MILESTONE 1

1. https://leetcode.com/problems/jewels-and-stones/

```
class Solution {
    public int numJewelsInStones(String jewels, String stones) {
        int c=0;
        for(int i=0;i<jewels.length();i++)
        {
            for(int j=0;j<stones.length();j++)
            {
                if(stones.charAt(j)==jewels.charAt(i))
                    c++;
            }
        }
        return c;
    }
}
```

2. https://leetcode.com/problems/merge-strings-alternately/

```
class Solution {
    public String mergeAlternately(String word1, String word2) {
        String merged="";
        int l1,l2;
        l1=word1.length();
        l2=word2.length();
        if(l1<l2){
            for(int i=0;i<l1;i++)
            {
                merged=merged+word1.charAt(i)+word2.charAt(i);
            }
            merged=merged.concat(word2.substring(l1));
        }
        else{
            for(int i=0;i<l2;i++)
            {
                merged=merged+word1.charAt(i)+word2.charAt(i);
            }
            merged=merged.concat(word1.substring(l2));
        }
        return merged;
    }}
```

3. https://leetcode.com/problems/minimum-number-of-steps-to-make-two-strings-anagram/

```
class Solution {
    public int minSteps(String s, String t) {
        int freq[]=new int[26];
        int res=0;
        for(int i=0;i<s.length();i++)
        {
            freq[s.charAt(i)-'a']++;
            freq[t.charAt(i)-'a']--;
        }
        for(int i=0;i<26;i++)
        {
            if(freq[i]>0)
                res=res+freq[i];
        }
        return res;
    }
}
```

4. https://leetcode.com/problems/spiral-matrix/

```
class Solution {
    public List<Integer> spiralOrder(int[][] matrix) {
        List<Integer> ans=new ArrayList<Integer>();
        int m,n;
        m=matrix.length;
        n=matrix[0].length;
        int dir=0;
        int t,d,l,r;//taking four pointers
        t=0;
        d=m-1;
        l=0;
        r=n-1;
        while(t<=d && l<=r)
        {
            if(dir==0){
            for(int i=l;i<=r;i++)
                    ans.add(matrix[t][i]);
            t++;}
            else if(dir==1){
            for(int i=t;i<=d;i++)
                    ans.add(matrix[i][r]);
            r--;}
            else if(dir==2){
            for(int i=r;i>=l;i--)
                    ans.add(matrix[d][i]);
            d--;}
            else if(dir==3){
            for(int i=d;i>=t;i--)
                    ans.add(matrix[i][l]);
            l++;}
            dir=(dir+1)%4;
        }
        return(ans);
    }
}
```

5. https://leetcode.com/problems/sort-array-by-parity/

```
class Solution {
    public int[] sortArrayByParity(int[] nums) {
        int[] ans=new int[nums.length];
        int k=0;
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]%2==0)
            {
                ans[k]=nums[i];
                k++;
            }
        }
        for(int i=0;i<nums.length;i++)
        {
            if(nums[i]%2!=0)
            {
                ans[k]=nums[i];
                k++;
            }
        }
        return ans;
    }}
```

6. https://leetcode.com/problems/best-time-to-buy-and-sell-stock/

```
class Solution {
    public int maxProfit(int[] prices) {
        int max_profit=0;
        int min=Integer.MAX_VALUE;
        for(int i=0;i<prices.length;i++)
        {
            if(prices[i]<min)
                min=prices[i];
            else if(prices[i]-min>max_profit)
                max_profit=prices[i]-min;
        }
        return max_profit;
    }
}
```

7. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-ii

```
class Solution {
    public int maxProfit(int[] prices) {
        int max_profit=0;
        for(int i=1;i<prices.length;i++)
        {
            if(prices[i]>prices[i-1])
                max_profit=max_profit+(prices[i]-prices[i-1]);
        }
        return max_profit;
    }
}
```