Name and, if possible, ID#:[redacted]

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**

7/15

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

Problem 1

The easiest way to implement rotation by **90⁰** 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]);
}
```

```
#include <iostream>
Using namespace std.;

int main()
  cons int size;             int a1D[length];
 int a2D[size][size];        int length=size;
 int * a2D;

void rotate(int * a2D, int size){

  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]);}
```

const must be immideately initialized

you may not declare a function inside another one

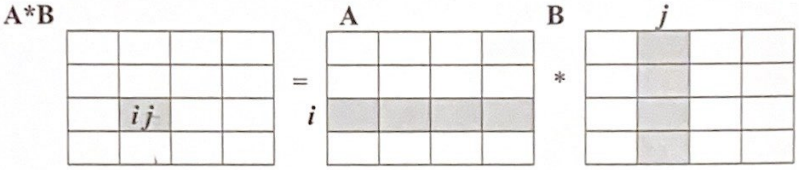
Use the backside, if needed

Name and, if possible, ID#:____

Problem 2

Using functions ***transpose()*** from **Problem 1** and ***scalar()*** from below, write a C++ function ***void mult(int *a2D, int *b2D, int *product, int size)*** that takes as its arguments pointers to the first elements of square arrays ***int *a2D*** and ***int *b2D*** of the same specified ***int size***, computes their product 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 of ****a2D*** and j^{th} column of ****b2D*** and is calculated by the expression: $p_{ij} = \sum_{k=0}^{size-1} a_{ik} b_{kj}$

```
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;
}
```



```
int main(){
 void mult(int a* 2D, int *b2D, int *product, int size){
     void transpore (int *b2D, int size){ see Problem 1
         for (int row=0; row<size row++){
             for(int col=row+1; col<size; col++)
                 swap(b2D[row*size+col], b2D[col*size+row]);
         }

   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;
   }

 }
```

1

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:

→1

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

↓-size

←-1 ↑-size

→

+6

Size

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

↑ +6

+6 ↓ -size

-6

←-1

```
int main(){
    const int even_size;
void spiral2(int*a2D, int even_size){
  while (int(length)%10==0){
   while (int increment=0){
     int length = even_size;
     int direction = 1
     int* start = int * a2D);
     ___________________________
     int* segment();
     int* a2D = int* a2D + even-size
     length = even_size -2
     int direction = -even_size
     int* segment();
     int* a2D = int* a2D - 1;
     length = even_size -2;
     int direction = -1
     int* segment();
     int* a2D = int* a2D - even_size;
     length = even_size - 4
     int direction = - even_size
     int* segment();
                                        int* a2D = int* a2D+1;
                                        length = even_size-4;
                                        int direction = 1
                                        int* segment();}
                                    else if(increment=1){
                                        rotate();
                                        rotate();
                                        X
                                        }
                                        }
                                      }
```

3