1. Write a program which prints the letters in a char array in reverse order. For example, if the array contains {'c', 's', 'c', '2', '6', '1'} the output (to the terminal) should be "162csc".

```c
void printReverse(const char car[], int len);
```

2. What is the output of the code segment below?

```c
int main()
{
    int nums[9] = {13, 11, 15, 9, 7, 5, 8, 3, 1};
    int n = mys(nums, 9);
    cout << n << endl; // (*)
    ....
}

int mys(int array[], int len)
{
    int n = 1;
    for (int i = 1; i < len; i++)
    {
        if (array[i] < array[i - 1])
            n++;
        else
            n = 1;
    return n;
}
```

3. Write a function `maxArray` which receives an array of double's and returns the maximum value in the array. You can assume the array contains at least one element. The prototype would be

```c
double maxArray(double dar[], int size);
```

4. Write a function `CountDigit` which receives two arguments: a char array and the size of the array (of type int). This function counts the number of digit letters in the char array, and returns the count (of type int). Use the C++ predefined function `isdigit` (defined in `<cctype>`).

5. Write a piece of code which calls `CountDigit` to find out how many digit characters are in arr.

```c
int main()
{
    char arr[5] = {'y', '2', '0', '0', '4'};
    int no_digit;

    // continue here
```
6. Write a function `equalsIgnoreCase`, which receives two char arrays and their sizes, and returns true if the two char arrays contain the same characters irrespective of the case. For example, for character arrays {'a', 'B', 'c'} and {'A', 'b', 'c'}, the function returns true, but for {'a', 'B', 'c'} and {'a', 'B'}, or {'a', 'B', 'c'} and {'X', 'b', 'z'}, the function returns false. The prototype would be:

```c
bool equalsIgnoreCase(const char ar1[], int size1, const char ar2[], int size2);
```

7. (*) Given an unsorted array whose elements are all 0 (zeros) or 1 (ones), write code to sort the array so that all the 0's appear first, followed by all the 1's. Your solution must be faster than any general purpose sorting algorithm. You may assume that `num_elements` always accurately indicates the number of elements in the array. Please use the following function header:

```c
// Sorts an array with 0's and 1's so that all the 0's appear first // in the array and all the 1's at the end of the array. void sort_elements(int array[], int num_elements);
```