Problem4A.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P4- Valid Palindrome
//Check whether given string is Palindrome or Not.

public class Problem4A {

		static boolean isPalindrome(String str) {

			StringBuilder sb=new StringBuilder(str); //O(1)
			sb.reverse(); // O(n)
			String revStr=sb.toString(); //O(1)

			return str.equals(revStr);// O(n)
		}

		public static void main(String[] args) {

		String str1 = "abcba";
		boolean b1 = isPalindrome(str1);
		System.out.println(b1);

		String str2 = "abc";
		boolean b2 = isPalindrome(str2);
		System.out.println(b2);

	}
}

//Time Complexity - O(n)
//Aux Space Complexity -  O(n)
```

Problem4B.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P4- Valid Palindrome
//Check whether given string is Palindrome or Not.

public class Problem4B {
	static boolean isPalindrome(String str) {
		int n = str.length();
		int start = 0;
		int end = n - 1;

		while (start < end) {
			if (str.charAt(start) != str.charAt(end)) {
				return false;
			}

			start++;
			end--;
		}
		return true;
	}
	public static void main(String[] args) {

		String str1 = "abccba";
		boolean b1 = isPalindrome(str1);
		System.out.println(b1);
	}
}

//Time Complexity - O(n/2) => O(n)
//Aux Space Complexity -  O(1)
```

Problem5A.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
import java.util.Arrays;

//P5- Valid Anagrams
//Check whether Two strings are anagrams are or not
public class Problem5A {
		static boolean isAnagram(String str1,String str2) {

			if(str1.length()!=str2.length())
				return false;

			char charr1[] = str1.toCharArray();
			Arrays.sort(charr1);
			str1 = new String(charr1);

			char charr2[] = str2.toCharArray();
			Arrays.sort(charr2);
			str2 = new String(charr2);

			return str1.equals(str2);
		}
		public static void main(String[] args) {
		String str1 = "sir";
		String str2 = "sri";

		boolean b1 = isAnagram(str1,str2);
		System.out.println(b1);
	}
}
//Time Complexity - O(n log(n))
//Aux Space Complexity -  O(m+n) // O(n)
```

Problem5B.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P5- Valid Anagrams
//Check whether Two strings are anagrams are or not
public class Problem5B {
		static boolean isAnagram(String str1,String str2) {
			if(str1.length()!=str2.length())
				return false;

			int counter[]  = new int[26];

			for(int i=0;i<str1.length();i++) {
				counter[str1.charAt(i) - 97]++;
				counter[str2.charAt(i) - 'a']--;
			}

			for(int i=0;i<counter.length;i++) {
				if(counter[i]!=0) {
					return false;
				}
			}
			return true;
		}
		public static void main(String[] args) {
		String str1 = "abc";
		String str2 = "cba";
		boolean b1 = isAnagram(str1,str2);
		System.out.println(b1);
	}
}
//Time Complexity - O(n+26) / O(n)
//Aux Space Complexity -  O(26) // O(1)
```

Problem6A.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P6- First Repeating Character in the given String
//Consider the String with Lower Alphabets

public class Problem6A {

		static int firstRepeat(String str) {
			int n = str.length();
			for(int i=0;i<n;i++) {
				for(int j=i+1;j<n;j++) {
					if(str.charAt(i) == str.charAt(j)) {
						return i;
					}
				}
			}
			return -1;
		}

		public static void main(String[] args) {

		String str = "abcdcdef";
		int index = firstRepeat(str);
		System.out.println(index);
	}
}

//Time Complexity - O(n^2)
//Aux Space Complexity -   O(1)
```

Problem6B.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P6- First Repeating Character in the given String
//Consider the String with Lower Alphabets

public class Problem6B {

	static int firstRepeat(String str) {
			int counter[]  = new int[26];

			for(int i=0;i<str.length();i++) {
				counter[str.charAt(i) - 'a']++;
			}

			for(int i=0;i<str.length();i++) {
				if(counter[str.charAt(i)-97]>1) {
					return i;
				}
			}
			return -1;
		}

	public static void main(String[] args) {
		String str = "abcdcdef";
		int index = firstRepeat(str);
		System.out.println(index);
	}
}

//Time Complexity - O(2n) => O(n)
//Aux Space Complexity -   O(26) => O(1)
```

Problem7A.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P7- First Unique Character in the given String
//Consider the String with Lower Alphabets

public class Problem7{

	static int firstUnique(String str) {

			int counter[]  = new int[26];

			for(int i=0;i<str.length();i++) {
				counter[str.charAt(i) - 'a']++;
			}

			for(int i=0;i<str.length();i++) {
				if(counter[str.charAt(i)-97]==1) {
					return i;
				}
			}
			return -1;
		}
	public static void main(String[] args) {
		String str = "abcdbac";
		int index = firstUnique(str);
		System.out.println(index);
	}
}

//Time Complexity - O(2n) => O(n)
//Aux Space Complexity -  O(26) => O(1)
```

Problem7B.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P7- First Unique Character in the given String
//Consider the String with AlphaNumeric (A-Z,a-z,0-9):

public class Problem7B{

	static int firstUnique(String str) {

			int counter[]  = new int[75];

			for(int i=0;i<str.length();i++) {
				counter[str.charAt(i) - '0']++;
			}

			for(int i=0;i<str.length();i++) {
				if(counter[str.charAt(i)-48]==1) {
					return i;
				}
			}
			return -1;
		}
	public static void main(String[] args) {

			String str = "1Ab2D1cd5Ab";
			int index = firstUnique(str);
			System.out.println(index);
	}
}
//Time Complexity - O(2n) => O(n)
//Aux Space Complexity -   O(75) => O(1)
```

Problem7C.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P7- First Non-Repeating Character in the given String
//Consider the String with Alphabets (A-Z,a-z):

public class Problem7C{

	static int firstUnique(String str) {

			int counter[]  = new int[58];

			for(int i=0;i<str.length();i++) {
				counter[str.charAt(i) - 'A']++;
			}

			for(int i=0;i<str.length();i++) {
				if(counter[str.charAt(i)-65]==1) {
					return i;
				}
			}
			return -1;
		}
	public static void main(String[] args) {

			String str = "AbDcdAbD";
			int index = firstUnique(str);
			System.out.println(index);
	}
}
//Time Complexity - O(2n) => O(n)
//Aux Space Complexity -   O(58) => O(1)
```

Problem7D.java

```
package com.jlcindia.strings;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
//P7- First Non-Repeating Character in the given String
//Consider the String with Special Symbols, Alphanumerics

public class Problem7D{

	static int firstUnique(String str) {

			int counter[]  = new int[95];

			for(int i=0;i<str.length();i++) {
				counter[str.charAt(i) - ' ']++;
			}

			for(int i=0;i<str.length();i++) {
				if(counter[str.charAt(i)-32]==1) {
					return i;
				}
			}
			return -1;
		}
	public static void main(String[] args) {

			String str = "#A1*2$BcdF#A1    *2BcdF#A1*2BcdF#A1*2BcdF";
			int index = firstUnique(str);
			System.out.println(index);
	}
}
//Time Complexity - O(2n) => O(n)
//Aux Space Complexity -   O(95) => O(1)
```