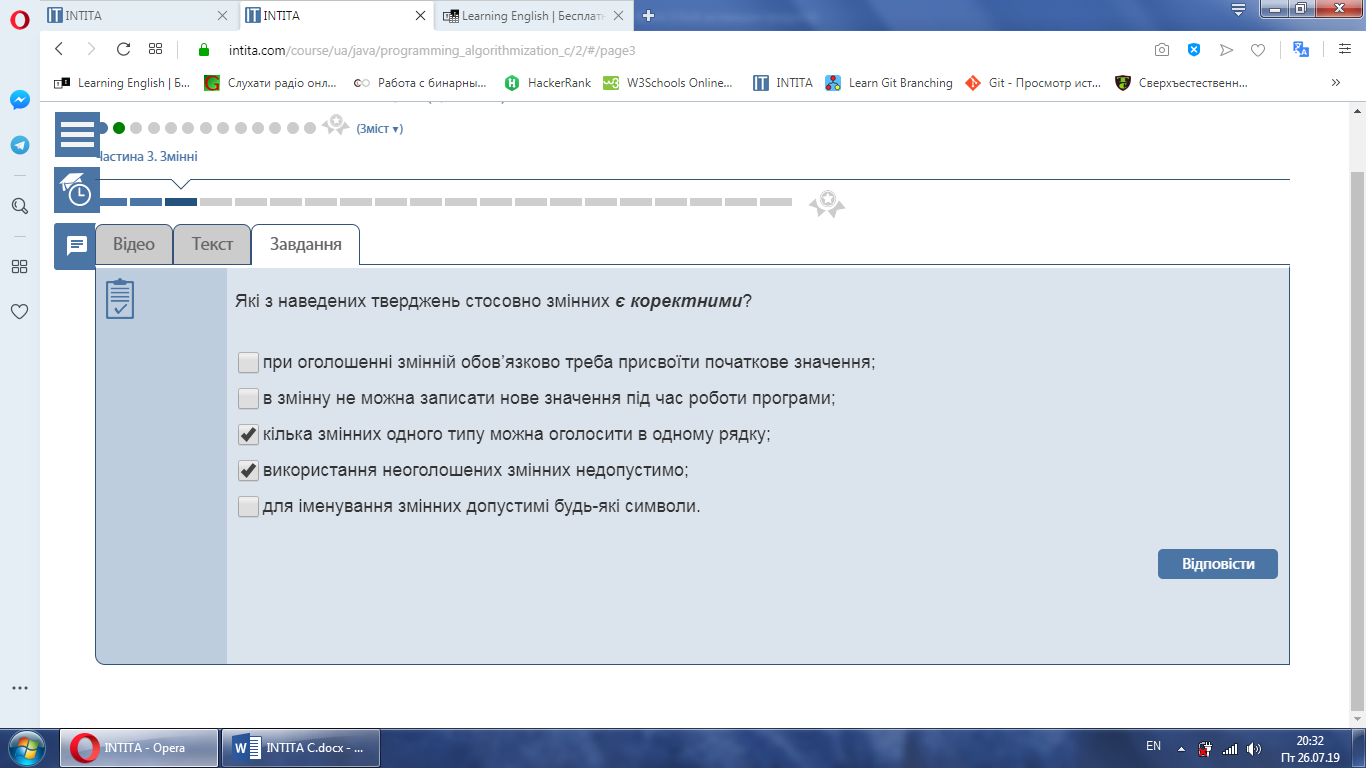
} while (d1==0 || d2==0 || d1==d2 || ((d1%d2)>3) || ((d1%d2)==0));

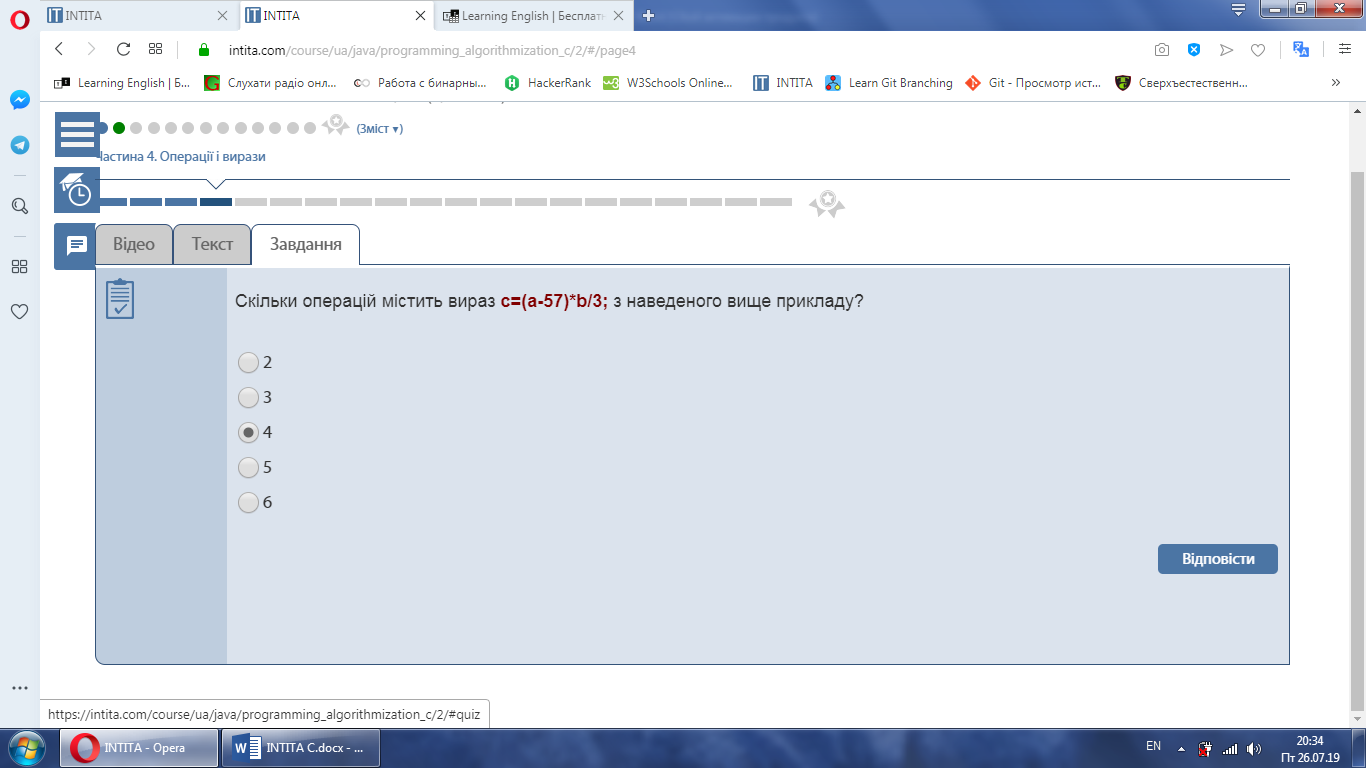
return 0;

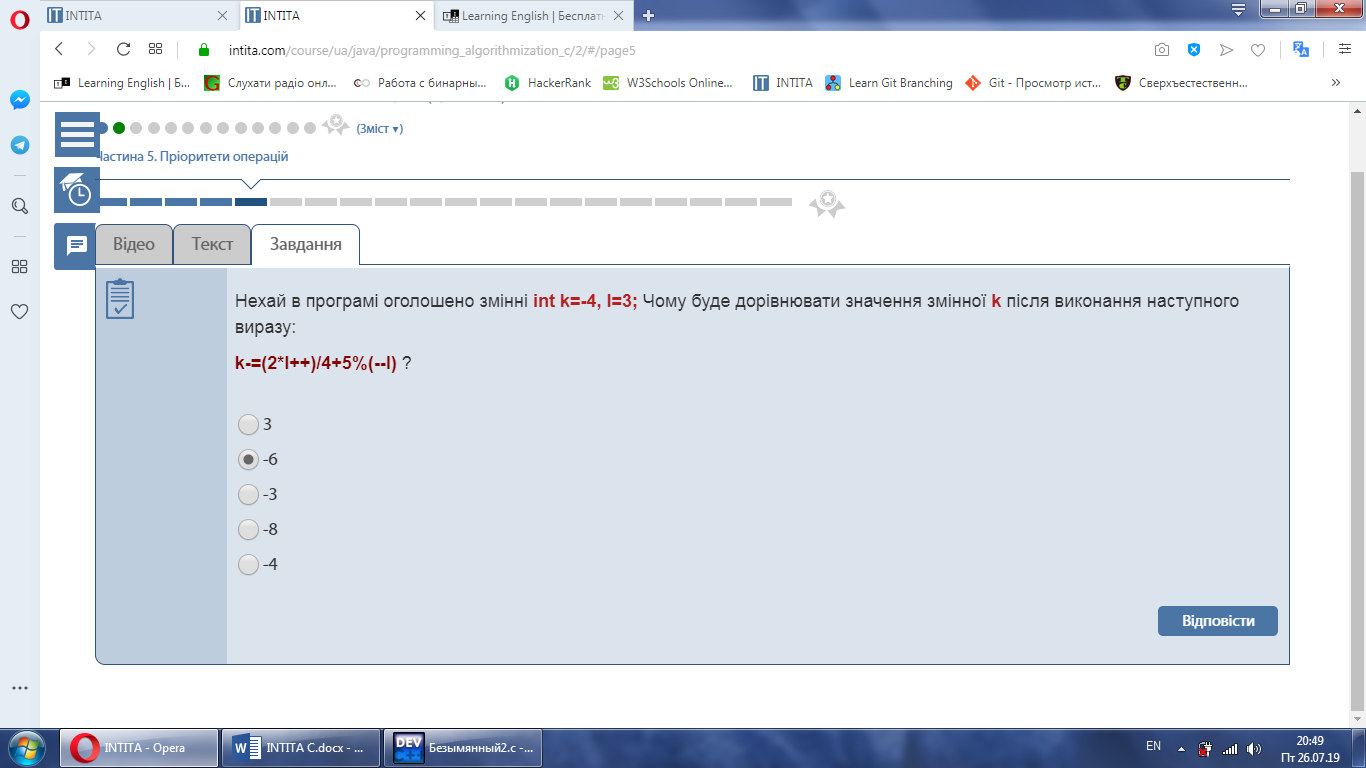
}











#include <stdio.h>

main()

{

int a=0, b=0, c;

switch (1)

{

case 1:

printf ("a = ");

scanf("%d", &a);

case 2:

printf ("b = ");

scanf("%d", &b);

}

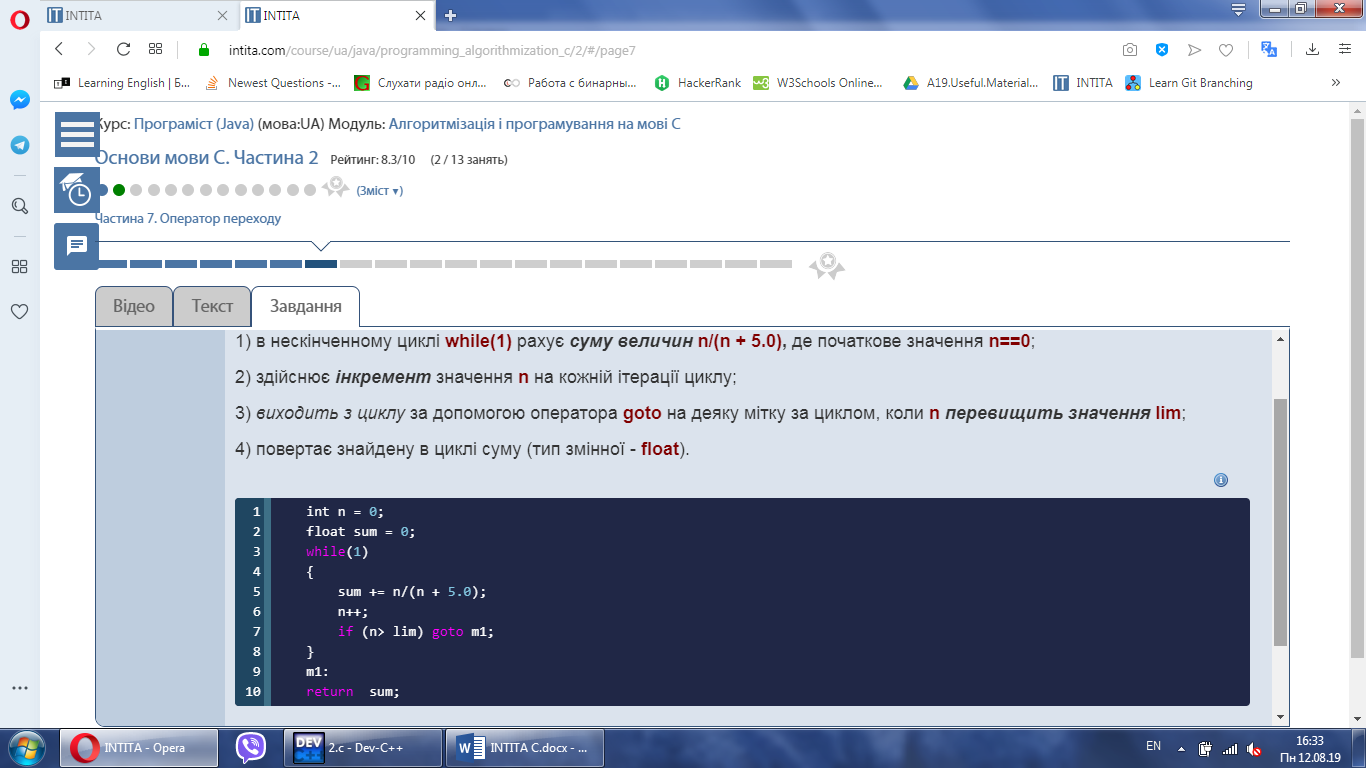
c = a + b;

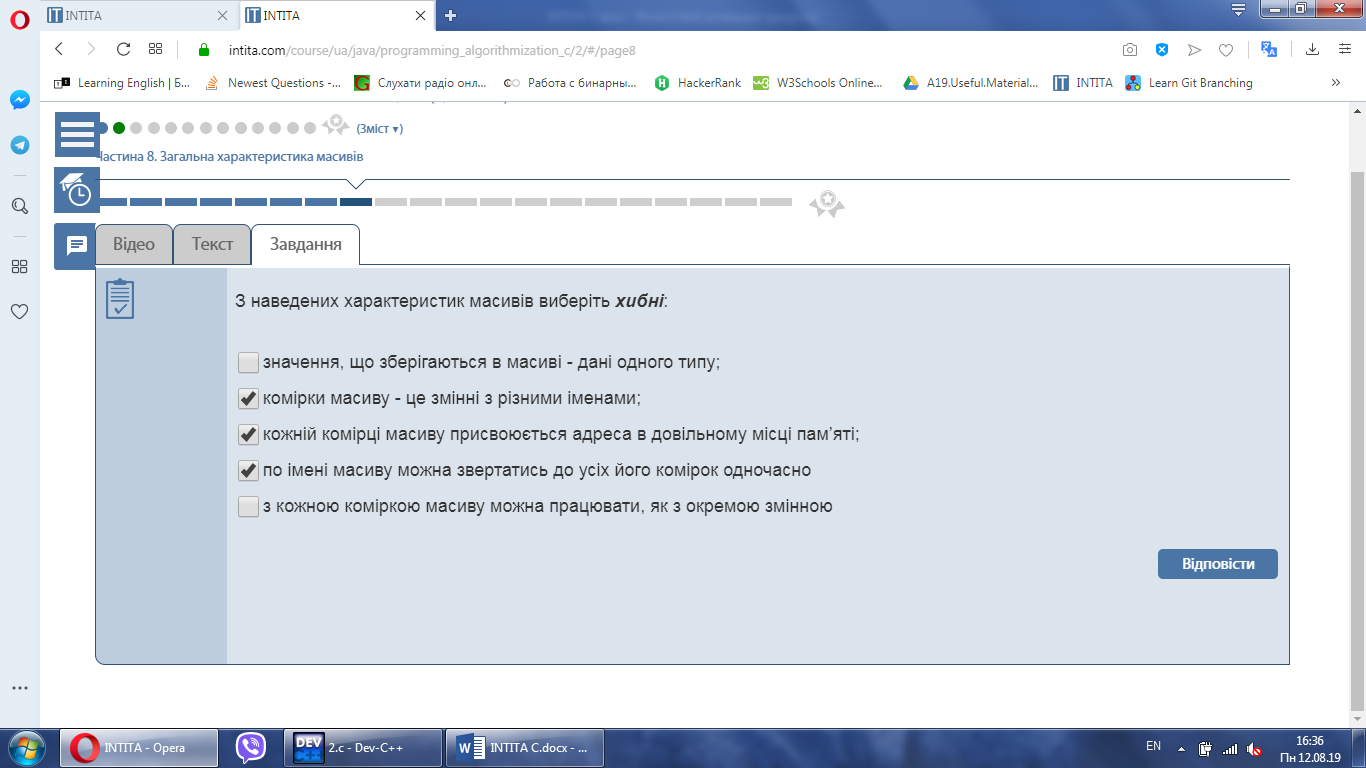
printf ( "result: %d + %d = %d \n", a, b, c );

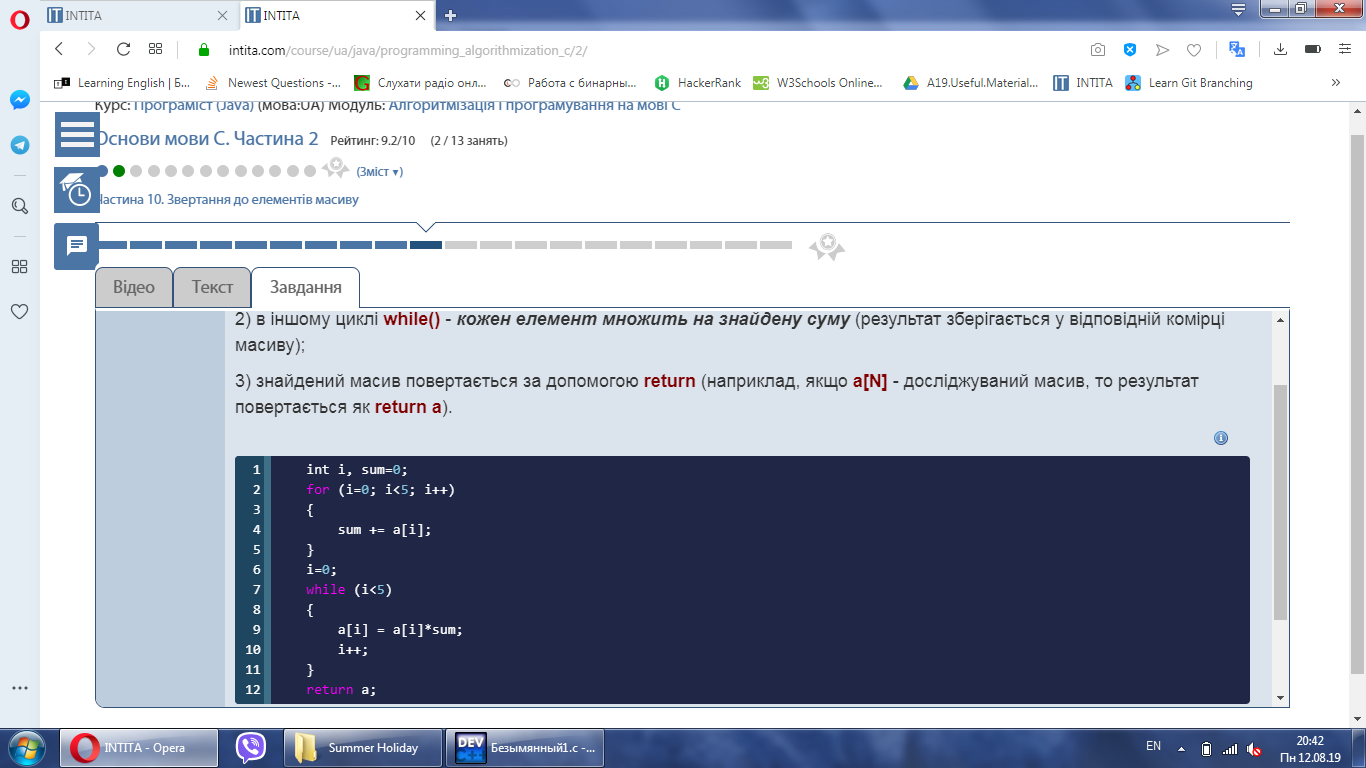
getchar();

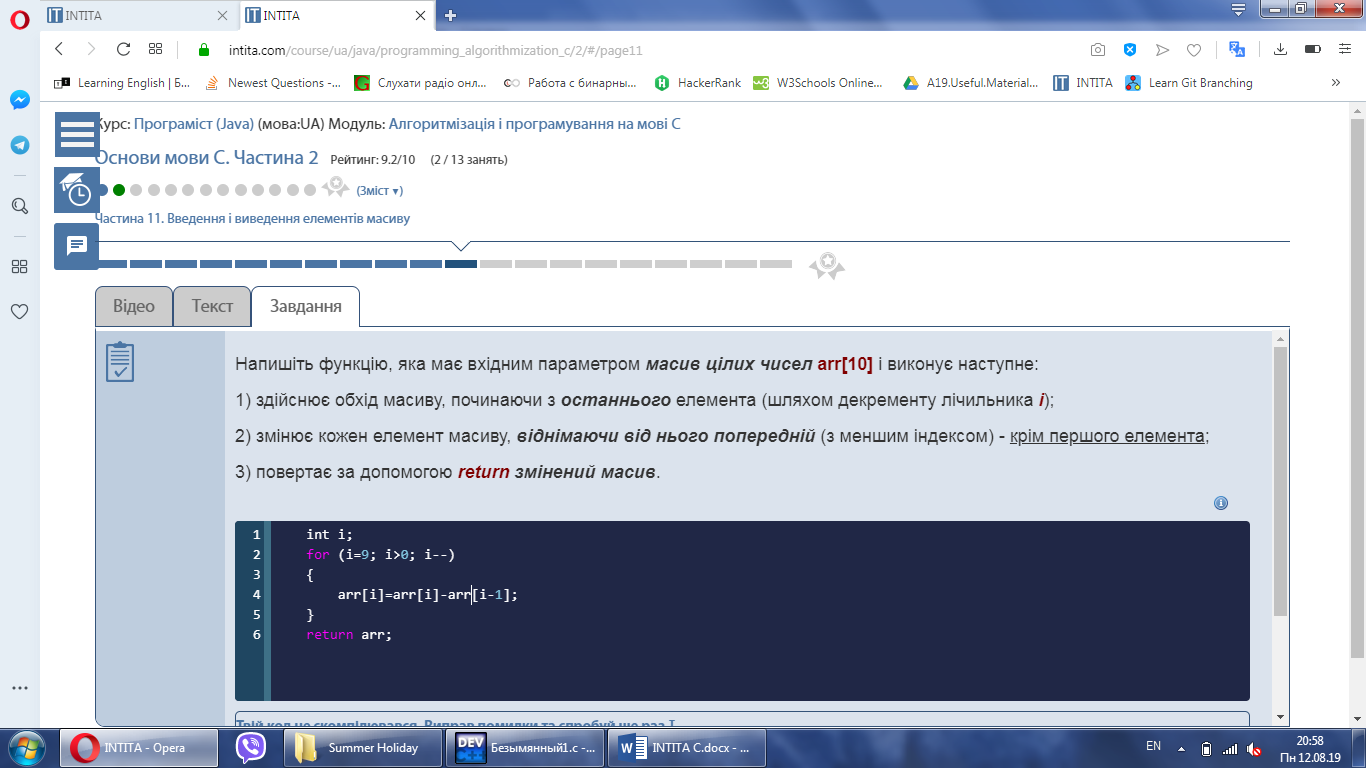
return 0;

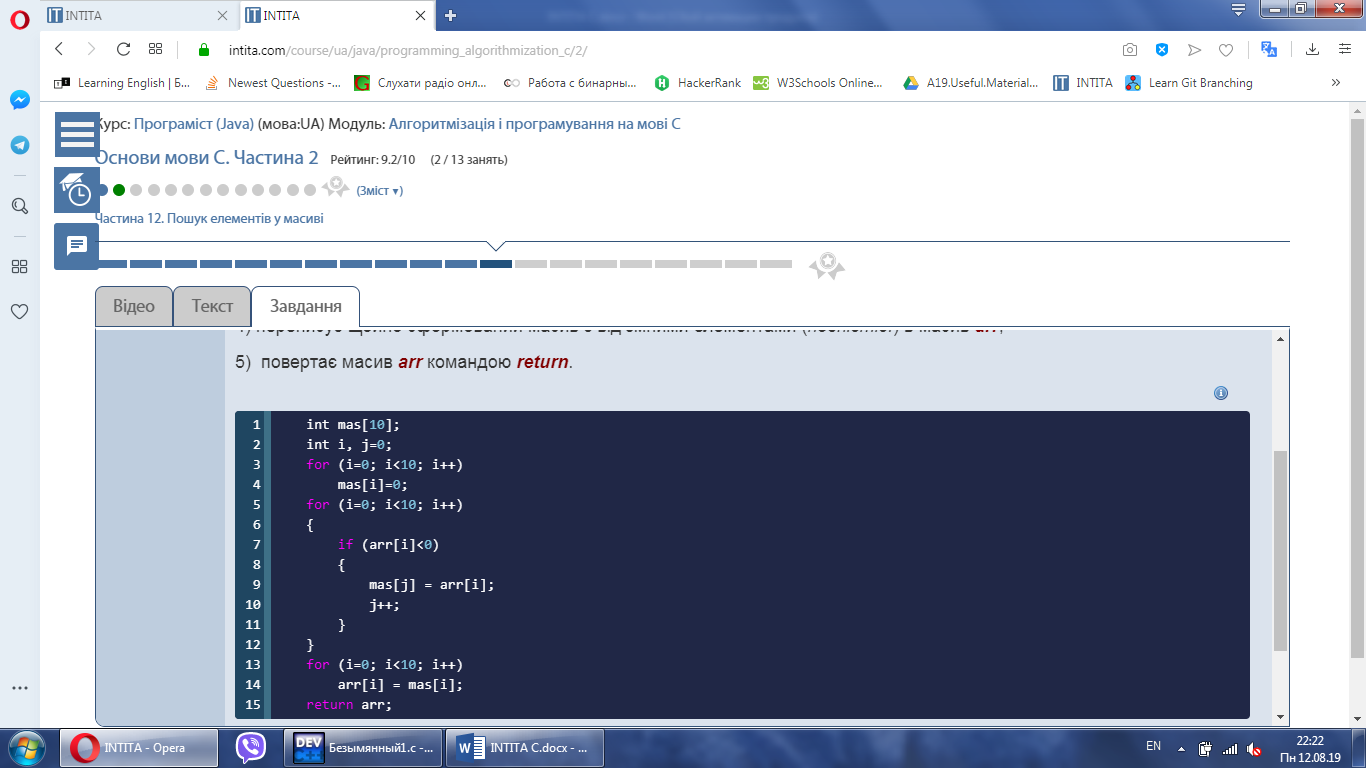
}

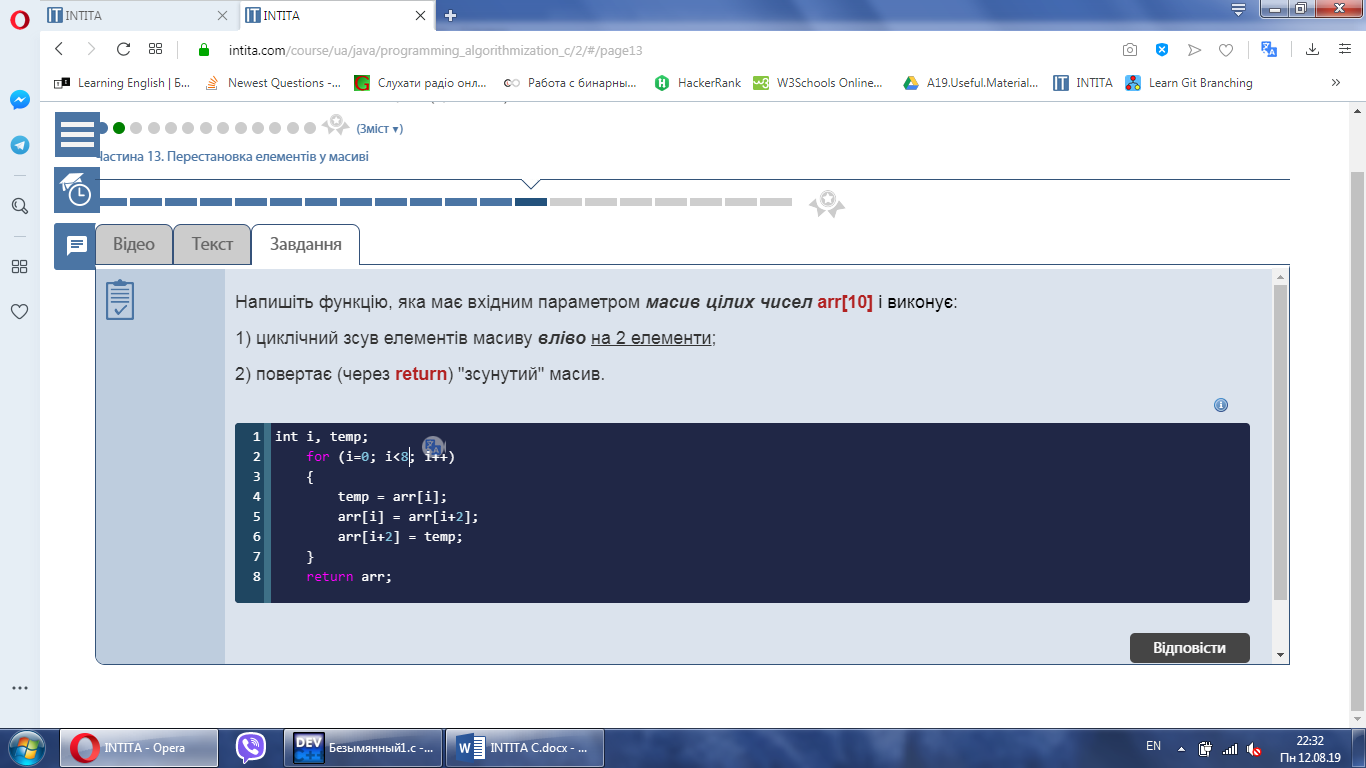


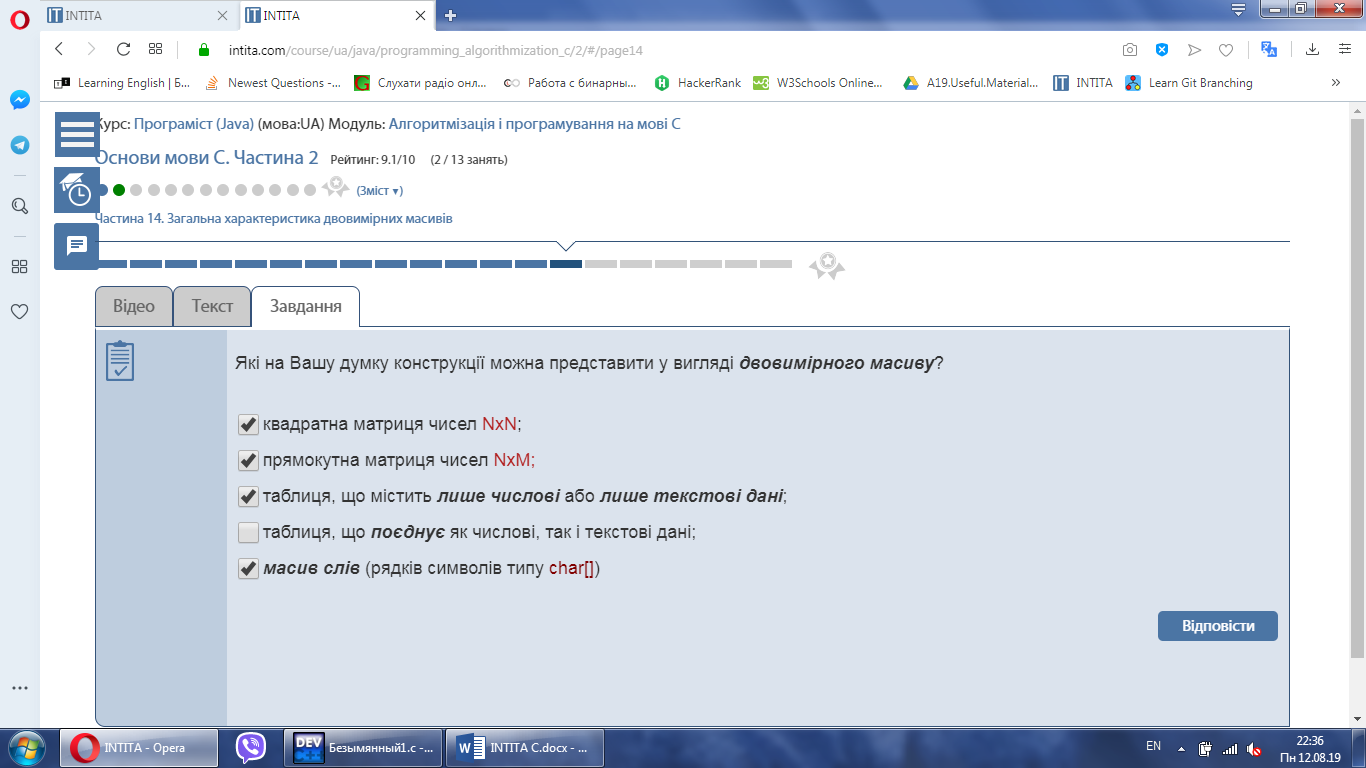


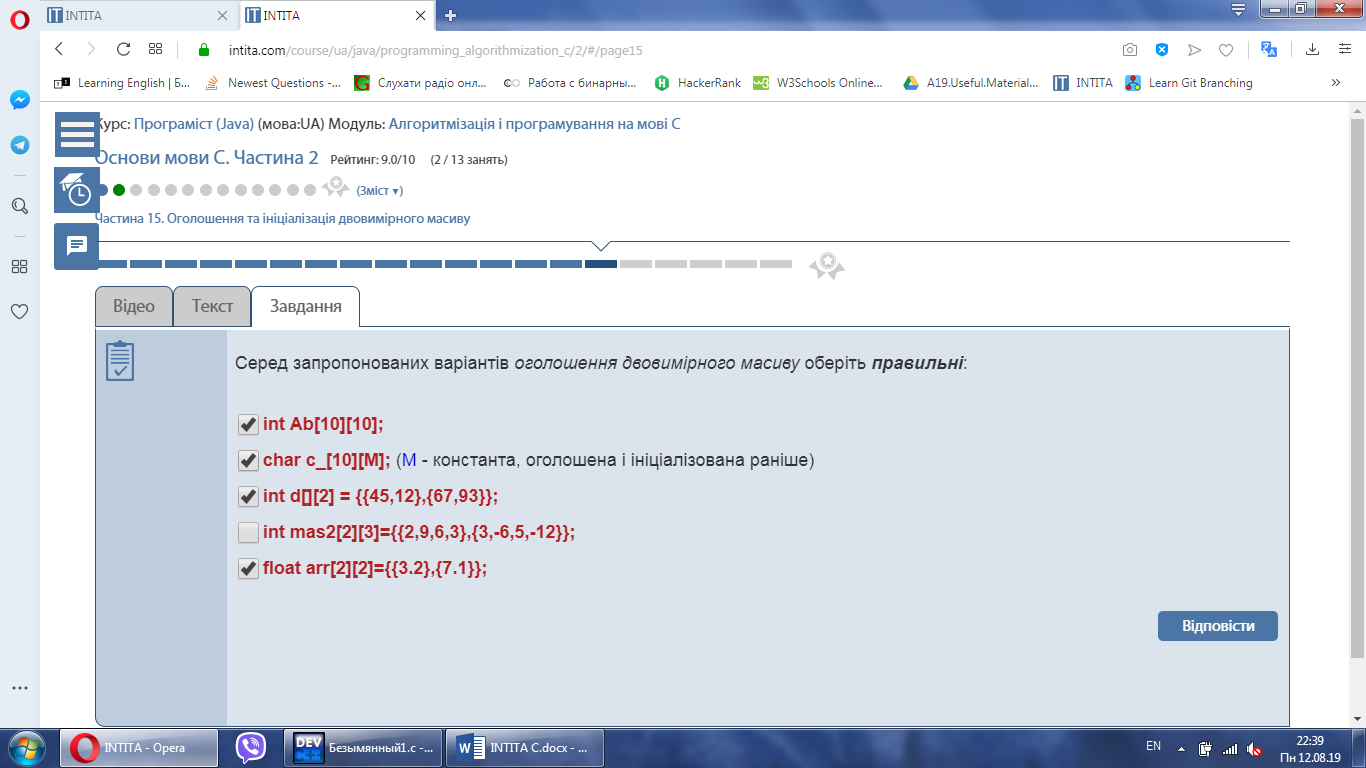


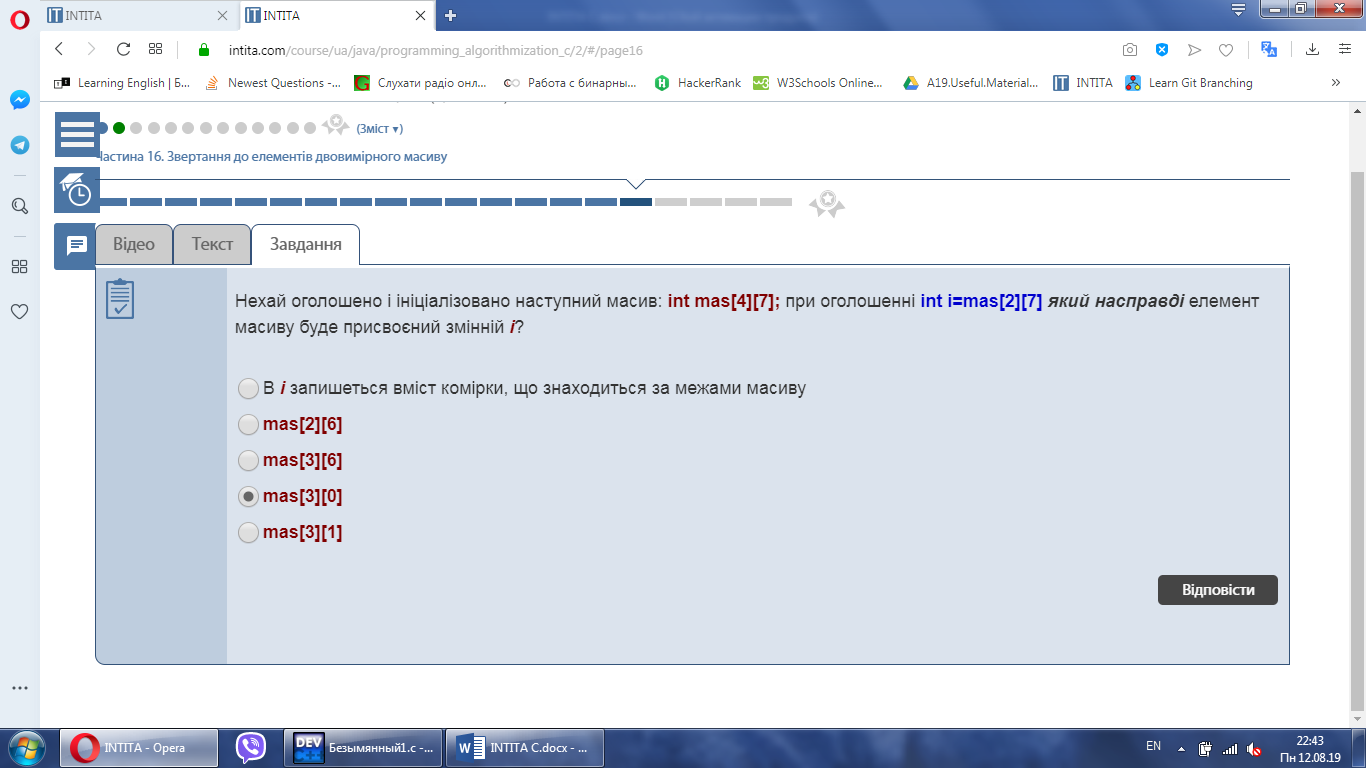


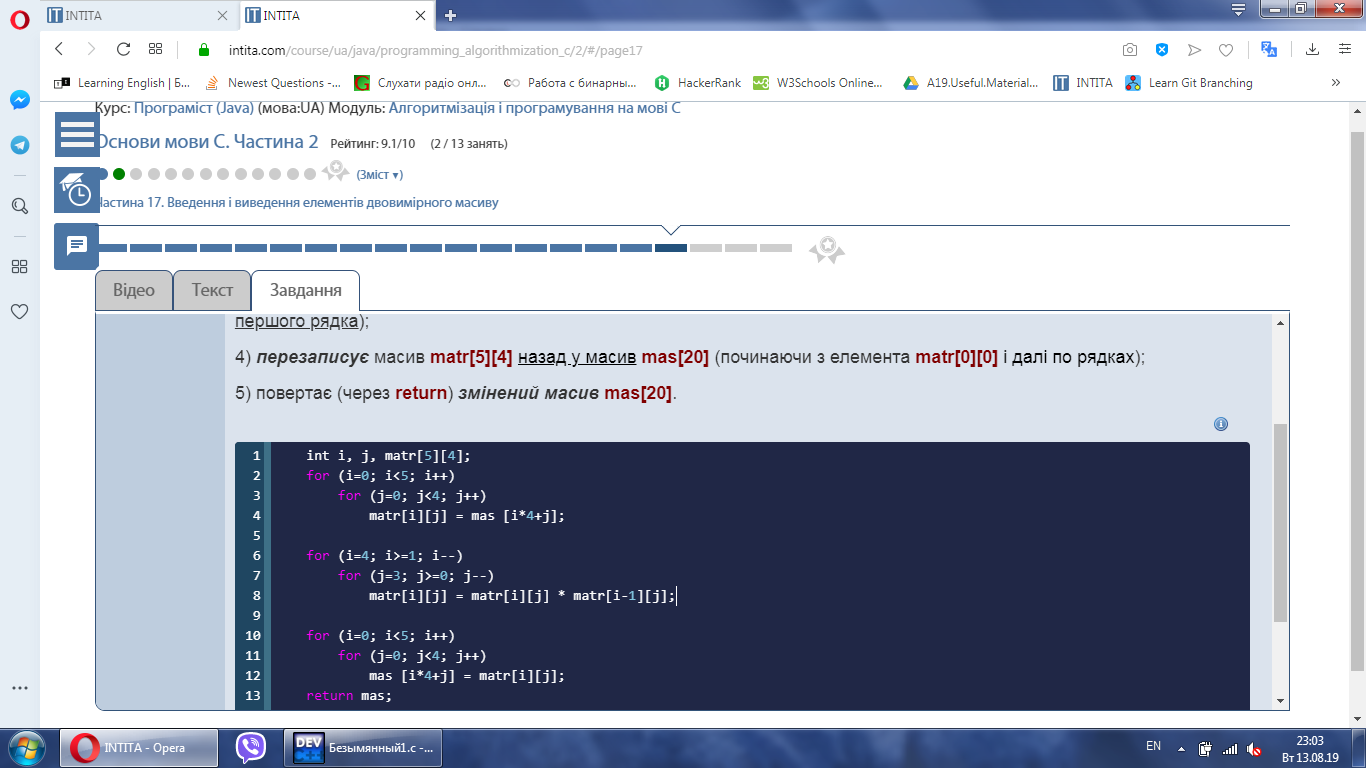


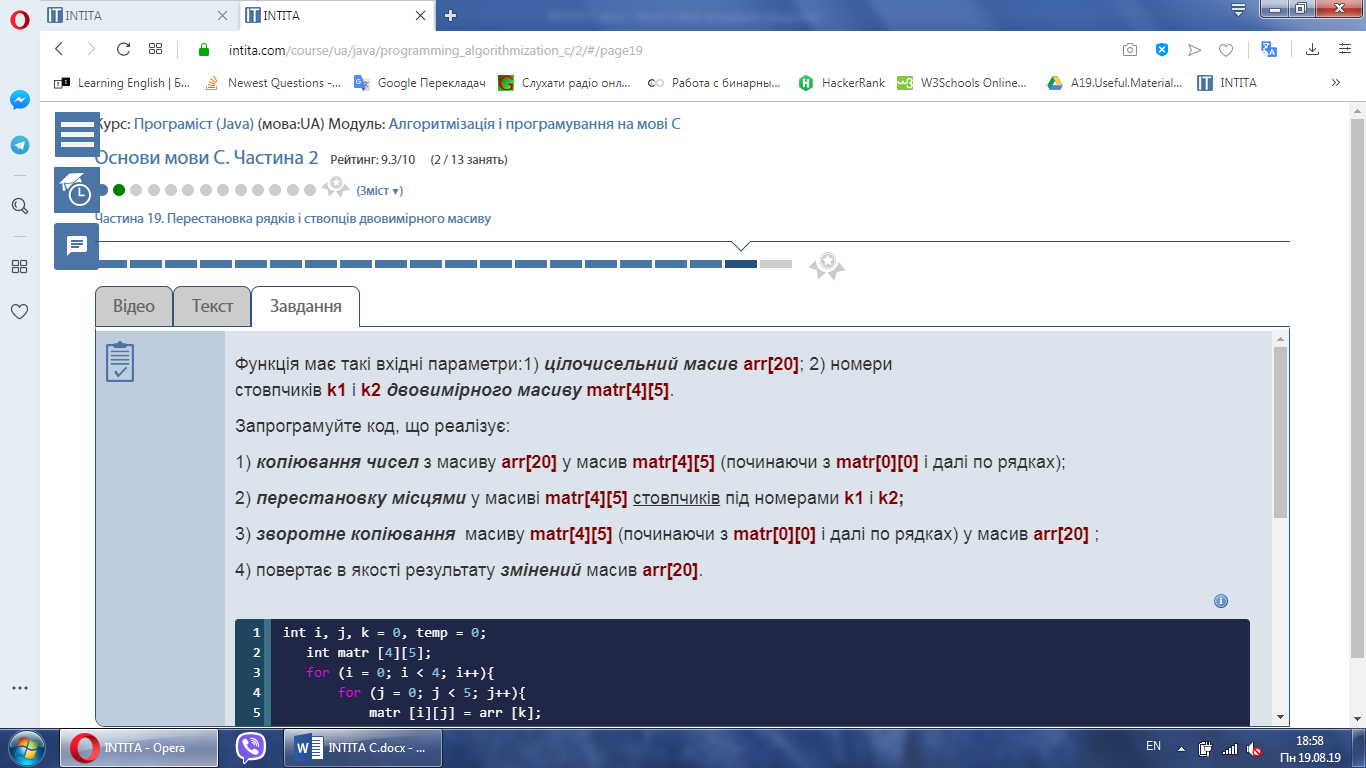
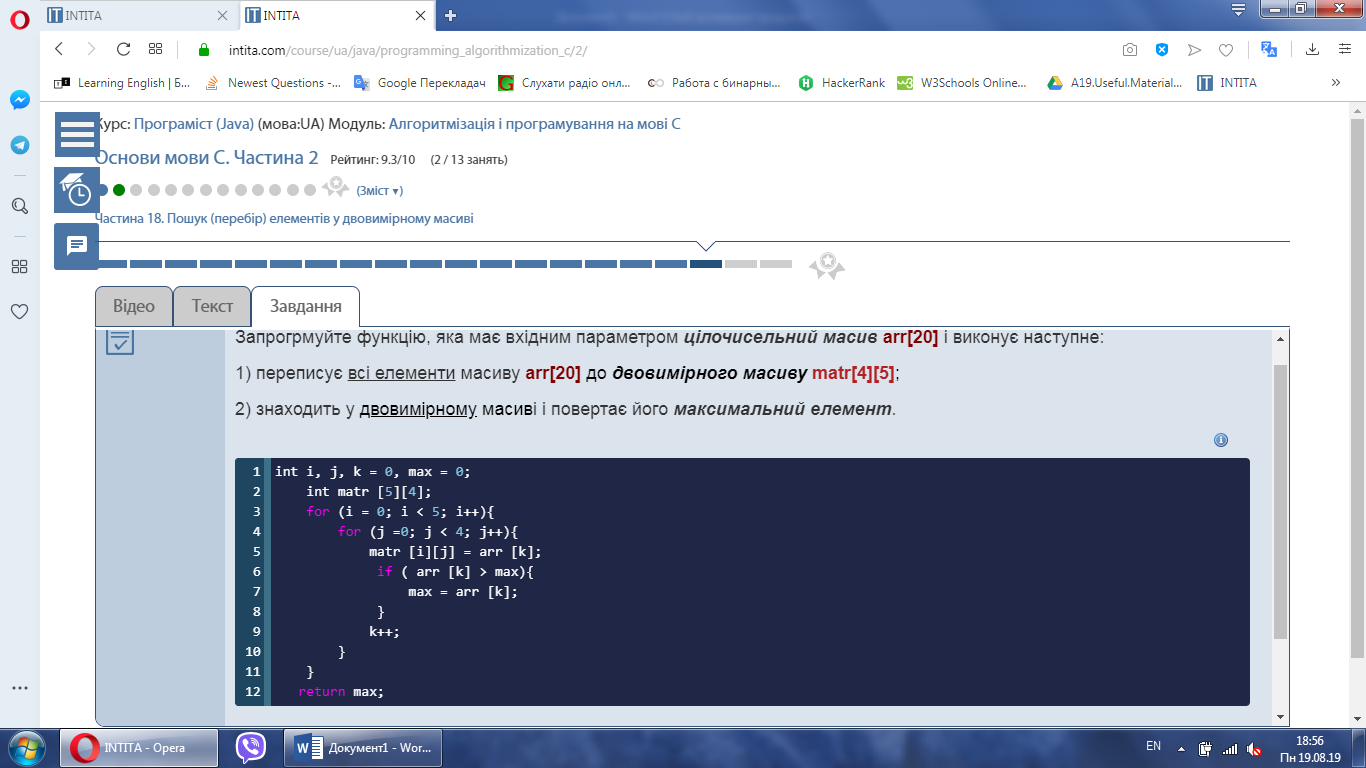












int i, j, k = 0, temp = 0;

int matr [4][5];

for (i = 0; i < 4; i++){

for (j = 0; j < 5; j++){

matr [i][j] = arr [k];

k++;

}

}

for (i = 0; i < 4; i++){

temp = matr [i][k1];

matr [i][k1] = matr [i][k2];

matr [i][k2] = temp;

}

k = 0;

for (i = 0; i < 4; i++){

for (j = 0; j < 5; j++){

arr [k] = matr [i][j];

k++;

}

}

return arr;

Складіть програму, яка б виконувала наступні дії:

1) заповнює числами деякий ***двовимірний масив***;

2) приймає з клавіатури номери одного рядка і одного стовпчика;

3) перевіряє, чи***кількість рядків і стовпчиків*** у масиві ***однакова***;

4) якщо кількість однакова - міняє місцями відповідні елементи вибраних рядка і стовпчика;

5) якщо кількість неоднакова - встановлює менше з цих двох величин (кількість рядків і стовпчиків) і міняє місцямивідповідну кількість елементів у вибраному рядку і стовпчику.

**Примітки:** а) елементи, що знаходяться на перетині вибраних рядка і стовпчика, **обміну не підлягають**(залишаються на місці);

б) зважаючи на попередню примітку, ***на місці залишаються*** також ті елементи, які після можливого обміну мали б стати на місце перетину рядка і стовпчика.

Наприклад, в масиві **arr** 4 рядки і 5 стовпчиків; вибрано 2-й рядок і 4-й стовпчик; оскільки 4 менше 5 – міняємо місцями всі елементи 4-го стовпчика ***з першими 4-ма елементами*** 2-го рядка). При цьому елемент **arr[2][4]** є "недоторканим", так само, як елементи **arr[2][2]** і **arr[4][4]**.

#include <stdio.h>

#include <conio.h>

int main()

{

int M, N;

printf ("Input number the line:");

scanf ("%i", &M);

printf ("Input number the column:");

scanf ("%i", &N);

int i, j, matrix[N][M], line, column;

srand(time(NULL));

for ( i = 0; i < M; i ++ )

for ( j = 0; j < N; j ++ )

matrix[i][j] = rand() % 10;

printf("Matrix a: \n");

for ( i = 0; i < M; i ++ )

{

for ( j = 0; j < N; j ++ )

printf(" a[%i][%i]=%i ", i, j, matrix[i][j]);

printf("\n");

}

printf ("Input the number of line:");

scanf ("%i", &line);

printf ("Input the number of column:");

scanf ("%i", &column);

int min=0;

if (N > M) min = M;

else min = N;

int temp=0;

for (i = 0; i < min; i ++ )

{

if (i == line || i == column) continue;

temp = matrix[line][i];

matrix[line][i] = matrix[i][column];

matrix[i][column] = temp;

}

for ( i = 0; i < M; i ++ )

{

for ( j = 0; j < N; j ++ )

printf(" a[%i][%i]=%i ", i, j, matrix[i][j]);

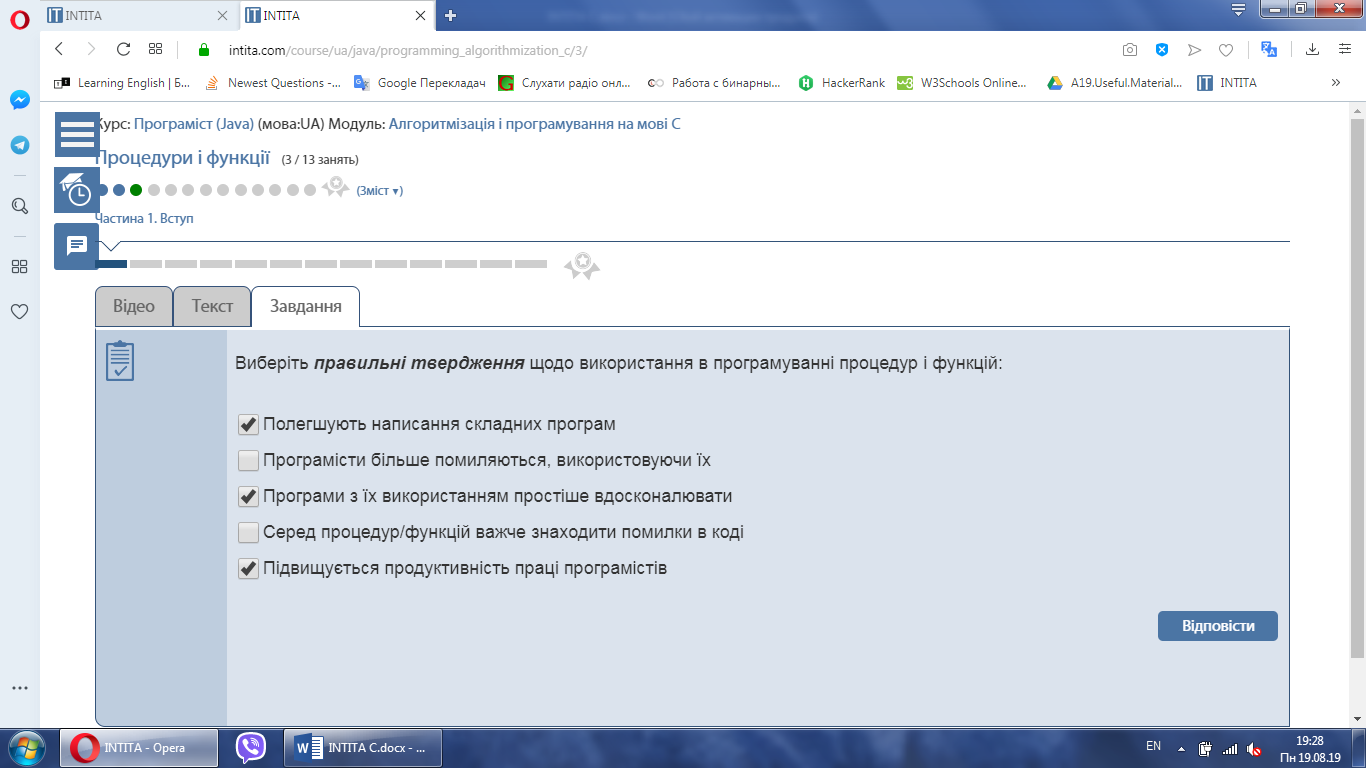
printf("\n");

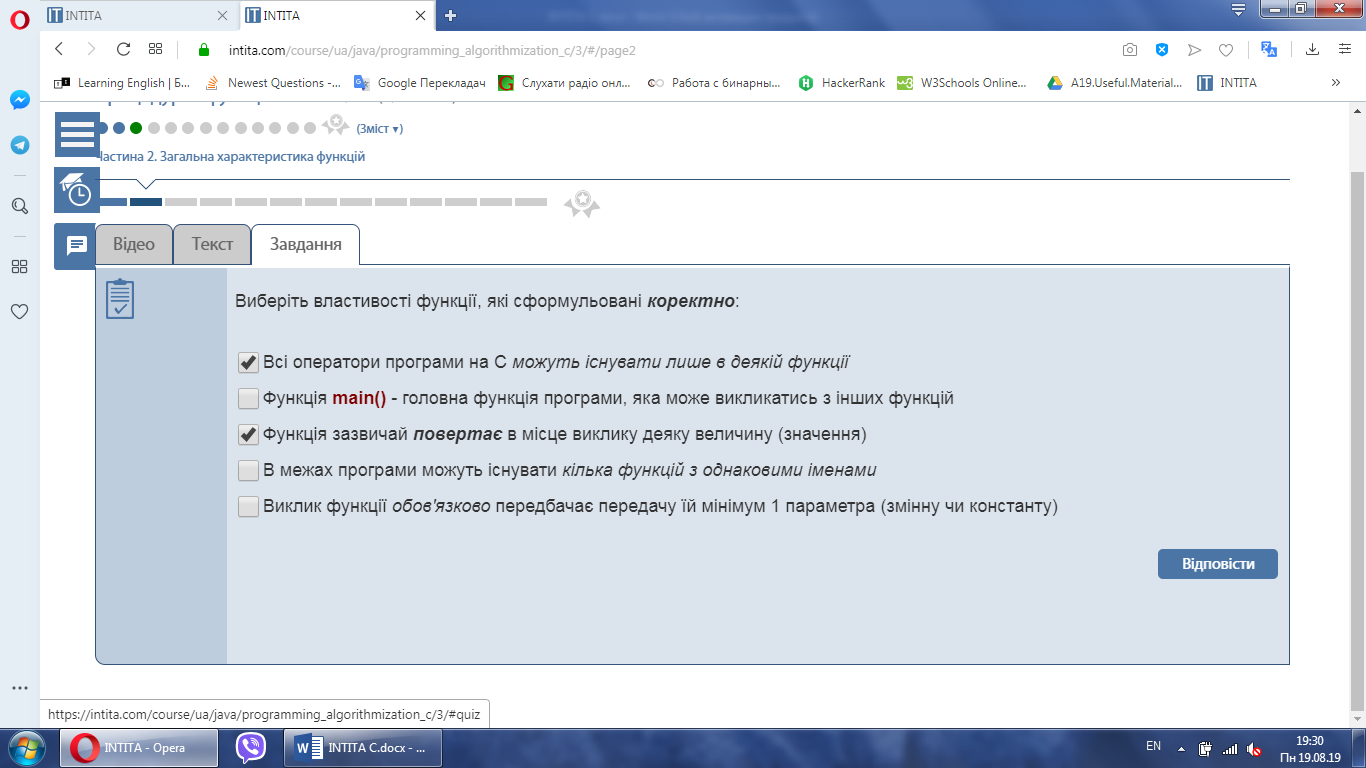
}

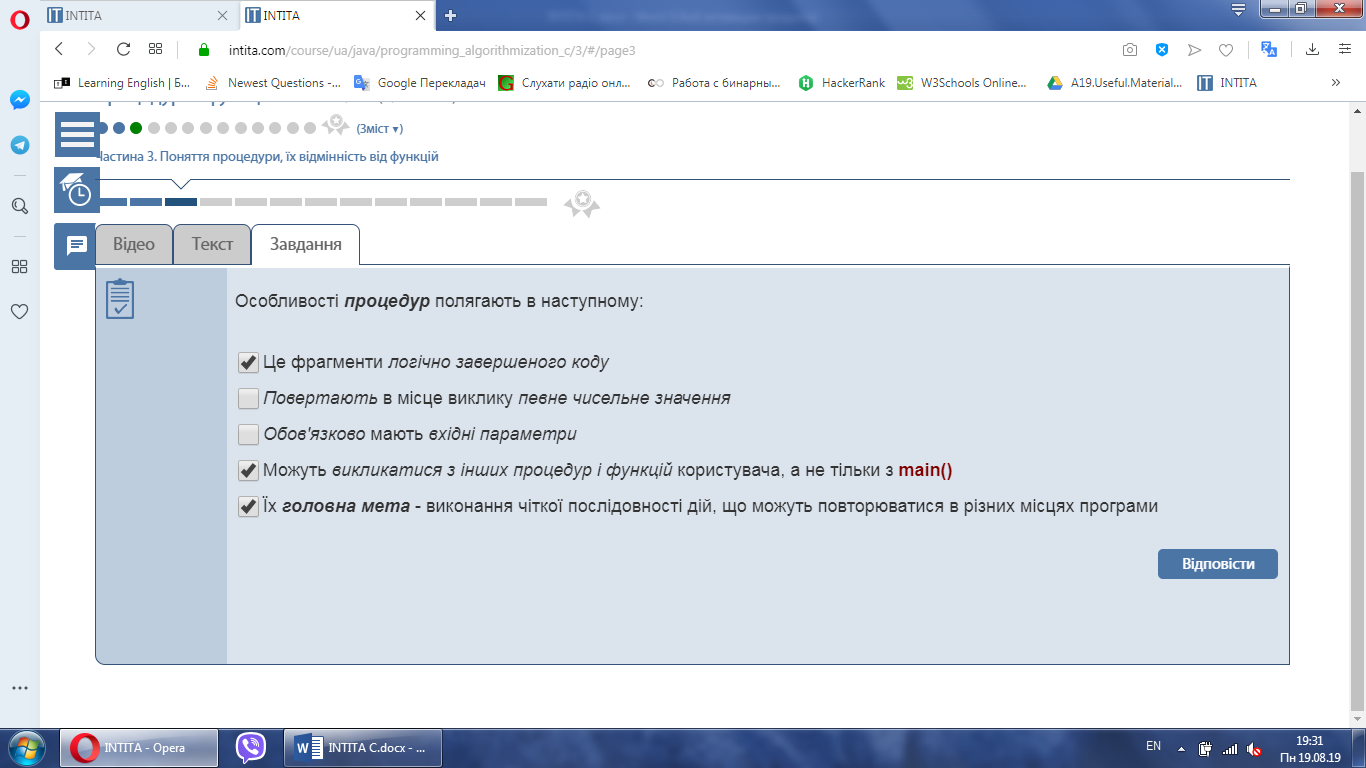
getch ();

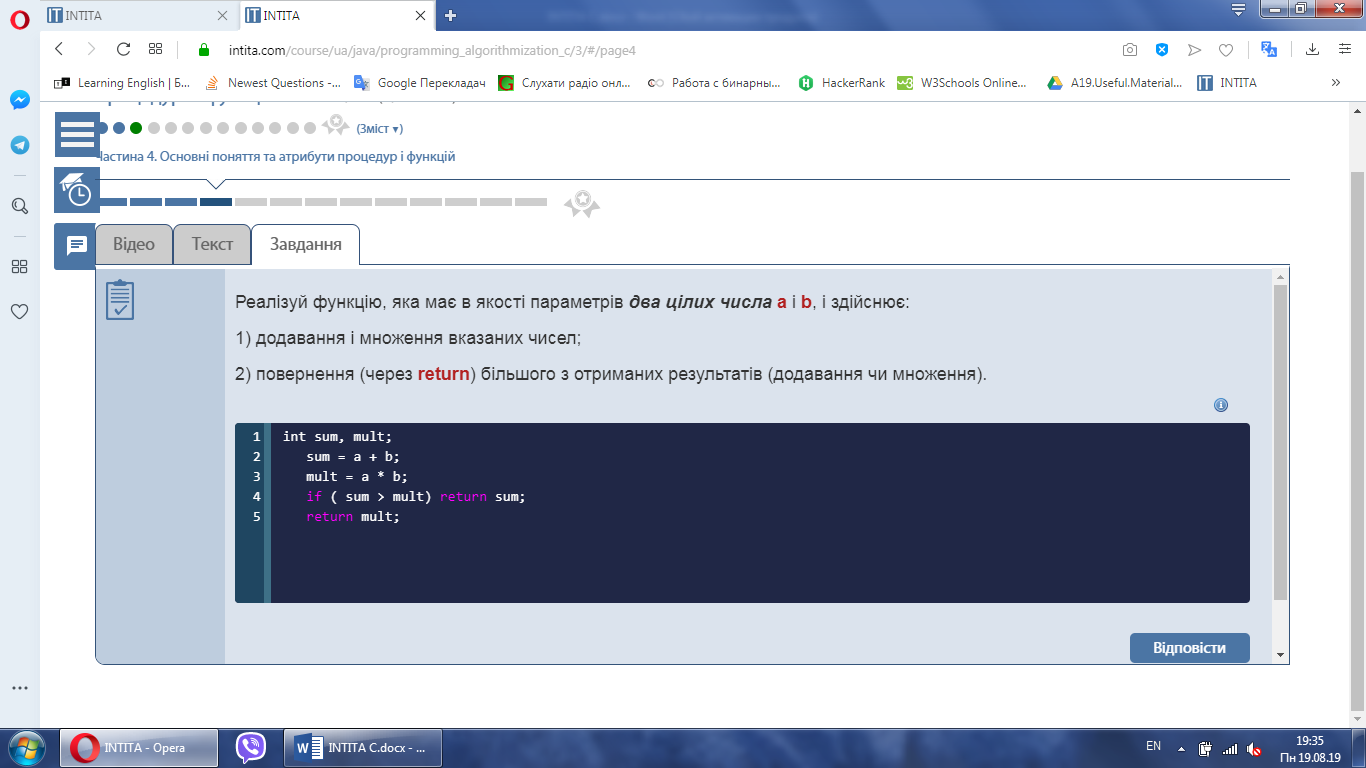
return 0;

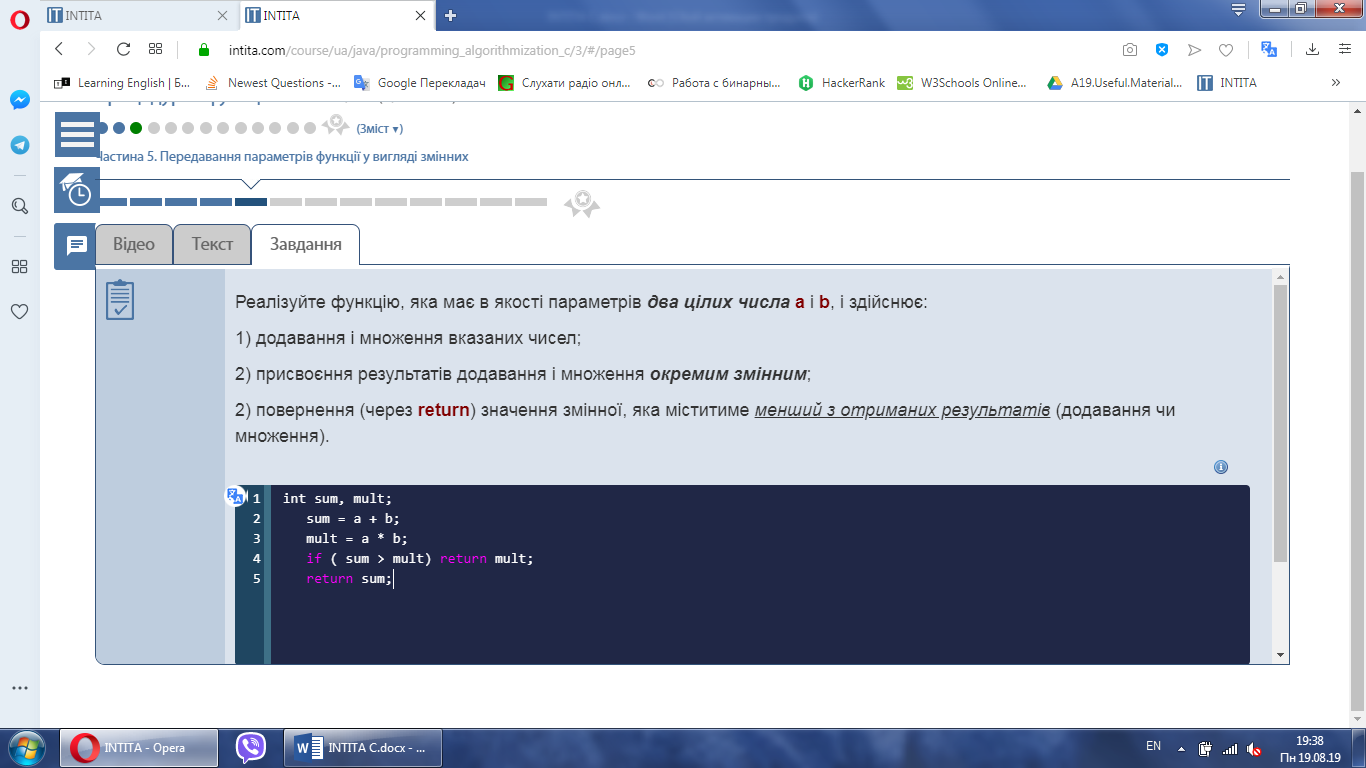
}

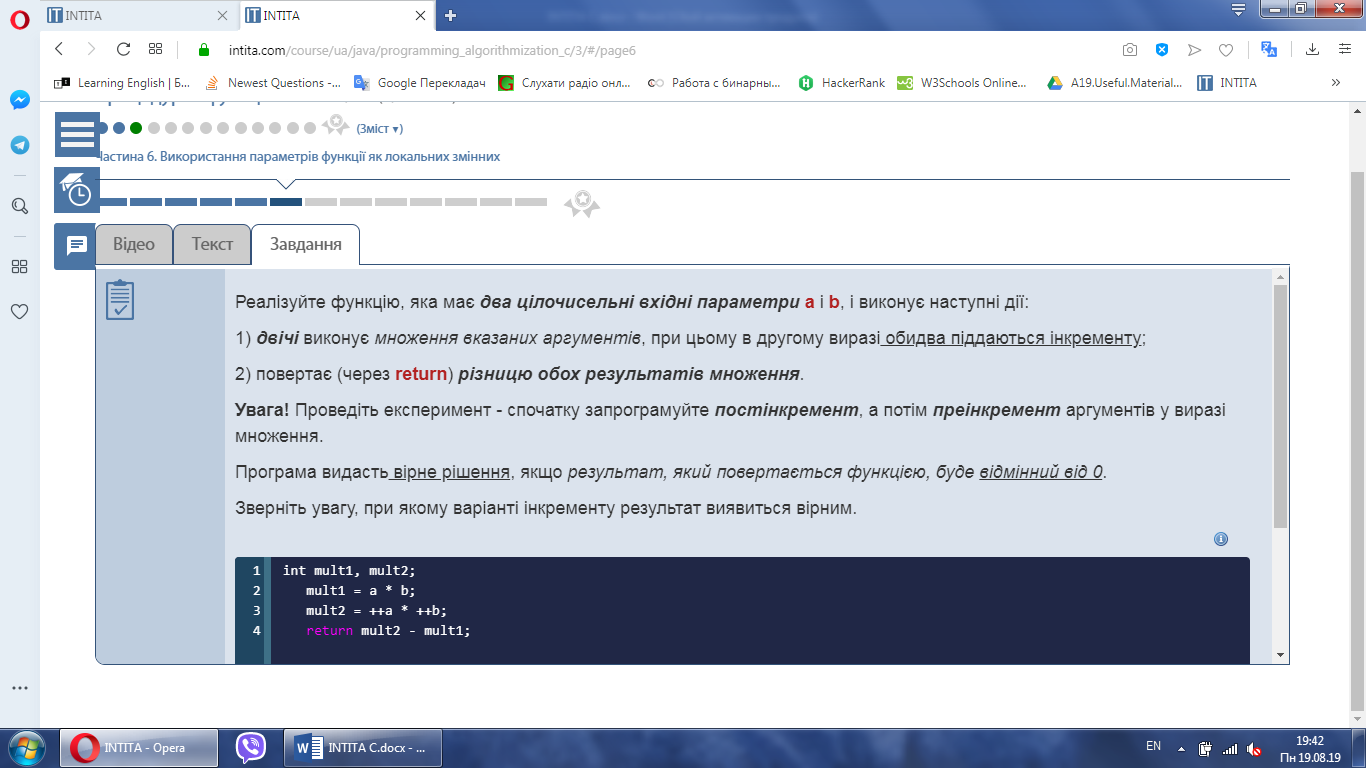












Напишіть програму, яка виконує наступне:

1) приймає з клавіатури ***два цілих числа***;

2) передає їх ***за посиланням*** у функцію **Mul()**;

3) функція **Mul() *інкрементує*** отримані аргументи (змінні з іншими іменами!) і ***перемножує їх***;

4) результат функції **Mul()**передається в основну програму з виведенням на екран у *вигляді виразу* (**наприклад**, з клавіатури прийнято **a=7** i **b=3**; на екрані повинно з'явитися **8\*4=32**).

#include <stdio.h>

#include <conio.h>

int mul (int \*first, int \*second);

int main()

{

int a, b, c;

printf("Enter number: A = ");

scanf("%i", &a);

printf("Enter number: B = ");

scanf("%i", &b);

c = Mul(&a, &b);

printf("%i \* %i = %i", a, b, c);

getch(); return 0;

}

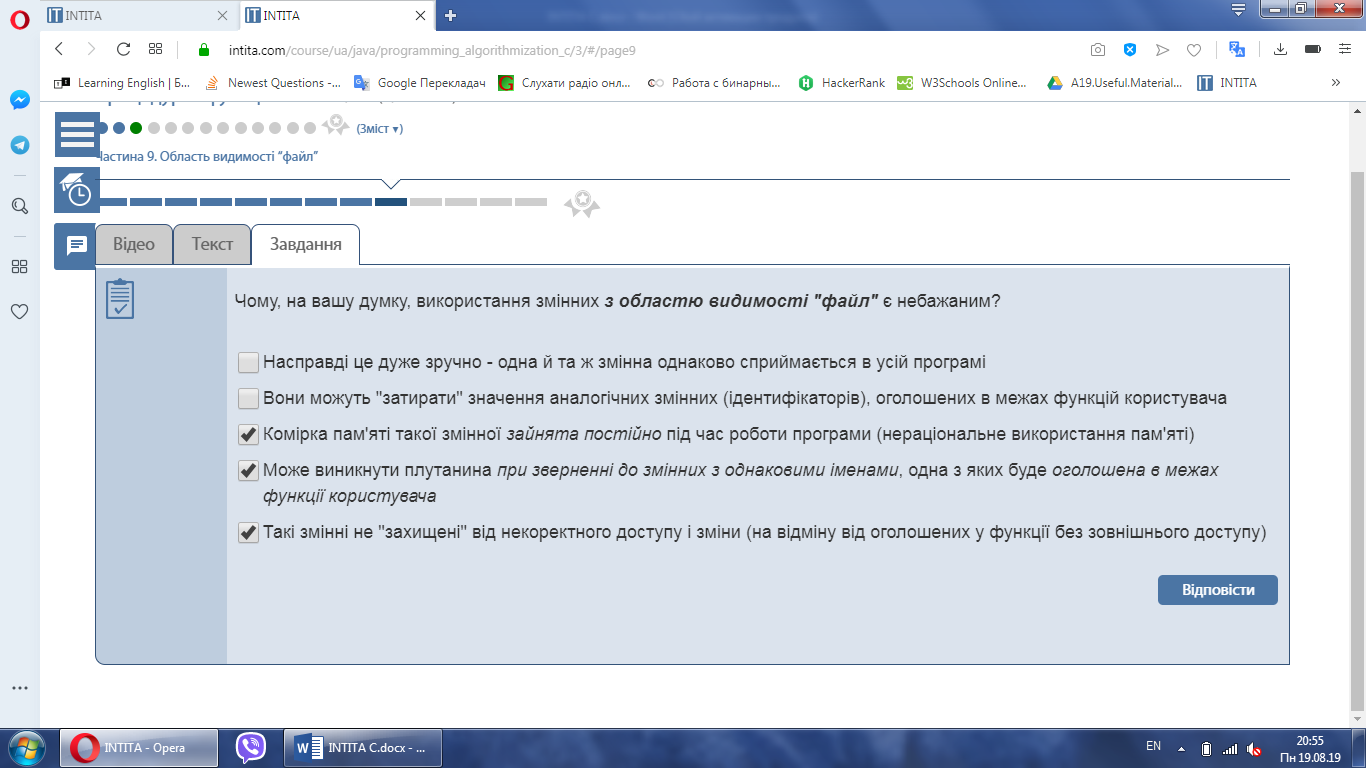
int Mul (int \*first, int \*second)

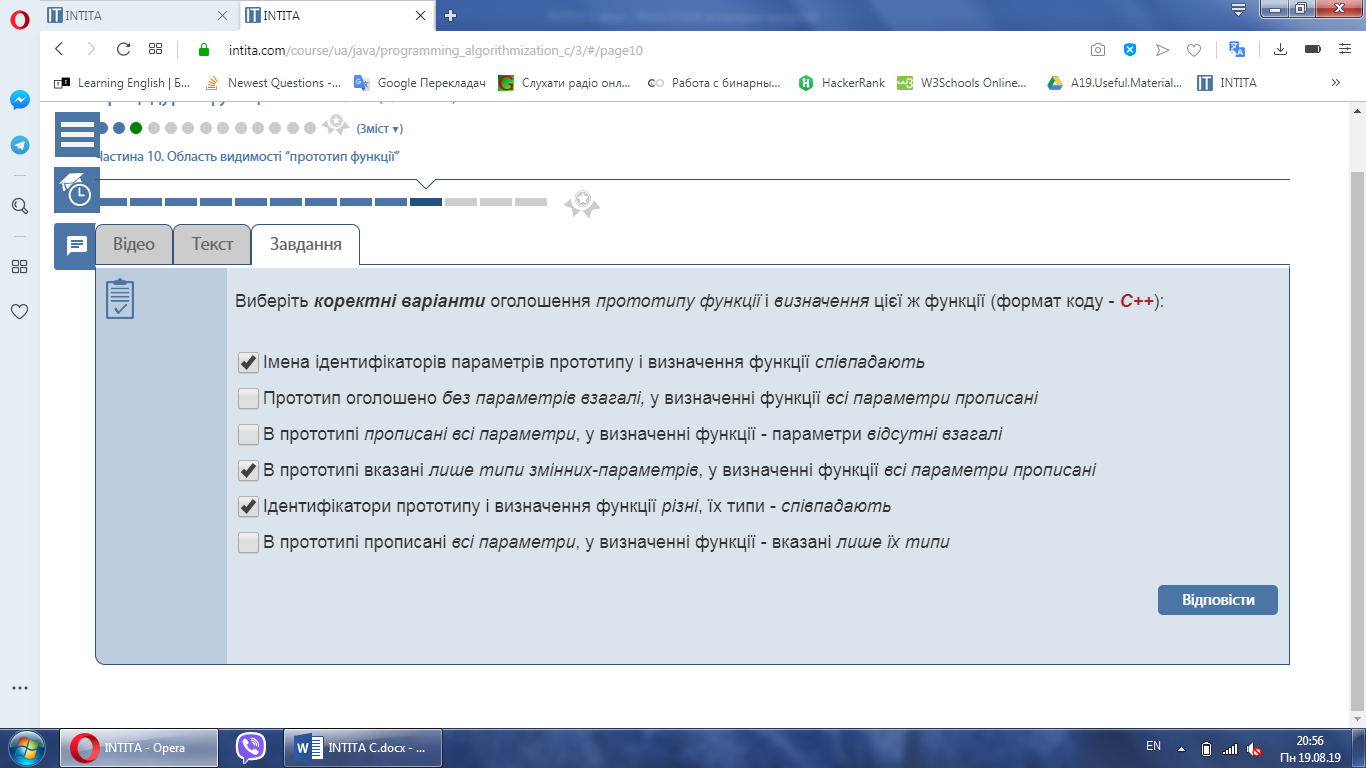
{

return ++\*first \* ++\*second;

}







Оформіть і відкомпілюйте текст наведеного зразка коду, після чого;

1) оцініть вручну (підрахуйте в умі) значення змінних **х**, які будуть виводитися кожною командою **printf()**;

2) запустіть програму і порівняйте результати на екрані з ручними розрахунками, результати поясніть;

3) робочий варіант програми завантажте в поле відповіді і відправте в якості виконання завдання.

#include <stdio.h>

#include <windows.h>

#include <conio.h>

void a ();

void b ();

void c ();

int x = 1;

int main()

{

SetConsoleCP (1251);

SetConsoleOutputCP (1251);

int x = 5; // змінна в області видимості "блок" (зовнішній блок функції main())

printf ("змінна х в зовнішній області видиомості main() = %d\n", x );

{

// початок нової області видимості (внутрішній блок функції main())

int x = 7;

printf ( "змінна х у внутрішній області видимості main() = %d\n", x );

} // кінець нової області видимості (внутрішнього блоку функції main())

printf ( "змінна х в зовнішній області видиомості main() = %d\n", x );

a(); // функція а() має змінну х з власної області видимості "блок"

b(); // функція b() використовує змінну х з області видимості "файл"

c(); // функція c() використовує змінні х з різних областей видимості ("файл" і "блок")

a(); // повторно

b(); // викликаємо

c(); // функції

printf ("\nзмінна х функції main() = %d\n", x );

getch();

return 0;

}

void a()

{

int x = 25; // змінна з областю видимості "блок"

printf ("\nзмінна х функції а() = %d після входу у функцію", x );

x++; // модифікація змінної внутрішнього рівня

printf ("\nзмінна х функції а() = %d перед виходом з функції \n", x );

}

void b()

{

printf ("\nзмінна зовнішнього рівня х = %d при вході у функцію b() \n", x );

x \*= 10; // модифікація змінної зовнішнього рівня

printf ("змінна зовнішнього рівня х = %d при виході з функції b() \n", x );

}

void c()

{

printf ("\nзмінна зовнішнього рівня х =%d при входi у функцію с() \n", x );

x += 15; // модифікація змінної зовнішнього рівня

printf ( "змінна зовнішнього рівня х = %d після модифікації в с() \n", x );

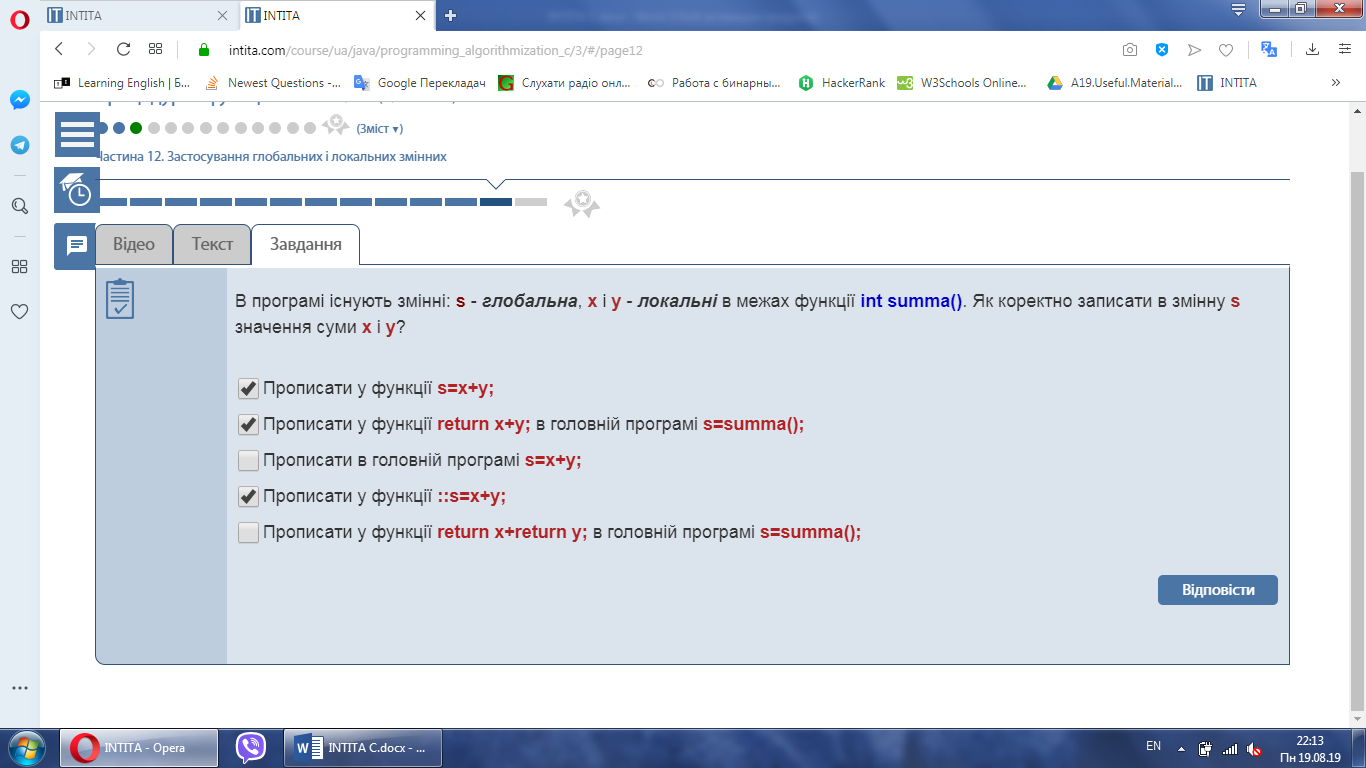
int x = 77; // початок дії області видимості змінної внутрішнього рівня х

printf ("\nзмінна внутрішнього рівня х = %d після оголошення в с()", x );

x--; // модифікація змінної внутрішнього рівня

printf ("\nзмінна внутрішнього рівня х = %d перед виходом з функції с() \n", x );

}



Реалізуй програму, що виконує наступні дії:

1) *приймає з клавіатури* деяке **N**-цифрове ***додатне ціле число*** (з перевіркою на додатність  і на кількість цифр за допомогою циклу **do-while**);

2) за допомогою *процедури* **Div()** розбиває його ***на окремі цифри*** (справа наліво), які *записує в масив*;

3) *приймає з клавіатури* число **K**- номер деякої цифри, починаючи з **1**;

4) за допомогою *функції* **NSymbol()** знаходить і повертає**K**-ту цифру в числі, рахуючи ***зліва направо*** (функцію викликати, якщо **K<=N**, інакше *вивести повідомлення про помилку*).

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

void Div (int num, int j, int\* point);

int NSymbol (int K, int\* point);

int main()

{

int number, N=0;

printf ("Input positive number:");

scanf ("%i", &number);

int temp = number;

if (temp < 1)

{

printf ("Number do not positive");

return -1;

}

do

{

temp/=10;

N++;

}while (temp>0);

int\* array = (int\*)malloc(N\*sizeof(int));

Div (number, N, array);

int K;

printf ("Input number of element K:");

scanf ("%i", &K);

if (K <= N)

{

NSymbol (K, array);

}

else printf ("K is greater than the digit of the first number");

getch();

return 0;

}

void Div (int num, int j, int\* point)

{

int i;

for (i=0; i<j; i++)

{

\*(point+i) = num%10;

num/=10;

}

}

int NSymbol (int K, int\* point)

{

return point[K-1];

}