National University of Computer and Emerging Sciences, Lahore Campus

Course:	**CP Lab**	**Code:**	**CL103**
Program:	**BS (Computer Science)**	**Semester:**	**Fall 2018**
Duration:	**150 minutes**	**T. Marks:**	**40**
Date:	**Tuesday 30-10-2018**	**Weight**	**30**
Section:	**A, B, C**	**Page(s):**	**2**
Exam:	**Lab midterm**		

Instructions/Notes:

- Use of the internet, notes, codes, lab manuals, and flash drives is strictly prohibited.
- Plagiarism will result in **F** grade in lab.
- Code must be **indented properly**, failure to comply will incur a penalty.

Question # 1: **(20 marks)**

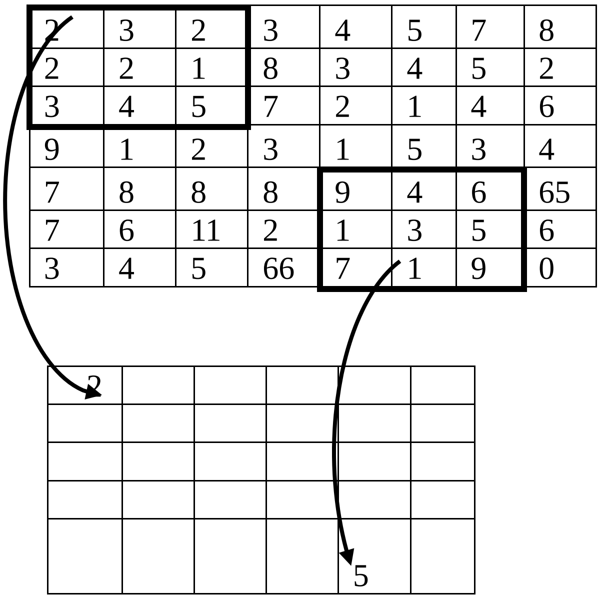
Implement the function compress, the compresses a 2D array to size R x C, and compresses it to size (R-2) x (C-2). The compression is done as follows: taking a slot and its 9 neighboring cells, and calculate their average. Store the average in output.

Int**compress(int **input, int rows, int cols);

Note: The input array will have minimum 3 x 3 rows and columns.

The working of the function is explained in the diagram below:

Input (7x8):



2	3	2	3	4	5	7	8
2	2	1	8	3	4	5	2
3	4	5	7	2	1	4	6
9	1	2	3	1	5	3	4
7	8	8	8	9	4	6	65
7	6	11	2	1	3	5	6
3	4	5	66	7	1	9	0

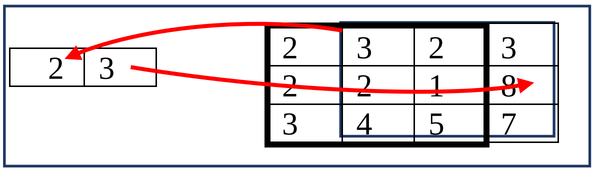
Output(5x6):

2					
				5	

For this you have to:

1. Write a function **int **Input (int &r, int &c);**
   which allocates the memory with r and c values after taking input from user.
2. Write a function **void output (int **matrix, int r, int c);**
   Which print the matrix.
3. Write a function **void deallocate (int **matrix, int c);**
   Which de-allocates the memory.

Call the Compress function which will return a compressed 2D array, you need to print that as well. And finally deallocate all the memory.



2	3

2	3	2	3
2	2	1	8
3	4	5	7

Here you can see, the input array is 3 x 4 and output array is 1 x 2.
Input (R x C) -> Output (R-2 x C-2).
In Output array, (first value is 2 which is average of first 9 elements and the value 3 is average of second 9 elements)

Question # 2: **(20 marks)**

In this question, you have to implement a HugeInt class that behaves like the int type, but can handle integers up to(maximum) 15 decimal digits. You will use dynamic arrays to store digits of various Huge Integers.

Define appropriate **constructors** *(Default,Parameterized and copy constructor)* and **destructor** required for this class.

Functions:

1. Arithmetic Function

   HugeInt addHugeInt (HugeInt obj2);
2. Function for comparing HugeInt numbers for

   bool isLessThan(HugeInt obj2)
3. Output Function for printing HugeInt number to the screen.
4. Input Function to take HugeInt number from the user.
   - Take input a HugeInt Number.
   - **Hint**: **For our ease,** you can take input in char array and then put every value in integer array.To take number from character array is like:

     char c = ‘5’; int num = c - 48;

Note: The integers can be positive only. Protype of HugeInt class and main function is given below:

```
class HugeInt{
            int *data, size;
public:
            void input();
            void output();
            HugeInt();
            HugeInt(int *d, int s);
            HugeInt(const HugeInt& obj);
            bool isLessThan(HugeInt& obj);
            HugeInt add(HugeInt obj);
            ~HugeInt()
};
```

```
void main()
{
            HugeInt obj1Int;
            obj1Int.input();
            int data[] = {1,2,3,4,5,6,7,8,9,4,3,1}, size = 12;
            HugeInt obj2Int(data,size);
            HugeInt obj3Int = obj1Int.add(obj2Int);
            obj3Int.output();
            cout<<" = ";
            obj1Int.output();
            cout<<" + ";
            obj2Int.output();
            cout<<endl;
            if(obj2Int.isLessThan(obj3Int)
            {
                cout<<"I am lesser\n";
            }
            else{
                cout<<"I am bigger\n";
            }
}
```

Output:

```
        Input a huge number size: 4
        Input a number: 5432
        123456794863 = 5432 + 123456789431
        I am lesser.
        Press any key to continu...
```