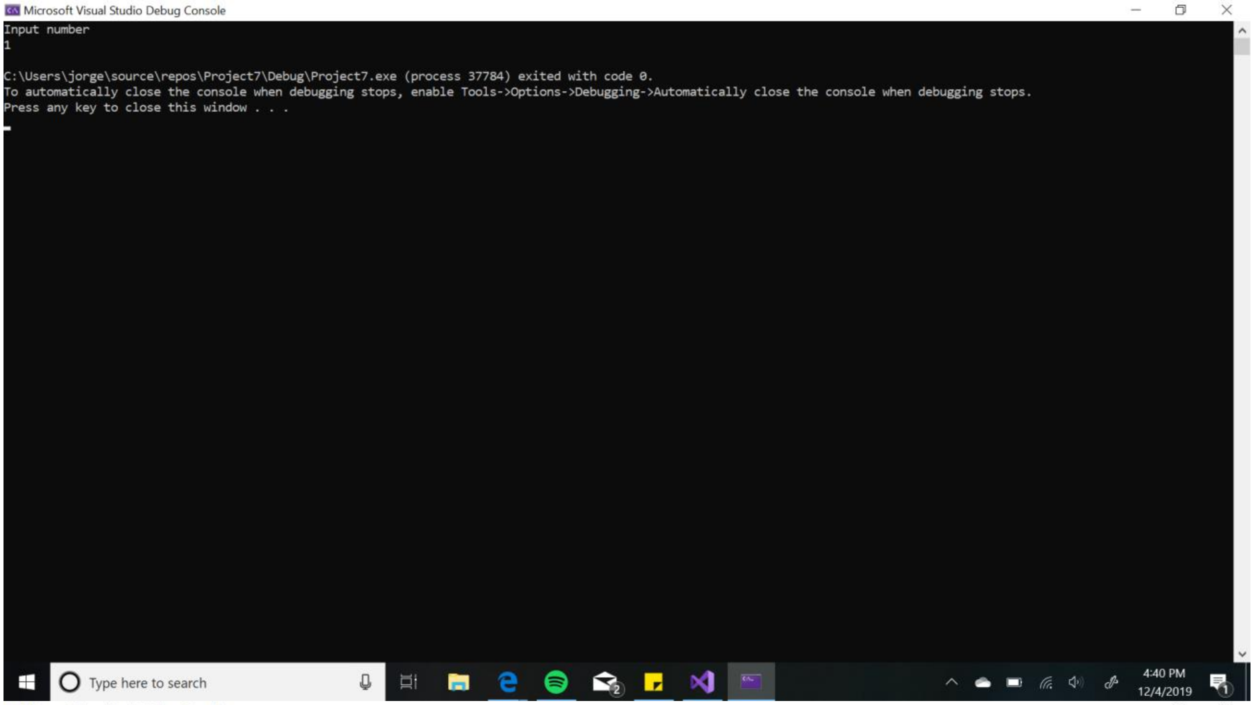
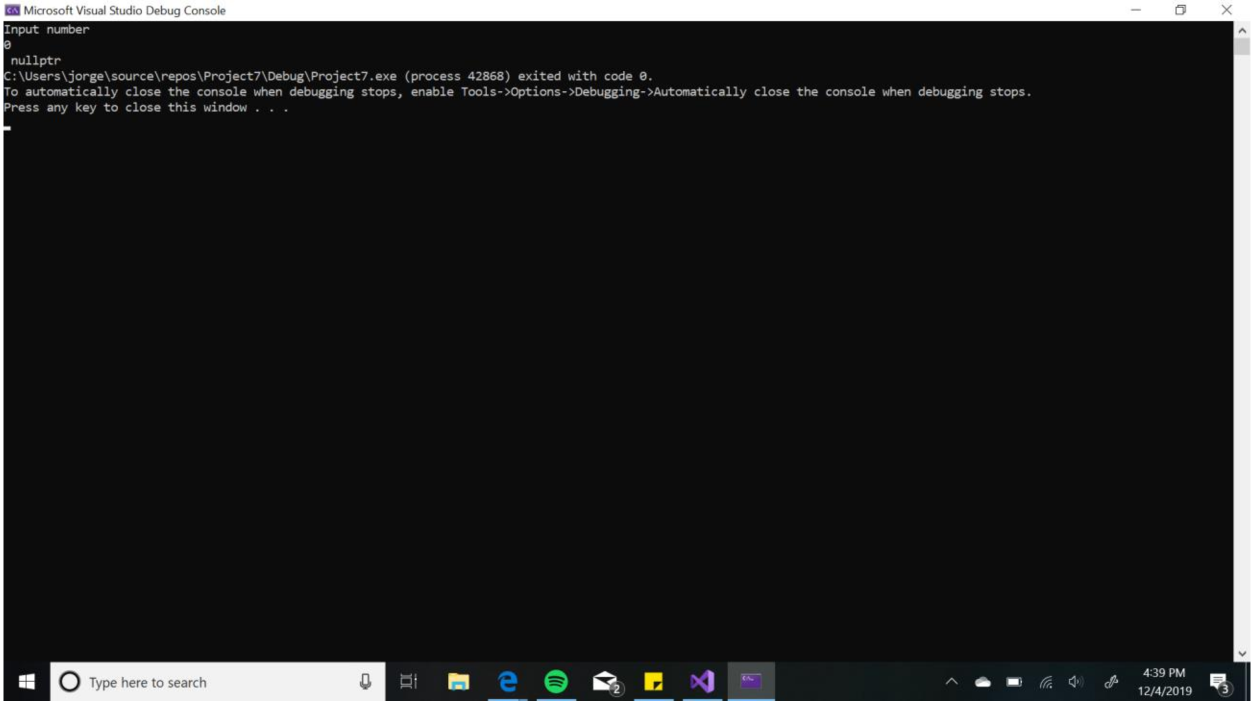
Program Challenge 1.

First **check** if *input size* is **negative or 0**, in that case *return* a **nullptr**. Otherwise, **dynamically** allocate memory using ***new***, then *return* that **pointer**.

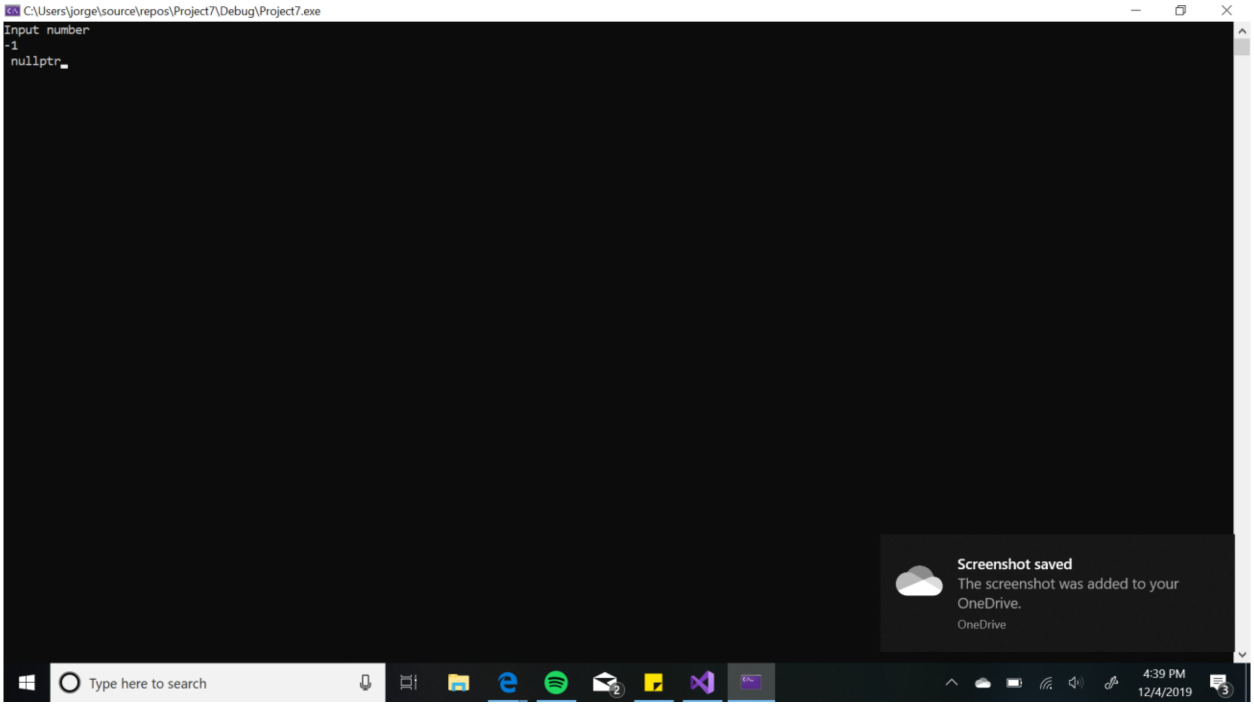
```
//Jorge Rivas
#include<iostream>
using namespace std;
int main()
{
        int numelements;
        int *arrayallocator(int numelements);
        {
                cout << "Input number" << endl;
                cin >> numelements;
                if (numelements <= 0)
                {
                        cout <<"nullptr";
                }
                int *ptr = new int[numelements];
                return 0;
        }
                return 0;
 }
```



```
Input number
1

C:\Users\jorge\source\repos\Project7\Debug\Project7.exe (process 37784) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```



```
Input number
0
 nullptr
C:\Users\jorge\source\repos\Project7\Debug\Project7.exe (process 42868) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```



C:\Users\jorge\source\repos\Project7\Debug\Project7.exe

```
Input number
-1
 nullptr_
```

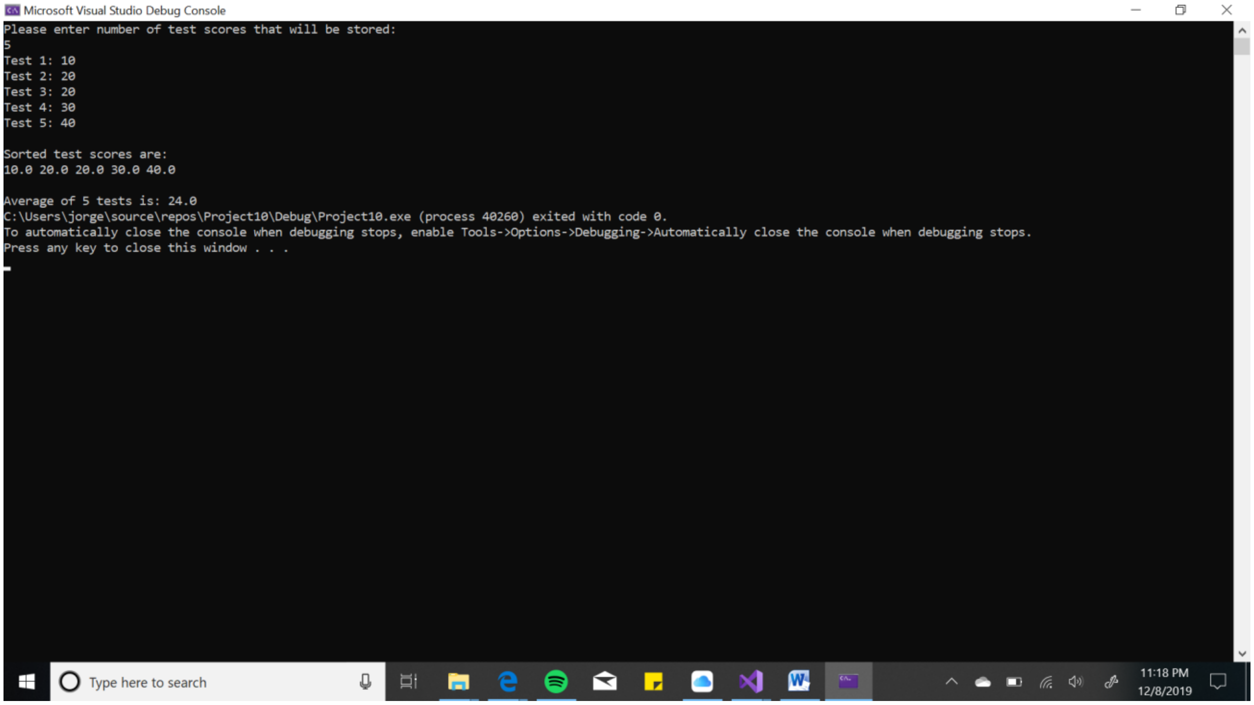
Screenshot saved
The screenshot was added to your OneDrive.
OneDrive

Program Challenge 2.

```
//Jorge Rivas
#include <iostream>
#include <iomanip>


using namespace std;

void sortArray(double*, int);
double getAverage(double*, int);
void sortArray(double* ptr, int arraySize)
{

        double temp;


        bool swapped = true;

        while (swapped)
        {
                swapped = false;

                for (int counter = 0; counter < arraySize - 1; counter++)
                {

                        if (*(ptr + counter) > * (ptr + counter + 1))
                        {
                                temp = *(ptr + counter + 1);
                                *(ptr + counter + 1) = *(ptr + counter);
                                *(ptr + counter) = temp;
                                swapped = true;

                        }
                }
        }
double getAverage(double* ptr, int arraySize)
        {

        double total = 0.0;


        for (int counter = 0; counter < arraySize; counter++)
        {
                total += *(ptr + counter);
        }


        return (double)total / arraySize;
        }


int main()
{

        int numTests;


        cout << "Please enter number of test scores ";
        cout << "that will be stored:\n";
        cin >> numTests;


        double* testScores = new double[numTests];


        for (int counter = 0; counter < numTests; counter++) {
                cout << "Test " << counter + 1 << ": ";


                cin >> *(testScores + counter);


                while (*(testScores + counter) < 0)
                {
                        cout << "ERROR! Negative values not allowed!";
                        cout << " Enter again!\n";
                        cin >> *(testScores + counter);
                }
        }


        sortArray(testScores, numTests);


        cout << fixed << setprecision(1);
```

```
	cout << endl;
	cout << "Sorted test scores are: \n";
	for (int counter = 0; counter < numTests; counter++) {
		cout << *(testScores + counter) << " ";
	}


	cout << "\n\nAverage of " << numTests << " tests is: ";
	cout << getAverage(testScores, numTests);


	return 0;
}
```

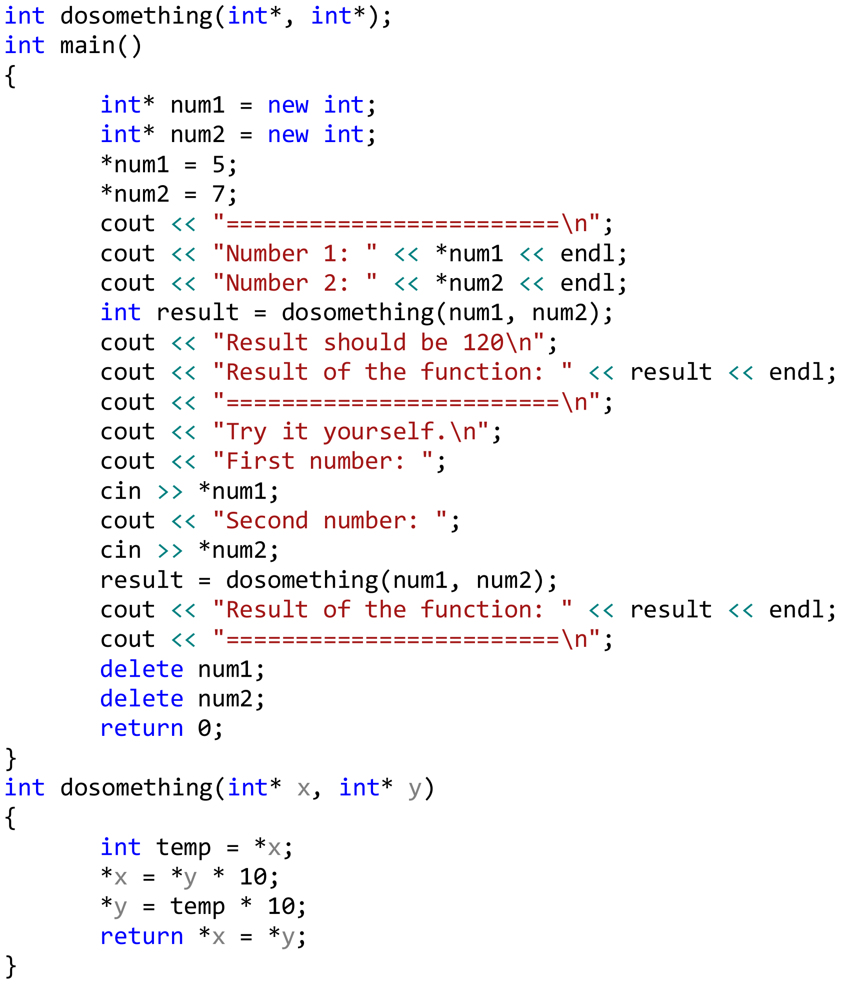
Read number of test that user will input. **Dynamically allocate** using *new* operator. Pointer notation is used everywhere in program. Don't forget that *when using pointer notation* to work with values, first they should be **dereferenced**. Functions **sortArray()** and **getAverage()** will accept as argument *pointer to array*, and *array size* stored from user input.



```
Please enter number of test scores that will be stored:
5
Test 1: 10
Test 2: 20
Test 3: 20
Test 4: 30
Test 5: 40

Sorted test scores are:
10.0 20.0 20.0 30.0 40.0

Average of 5 tests is: 24.0
C:\Users\jorge\source\repos\Project10\Debug\Project10.exe (process 40260) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

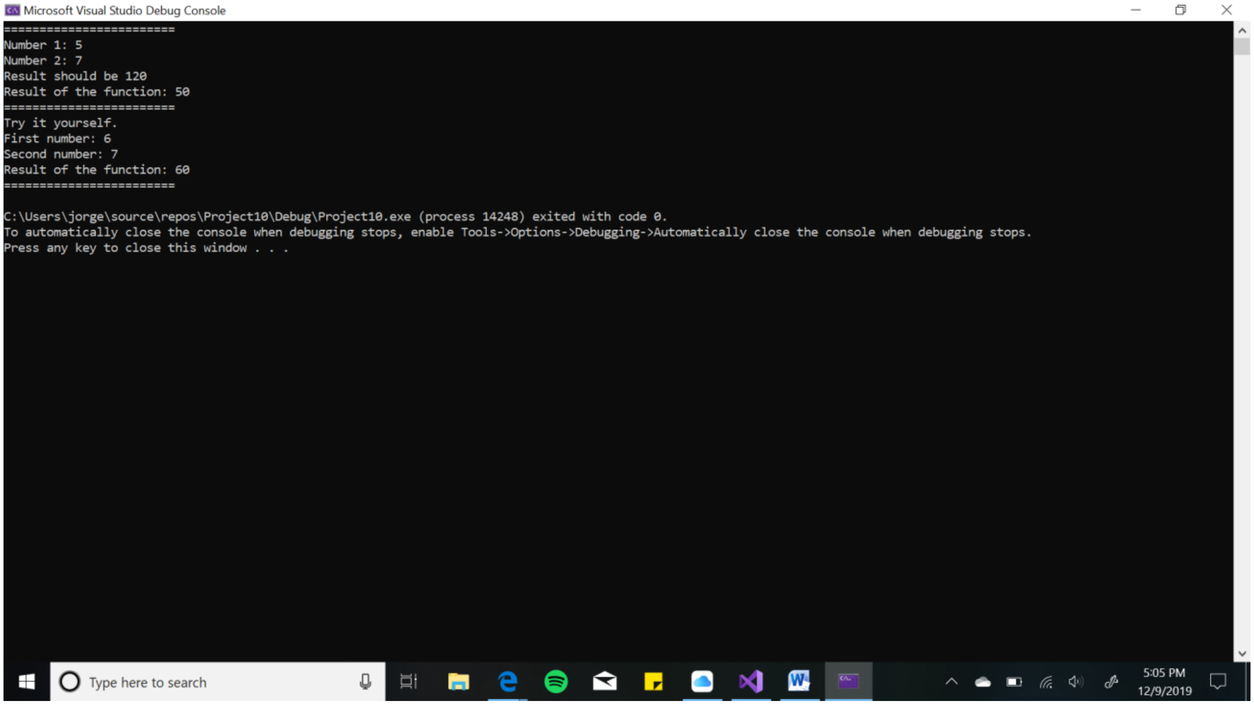
Program Challenge 5.

```
//Jorge Rivas
#include <iostream>

using namespace std;
```



```
int dosomething(int*, int*);
int main()
{
	int* num1 = new int;
	int* num2 = new int;
	*num1 = 5;
	*num2 = 7;
	cout << "========================\n";
	cout << "Number 1: " << *num1 << endl;
	cout << "Number 2: " << *num2 << endl;
	int result = dosomething(num1, num2);
	cout << "Result should be 120\n";
	cout << "Result of the function: " << result << endl;
	cout << "========================\n";
	cout << "Try it yourself.\n";
	cout << "First number: ";
	cin >> *num1;
	cout << "Second number: ";
	cin >> *num2;
	result = dosomething(num1, num2);
	cout << "Result of the function: " << result << endl;
	cout << "========================\n";
	delete num1;
	delete num2;
	return 0;
}
int dosomething(int* x, int* y)
{
	int temp = *x;
	*x = *y * 10;
	*y = temp * 10;
	return *x = *y;
}
```

Simply **modify** function so that it uses **pointer notation**. Remember to **dereference** pointers when working with their **values**. Do **not confuse** *dereference* operator with *multiplication* operator. Also keep in mind that, when returning, *dereference* operator has **precedence** over *addition* operator.



```
========================
Number 1: 5
Number 2: 7
Result should be 120
Result of the function: 50
========================
Try it yourself.
First number: 6
Second number: 7
Result of the function: 60
========================

C:\Users\jorge\source\repos\Project10\Debug\Project10.exe (process 14248) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```