National University of Computer and Emerging Sciences, Lahore Campus

Course Name:	Introduction to Computing	Course Code:	CS101
Program:	BS(CS)	Semester:	Spring 2018
Duration:	3 hr	Total Marks:	80
Paper Date:	Monday, 21 May 2018	Weight	40
Section:	ALL	Page(s):	8
Exam Type:	Final	Roll No - Sec	-

Instruction/Notes:

Marks Obtained	Q 1	Q 2	Q 3	Q 4	Q 5	Total

1. Solve the exam on this question paper.
2. You can use sheets for rough work but **do not attach or submit**, with paper.

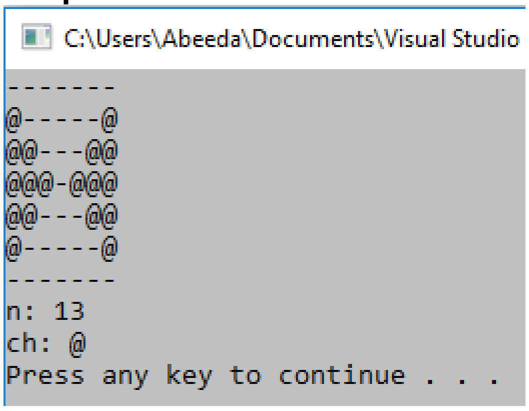
Question # 1: Write the output of following code. **[10 Marks]**

```
void doSomeMoreWork(int & m, int & n, char ch){
      for (int i = 0; i < n; i++){
            for (int j = 0; j < n - i - 1; j++)
                  cout << ch;
            for (int j = 0; j < m; j++)
                  cout << " ";
            for (int j = 0; j < n - i - 1; j++)
                  cout << ch;
            m = m + 2;
            cout << endl;
      }
      n = m + n;
}
```

```
void doSomework(int & n, char ch){
      n = n / 2 + 1;
      int m = (n * 2) - 1;
      for (int i = 0; i < n - 1; i++){
            for (int j = 0; j < i; j+
+)
                  cout << ch;
            for (int j = 0; j < m; j+
+)
                  cout << " ";
            for (int j = 0; j < i; j+
+)
                  cout << ch;
            m = m - 2;
            cout << endl;
      }
      m = 1;
      doSomeMoreWork(m, n, ch);
}
```

```
int main()
{
      int n = 6;
      char ch = '@';
      doSomework(n, ch);
      cout << "n: " << n << endl;
      cout << "ch: " << ch << endl;
      return 0;
}
```

Output:



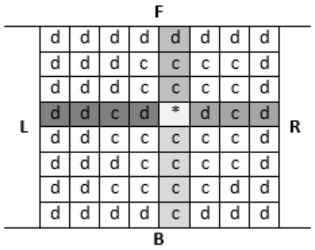
```
-------
@-----@
@@---@@
@@@-@@@
@@---@@
@-----@
-------
n: 13
ch: @
Press any key to continue . . .
```

Question # 2: [15 Marks]

Write a C++ function "**ScrambleWord**" that changes the order of characters in a word in a given string. For changing the order of letters, the simplest thing you could do is either sort the word in ascending or descending order.
For example, if the string given to the function is, "**National University of Computer and Emerging Sciences**", and the word to scramble is "**Computer**", then the original string should change to,
"National University of **Cemoprtu** and Emerging Sciences", if the word is sorted in ascending order.

Question # 3: [2+3+10= 15 Marks]

Robotics International is starting its new project “**RoboCleaner**”. RoboCleaner is a robot which will clean the carpets. For cleaning, it



d	d	d	d	d	d	d	d
d	d	d	c	c	c	c	d
d	d	d	c	c	c	c	d
d	d	c	d	*	d	c	d
d	d	c	c	c	c	c	d
d	d	d	c	c	c	c	d
d	d	c	c	c	c	d	d
d	d	d	d	c	d	d	d

Figure 1 – 8×8 carpet’s sample configuratio

divides a carpet into 8×8 blocks. If there is any garbage in a cell then those cells in block are marked **dirty** (by character ‘d’), otherwise it is marked **clean** (by character ‘c’). The block having RoboCleaner in it, is marked with character ‘*’ (star). RoboCleaner always keeps its face towards the first row. It has four eyes so it can see in four directions i.e. front, back, left and right (from the cell in which it is standing).

For example, according to the carpet configuration shown in Figure 1, **RoboCleaner’s eyes give following readings:**

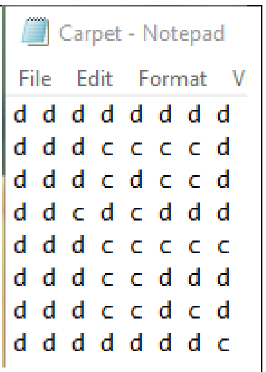
Front Eye – 1 dirty block(s) in the front	**Back Eye** – 0 dirty block(s) at the back
Right Eye – 2 dirty block(s) on right	**Left Eye** – 3 dirty block(s) on left

Your task is to write a program which performs following tasks:

1. Declare a carpet of 8×8 blocks and initialize blocks’ state as given in file carpet.txt and
2. Ask user to enter current position of robot (current row and current column. you can assume that user will never enter out of bound indices). Mark the cell having robot in it (robot can overwrite a dirty or clean block)

```
    char arr[8][8];
    int ri, rj;
    ifstream fin;
    fin.open("Carpet.txt");
    for (int i = 0; i < 8; i++){
          for (int j = 0; j < 8; j++)
                fin >> arr[i][j];
    }
    fin.close();

    cout << "Enter RoboCleaner Position: ";
    cout << "i: ";
    cin >> ri;
    cout << "j: ";
    cin >> rj;
    arr[ri][rj] = '*';
```



```
d d d d d d d d
d d d c c c c d
d d d c d c c d
d d c d c d d d
d d d c c c c c
d d d c c d d d
d d d c c d c d
d d d d d d d c
```

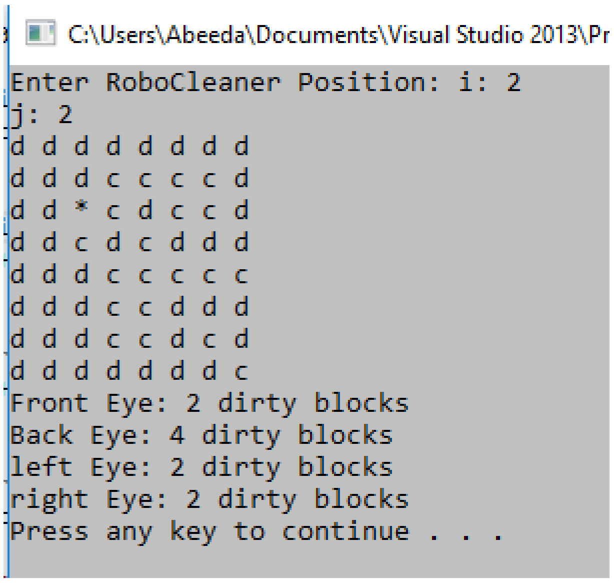
3. Print the carpet.

```
        //Print
                for (int i = 0; i < 8; i++){
                        for (int j = 0; j < 8; j++)
                                cout<< arr[i][j] << " ";
                        cout << endl;
        }
```

4. Write a function **SeeCarpet** which will print the information seen by all four eyes of the robot from its current position as given in example.

```
 void seeCarpet(char arr[][8], int size, int ri, int rj){
       //Front
       int front = 0, back = 0, left = 0, right = 0;
       for (int i = ri - 1; i >= 0; i--){
             if (arr[i][rj]=='d')
                   front++;
       }
       for (int i = ri + 1; i < size; i++){
             if (arr[i][rj] == 'd')
                   back++;
       }
       for (int j = rj - 1; j >=0; j--){
             if (arr[ri][j] == 'd')
                   left++;
       }
       for (int j = rj +1; j <size; j++){
             if (arr[ri][j] == 'd')
                   right++;
       }
       cout << "Front Eye: " << front << " dirty blocks" << endl;
       cout << "Back Eye: " << back << " dirty blocks" << endl;
       cout << "left Eye: " << left << " dirty blocks" << endl;
       cout << "right Eye: " << right << " dirty blocks" << endl;

}
```



```
C:\Users\Abeeda\Documents\Visual Studio 2013\Pr
Enter RoboCleaner Position: i: 2
j: 2
d d d d d d d d
d d d c c c c d
d d * c d c c d
d d c d c d d d
d d d c c c c c
d d d c c d d d
d d d c c d c d
d d d d d d d c
Front Eye: 2 dirty blocks
Back Eye: 4 dirty blocks
left Eye: 2 dirty blocks
right Eye: 2 dirty blocks
Press any key to continue . . .
```

Question # 4: [7+ 3= 10 Marks]

A positive integer is called an Armstrong number, if the sum of cubes of individual digits of number is equal to that number itself. For example

Number **153** is Armstrong because sum of cubes of individual digits **(1 * 1 * 1 + 5 * 5 * 5 + 3 * 3 * 3)** = **153**
Number **12** is not Armstrong because sum of cubes of individual digits **(1 * 1 * 1 + 2 * 2 * 2)** = **9**

a. Write a C++ function that takes an integer as parameter and finds whether the number is Armstrong or not and return true or false accordingly.

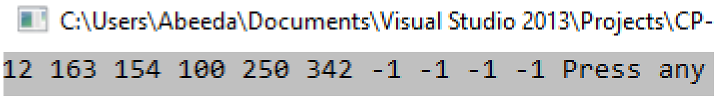
```
bool isArmstrong(int n){
      if (n == 0 || n==1) return true;
      else{
            int num=n, r, sum= 0;
            while (num > 0){
                  r = num % 10;
                  num = num / 10;
                  sum = sum + (r*r*r);
            }
            if (sum == n)
                  return true;
            else
                  return false;
      }
   }
```

b. Write a C++ function which takes 1-D array of integers and its size as input and it removes all of the Armstrong numbers from this array. (May require shifting)

```
void removeArmstrong(int arr[], int size){
	for (int i = 0; i < size && arr[i] !=-1; i++){
		if (isArmstrong(arr[i])){
			for (int j = i; j < size-1&& arr[j]!=-1; j++){
				arr[j] = arr[j + 1];
			}
			arr[size - 1] = -1;
			i--;
		}
	}
}

int main(){
	int arr[10] = { 12, 153, 371, 163, 154, 100, 250, 370, 342, 407 };
	removeArmstrong(arr, 10);
	for (int i = 0; i < 10; i++){
		cout << arr[i]<<" ";
	}
	system("Pause");
	return 0;

}
```



```
C:\Users\Abeeda\Documents\Visual Studio 2013\Projects\CP-

12 163 154 100 250 342 -1 -1 -1 -1 Press any
```

Question # 5: [10+5+15= 30 Marks]

A diagonal-constant matrix, is a square matrix in which each descending diagonal from left to right is constant with the same value. For instance, the following matrix is a diagonal-constant matrix:

A diagonal-constant Matrix

A	B	C	D	E
F	A	B	C	D
G	F	A	B	C
H	G	F	A	B
I	H	G	F	A

Not a diagonal-constant Matrix

A	B	C	D	E
F	A	B	K	D
G	F	A	B	C
H	G	F	A	B
I	H	G	F	A

a. Write a C++ function that takes, as parameters a 2D matrix and its dimensions and determines, whether it is a diagonal-constant matrix or not by returning true or false.

```
bool isDiagonal(char arr[][5], int size)
{
	for (int i = 0; i < size - 1; i++){
		if (arr[i][i] != arr[i + 1][i + 1])
			return false;
	}

	for (int k = 1; k < size; k++){
		for (int i = 0, j = k; i < (size-k)-1; i++, j++)
		{
			if (arr[i][j] != arr[i + 1][j + 1])
				return false;
		}
		for (int i = k, j = 0; j <(size-k)-1; i++, j++)
		{
			if (arr[i][j] != arr[i + 1][j + 1])
				return false;
		}
	}
	return true;

}
```

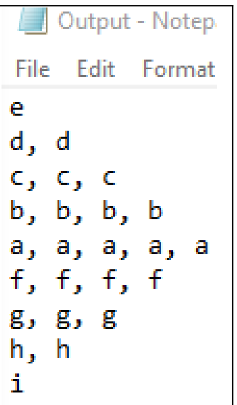
b. Write another C++ function that takes as parameters a 2D matrix and its dimensions and it writes its data in a file “Output.txt”, if it is a diagonal-constant matrix, output format is given below.

```
void output(char arr[][5], int size){
        if (isDiagonal(arr, 5)){
                ofstream fout;
                fout.open("Output2.txt");
                for (int k = size-1; k>=1; k--){
                        for (int i = 0, j = k; j < size; i++, j++)
                        {
                                fout << arr[i][j] << ",";
                        }
                        fout << endl;
                }

for (int i = 0; i < size; i++){
                        fout << arr[i][i] << ",";
                }
                fout << endl;
                for (int k = 1; k < size; k++){
                        for (int i = k, j = 0; j <(size - k); i++, j++)
                        {
                                fout << arr[i][j] << ",";
                        }
                        fout << endl;
                }
                fout.close();
        }


}

//other Solution
int count = 2;
                for (int k = 1; k <=size; k++)
                {
                        for (int j = size - k, i = 1; i < count; i++)
                        {
                                fout << arr[0][j];
                                if (i != count - 1) fout << ",";
                        }
                        fout << endl;
                        count++;
                }
                count = size - 1;
                for (int k = size-1; k>0; k--)
                {
                        for (int j = size-k, i = 1; i < count; i++)
                        {
                                fout << arr[j][0];
                                if (i != count - 1) fout << ",";
                        }
                        fout << endl;
                        count--;
                }
```



```
e
d, d
c, c, c
b, b, b, b
a, a, a, a, a
f, f, f, f
g, g, g
h, h
i
```

c. Write a C++ function that takes as parameters a 2D matrix and its dimensions and rotates the matrix in anti-clock wise direction, if it is a constant diagonal matrix.

A diagonal-constant Matrix

A	B	C	D	E
F	A	B	C	D
G	F	A	B	C
H	G	F	A	B
I	H	G	F	A

After Rotation

B	C	D	E	D
A	A	B	C	C
F	F	A	B	B
G	G	F	A	A
H	I	H	G	F

```
void rotate(char arr[][5], int size){

      if (isDiagonal(arr, size)){
            int temp = arr[0][0];
            //shift top row
            for (int j = 0; j < size - 1; j++)
                  arr[0][j] = arr[0][j + 1];
            arr[0][size - 1] = arr[1][size - 1];

            //shift last column
            for (int i = 1; i < size - 1; i++)
                  arr[i][size - 1] = arr[i + 1][size - 1];
            arr[size - 1][size - 1] = arr[size - 1][size - 2];

            //shift last row
            for (int j = size - 2; j>0; j--)
                  arr[size-1][j] = arr[size-1][j-1];
            arr[size-1][0] = arr[size-2][0];

            //shift first column
            for (int i = size - 2; i>0; i--)
                  arr[i][0] = arr[i-1][0];
            arr[1][0] = temp;
      }

}
```