Name and, if possible, ID#: _____

AMERICAN UNIVERSITY OF ARMENIA
College of Science and Engineering
COMP120 Introduction to Object-Oriented Programming
MIDTERM 2 EXAM

Date:	Tuesday, March 24 2015
Starting time:	10:30
Duration:	1 hour 20 minutes
Attention:	**ANY COMMUNICATION IS STRICTLY PROHIBITED**

14/15

Please write down your name at the top of all used pages

Problem 1

The easiest way to implement rotation by 90^0 of a square array is to transpose it and then reverse all its rows separately. Write a C++ function ***void rotate(int *a2D, int size)*** that takes as its argument a pointer to the first element of a square array ***int *a2D*** of the specified ***int size*** and rotates its. Use already implemented functions ***void reverse(int a1D[], int length)*** and ***void transpose(int *a2D, int size)***:

```
void reverse(int a1D[], int length)
{
      for (int i = 0; i < length / 2; i++)
            swap(a1D[i], a1D[length - 1 - i]);
}

void transpose(int *a2D, int size)
{
      for (int row = 0; row < size; row++)
            for (int col = row + 1; col < size; col++)
                  swap(a2D[row * size + col], a2D[col * size + row]);
}
```

```
Void rotate(int *a2D, int size) {
  Transpose(a2D, size);
  For (int *row = a2D; row < a2D + size*size; row+=size)
      reverse(row, size); }
```

5

oop.mt2.240315.H087

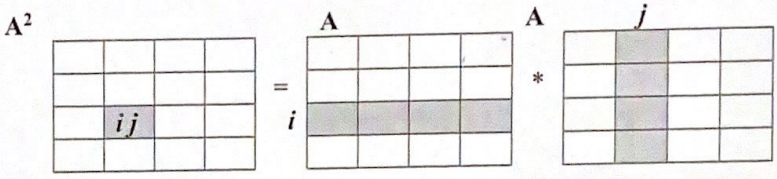
Use the backside, if needed

Name and, if possible, ID#: _____

Problem 2

Using functions ***transpose()*** from **Problem 1** and ***scalar()*** from below, write a C++ function ***void square(int *a2D, int *product, int size)*** that takes as its argument a pointer to the first element of a square array ***int *a2D*** of the specified ***int size***, computes its square (multiplies it by itself) and saves it in another square array of the same size, the pointer to the first element of which is given by ***int *product***. Each element p_{ij} in the i^{th} row and j^{th} column of the array ***product** is the scalar product of the i^{th} row and j^{th} column of the array ***a2D** and is calculated by the expression: $p_{ij} = \sum_{k=0}^{size-1} a_{ik} a_{kj}$

```
int scalar(int a[], int b[], int length)
{
      int result = 0;
      for (int i = 0; i < length; i++)
            result += a[i] * b[i];
      return result;
}
```




```
Void square(int *a2D, int *product, int size)
{int B[size*size]; new is zquized
 for(int C=0; C< size*size; C++)
   B[C]=a2D[C];
 Transpose(B, size);
 For(int*i=a2D; i<a2D+(size*size); i+=size)
   For(int*j=B; j<B+(size*size); j+=size)
{*product=scalar(i,j,size);
 Product++; } perfect!
```

4

OOP.MT2.240315.H087

Name and, if possible, ID#: _____

Problem 3

Using functions ***segment()*** from below and ***rotate()*** from **Problem 1**, write a C++ function ***void spiral2(int *a2D, int even_size)*** that takes as its argument a pointer to the first element of a square array ***int *a2D*** of the specified even size ***int even_size*** and fills it with two spirals of ***zeros*** and ***ones***. The entire first row starting from the first element is filled with ***zeros*** and, symmetrically, entire last row starting from the last element is filled with ***ones***. Then, the entire last column, except the last element, is filled with ***zeros*** and, symmetrically, the entire first column, except the first element – with ***ones***. And so on, until the central elements are reached. A shaded example is shown below:

```
int* segment(int *start, int length, int direction, int increment)
{
      for (; length > 0; length--)
      {
            *(start + direction) = *start + increment;
            start += direction;
      }
      return start;
}
```

0	0	0	0	0	0
1	1	1	1	1	0
1	0	0	0	1	0
1	0	1	1	1	0
1	0	0	0	0	0
1	1	1	1	1	1

void spiral2(int *a2D, int even_size)
{ *a2D = 0 ; *(a2D+1) = 0
int* a=segment(a2D, size-1, +1, 0)
for (i = size-2 ; i > 1 ; i-= 2) {
a= segment(a, i, *size, 0);
a = segment(a, i, -size-1, 0); }
a = segment(a, i-2, -size, 0);
a= segment(a, i-2, 1, 0) ; }
rotate(a2D, size); rotate(a2D, size);
int* a2D = 1 ; *(a2D+1) = 1
int* b = segment(a2D, size-1, +1, 0)
the same thig but instead of (a)->b ~~and the increment is 0~~
~~in case that all elements were zero~~
~~from the beginig~~
}
rotate(a2D, size); rotate(a2D, size); }

OOP.MI2.240315.HOB7

5