Two Pointers

- Reverse string

```
class Solution {
    public void reverseString(char[] s) {

        int left=0,right=s.length-1;

        while(left<=right)
        {
            char temp=s[left];
            s[left]=s[right];
            s[right]=temp;

            left++;
            right--;
        }

        System.out.println(s);
    }
}
```

- Valid palindrome

```
class Solution {
    public boolean isPalindrome(String s) {
```

```
        String wd="";

        int l=0,r=s.length()-1;

        // Removing all non alpha numeric characters from the
input string
        while(l<=r)
        {
            char ch=s.charAt(l);
            if(Character.isLetterOrDigit(ch))
                wd+=ch;
            l++;
        }

        s=""+wd;
        wd="";

        l=0;
        r=s.length()-1;

        while(l<=r)
        {
            char ch=s.charAt(r);
            wd+=ch;
            r--;
        }

        wd= wd.toLowerCase();
        s=s.toLowerCase();

        return (wd.equals(s));
    }
}
```

Or

```
class Solution {
    public boolean isPalindrome(String s) {

        String wd="";

        int l=0,r=s.length()-1;

        // Removing all non alpha numeric characters from the
input string
        while(l<=r)
        {
            char ch=s.charAt(l);
            if(Character.isLetterOrDigit(ch))
                wd+=Character.toLowerCase(ch);
            l++;
        }

        l=0;
        r=wd.length()-1;

        while(l<=r)
        {
            if(wd.charAt(l)!=wd.charAt(r))
                return false;

            l++;
            r--;
```

```
        }


        return true;
    }
}
```

- Valid palindrome II

```
class Solution {
    public boolean validPalindrome(String s) {
        int c=0,l=0,r=s.length()-1;

        while(l<=r)
        {
            // Try skipping either left or right char
            if(s.charAt(l)!=s.charAt(r))
                return ispalin(s,l+1,r) || ispalin(s,l,r-1);

            l++;
            r--;
        }

        return true;
    }

    private boolean ispalin(String s,int l,int r)
    {
        while(l<=r)
```

```
        {
            if(s.charAt(l)!=s.charAt(r))
                return false;

            l++;
            r--;
        }
        return true;
    }
}
```

- Merge strings alternately

```
class Solution {
    public String mergeAlternately(String word1, String word2) {

        String str="";
        int i=0,j=0,idx=0;

        while(i<word1.length() && j<word2.length())
        {
            if(idx%2==0)
                str+=word1.charAt(i++);
            else
                str+=word2.charAt(j++);

            idx++;
        }
```

```
        // if word1 is remaining
        while(i<word1.length())
            str+=word1.charAt(i++);

        // if word2 is remaining
        while(j<word2.length())
            str+=word2.charAt(j++);

        return str;
    }
}
```

Or

```
public class Solution {
    public String mergeAlternately(String word1, String word2) {
        StringBuilder res = new StringBuilder();
        int i = 0, j = 0;
        while (i < word1.length() && j < word2.length()) {
            res.append(word1.charAt(i++));
            res.append(word2.charAt(j++));
        }
        res.append(word1.substring(i));
        res.append(word2.substring(j));
        return res.toString();
    }
}
```

Or

```
public class Solution {
    public String mergeAlternately(String word1, String word2) {
```

```
        int n = word1.length(), m = word2.length();
        StringBuilder res = new StringBuilder();
        int i = 0, j = 0;
        while (i < n || j < m) {
            if (i < n) res.append(word1.charAt(i++));
            if (j < m) res.append(word2.charAt(j++));
        }
        return res.toString();
    }
}
```

-