ARRAY PROGRAMS

1.WAP SORT AN ARRAY IN ASCENDING ORDER USING BUBBLE SORT ✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Array_Ascending_bubbleSort
{
	@Test

	public void Ascending()
	{
		int[]a= {10,5,22,15,13};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]>a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}
			System.out.print(a[i]+",");

	}

}
```

OUTPUT: 5,10,13,15,22

2.WAP SORT AN ARRAY IN DESCENDING ORDER USING BUBBLE SORT ✅

```java
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Array_Descending_bubbleSort
{
	@Test

	public void Ascending()
	{
		int[]a= {20,12,18,10,8};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]<a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}
			System.out.print(a[i]+",");
		}

	}

}
```

OUTPUT: **20,18,12,10,8**

3.WAP FOR THE COMBINATION OF NUMBERS ✅

```java
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Combination_of_Numbers
{
	@Test

	public void Combination()
	{
		int[]a= {8,2,3,7,6,4,9,5};
		int sum=11;

		for(int i=0; i<a.length; i++)
		{
			int c=sum-a[i];

		   System.out.println(a[i]+""+"+"+c+"="+sum);

		}

	}



}
```

OUTPUT:

```
8+3=11
2+9=11
3+8=11
7+4=11
6+5=11
4+7=11
9+2=11
5+6=11
```

4.WAP TO FIND THE FREQUENTLY REPEATED NUMBER IN ARRAY ✅

```java
package Array_Practice_Program;

import org.testng.annotations.Test;


public class FindFrequentlyRepeatedNumberInArray
{
	@Test

	public void frequentlyRepeated()
	{
		int[]a= {1,2,4,2,6,2,8,10,2};
		int max=0;
		int value=0;

		for(int i=0; i<a.length; i++)
		{
			int count=1;
			for(int j=i+1; j<a.length;j++)
			{
				if(a[i]==a[j])
				{
					count++;
				}

			}
			if(count>max)
			{
				max=count;
				value=a[i];
			}


		}
		System.out.println(value +" is repeating "+max +" times");
	}

}
```

OUTPUT: **2 is repeating 4 times**

5.WAP TO FIND THE MAXIMUM NUMBER IN ARRAY WITHOUT USING BUBBLE SORT ✅

```
package Array_Practice_Program;


import org.testng.annotations.Test;


public class First_MaximumNoArray_WithoutusingBuublesort
{
	@Test

	public void Maximum()
	{
		int[]a= {2,5,6,8,1};
		int max=a[0];

		for(int i=0; i<a.length;i++)


		{
			if(max<a[i])
			{
				max=a[i];
			}
		}
		System.out.println(max);
	}

}
```

OUTPUT:8

6.WAP TO FIND THE MINIMUM NUMBER IN ARRAY WITHOUT USING BUUBLE SORT ✅

```
package Array_Practice_Program;


import org.testng.annotations.Test;


public class First_MinimumNoArray_WithoutusingBuublesort
{
	@Test

	public void Minimum()
	{
		int[]a= {2,5,6,8,1};
		int min=a[0];

		for(int i=0; i<a.length;i++)
		{

			if(min>a[i])
			{
				min=a[i];
			}
		}
		System.out.println(min);
	}
}
```

OUTPUT:1

7.WAP TO FIND THE MAXIMUM AND MINIMUM NUMBER IN ARRAY WITHOUT USING BUUBLE SORT ✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;

public class Maximum_Minimum_Number_In_Array
{
	@Test

	public void Maximum()
	{
		int[]a= {2,5,6,8,1};
		int max=a[0];
		int min=a[0];

		for(int i=0; i<a.length;i++)


		{
			if(max<a[i])
			{
				max=a[i];
			}
			else
			{
				min=a[i];
			}
		}
		System.out.println(max);
		System.out.println(min);
	}

}
```

OUTPUT:8,1

8.WAP TO FIND THE MAXIMUM NUMBER IN ARRAY USING BUBBLE SORT✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class MaximumNumber_InArray
{
	@Test

	public void Maximum()
	{
		int[]a= {0,5,55,58,23,14,};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]<a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}

		System.out.println(a[0]);

	}
}
```

OUTPUT:58

9.WAP TO FIND THE MINIMUM NUMBER IN ARRAY USING BUBBLE SORT
✅

```
import org.testng.annotations.Test;


public class Minimum_number_in_Array
{
	@Test

	public void Minimum()
	{
		int[]a= {10,5,22,15,13};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]>a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}
		System.out.println(a[0]);

	}
}
```

OUTPUT:5

10.WAP TO FIND THE MISSING NUMBER IN ARRAY BETWEEN 1 TO 9 ✅

```
package Array_Practice_Program;


import org.testng.annotations.Test;


public class Missing_No_InArray_Between_1_To_9
{
	@Test

	public void Missing ()
	{
		int[]a= {1,3,6,7,9};

			for (int i = 1; i <=9; i++)
			{
				int count = 0;
				for (int j = 0; j < a.length; j++)
				{
					if (a[j] == i)
					{
						count = 1;
						break;
					}

				}
				if (count == 0)
				{
					System.out.println(i);
				}

			}
	}
}
```

OUTPUT:2,4,5,8

11.WAP TO FIND THE MULTIPLICATION OF MAXIMUM 3 NOS IN ARRAY✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Multiplication_Of_Maximum3_No_InArray
{
	@Test

	public void MultiplicationofMaximum3NoinArray()
	{
		int[]a= {0,5,2,6,8};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]<a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}

		 int product=1;
		 for (int i=0; i<3; i++)
		 {
			  product=product*a[i];
		 }
		 System.out.println(product);
	}

}
```

OUTPUT:240

12.WAP TO FIND THE MULTIPLICATION OF MINIMUM 3 NOS IN ARRAY✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Multiplication_Of_Minimum3_No_InArray
{
	@Test

	public void MultiplicationofMinimum3NoinArray()
	{
		int[]a= {1,5,2,6,8};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]>a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}

		 int product=1;
		 for(int i=0; i<3; i++)
		 {
			  product=product*a[i];
		 }
		 System.out.println(product);
	}

}
```

OUTPUT:10

13.WAP TO FIND THE SUM OF MAXIMUM 3 NOS IN ARRAY ✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Sum_Of_Maximum3_No_InArray
{
	@Test

	public void sumofMaximum3NoinArray()
	{
		int[]a= {0,5,2,6,8};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]<a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}

		 int sum=0;
		 for(int i=0; i<3; i++)
		 {
			  sum=sum+a[i];
		 }
		 System.out.println(sum);
	}

}
```

OUTPUT:19

14.WAP TO FIND THE SUM OF MINIMUM 3 NOS IN ARRAY✅

```
package Array_Practice_Program;

import org.testng.annotations.Test;


public class Sum_Of_Minimum3_No_InArray
{
	@Test

	public void sumofMinimum3NoinArray ()
	{
		int[]a= {0,5,2,6,8};

		for(int i=0; i<a.length;i++)
		{
			for(int j=i+1; j<a.length; j++)
			{
				if(a[i]>a[j])
				{
					int temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}

		}

		 int sum=0;
		 for(int i=0; i<3; i++)
		 {
			 sum=sum+a[i];
		 }
		 System.out.println(sum);
	}

}
```

OUTPUT:7

15.WAP TO SWAP 2 NOS WITH USING 3RD VARIABLE ✅

```java
package Array_Practice_Program;
import org.testng.annotations.Test;


public class Swap2_Nos_Using_3rd_Variable
{
	@Test

	public void swap2number()
	{
		int a=40;
		int b=80;
		int c;
		c=a;
		a=b;
		b=c;
		System.out.println(a);
		System.out.println(b);
	}
}
```

OUTPUT:80,40

16.WAP TO SWAP 2 NOS WITHOUT USING 3RD VARIABLE ✅

```
package Array_Practice_Program;
import org.testng.annotations.Test;


public class Swap2Nos_WithoutUsing_3rdVariable
{
	@Test

	public void swap2numberwithoutusing3rdvariable()
	{
		int a=80;
		int b=40;
		System.out.println("Before swapping a=" +a +"  "+ "b="+b);

		a=a+b;
		b=a-b;
		a=a-b;
		System.out.println("After swapping a=" +a +"  "+ "b ="+b);

	}

}
```

OUTPUT:

```
Before swapping a=80 b=40
After swapping a=40 b =80
```

MOCK QUESTIONS

17.WAP FOR THE ADDITION OF ARRAY ✅

```
package MOCK_TEST;

public class Array_Addition_Using_If_Else
{
	public static void main(String[] args)
	{
		int[]a= {1,2,3};
		int[]b= {2,4,6,8};

		int[] l;
		int[] s;

		if(a.length > b.length)
		{
			l=a;
			s=b;
		}
		else
		{
			l=b;
			s=a;
		}

		for(int i=0; i< s.length; i++)
		{
			System.out.println(l[i]+s[i] + " ");
		}

		for(int i=s.length; i<l.length; i++)
		{
			System.out.println(l[i] + " ");
		}

	}

}
```

OUTPUT:3,6,9,8

18.WAP FOR THE ADDITION OF ARRAY USING TRY CATCH BLOCK

```
package MOCK_TEST;

public class Array_Addition1_Using_Try_Catch
{
	public static void main(String[] args)
	{
		int[]a= {4,5,6,2,3};
		int[]b= {2,3,3,4};

		int length=a.length;

		if(a.length<b.length)
		{
			length=b.length;
		}

		for(int i=0; i<length; i++)
		{
			try
			{
				System.out.println(a[i]+b[i]);
			}

			catch (Exception e)
			{
				if(a.length<b.length)
				{
					System.out.println(b[i]);
				}

				else
				{
					System.out.println(a[i]);
				}
			}
		}

	}

}
```

OUTPUT:6,8,9,6,3

19.WAP FOR THE FIRST MAXIMUM AND SECOND MINIMUM, FIRST MINIMUM AND SECOND MAXIMUM

```
public class First_Min_And_Second_Min_And_First_Max_And_Second_Max
{
	@Test

	public void test()
	{
		int a[]= {4,5,6,2,3};
		int firstMax=a[0];
		int secondMax=a[0];
		int firstMin=a[0];
		int secondMin=a[0];

		for (int i = 0; i < a.length; i++)
	   {
		    if (a[i] > firstMax)
		    {
		        secondMax = firstMax;
		        firstMax = a[i];
		    }
		    else if (a[i] > secondMax && a[i] < firstMax)
		    {
		        secondMax = a[i];
		    }
		    if (a[i] < firstMin)
		    {
		        secondMin = firstMin;
		        firstMin = a[i];
		    }
		    else if (a[i] < secondMin && a[i] > firstMin)
		    {
		        secondMin = a[i];
		    }
	   }
	   System.out.println("First Maximum:" + firstMax);
	   System.out.println("Second Maximum:" + secondMax);
	   System.out.println("First Minimum:" + firstMin);
	   System.out.println("Second Minimum:" + secondMin);
	}
}
```

OUTPUT:

```
First Maximum:6
Second Maximum:5
First Minimum:2
Second Minimum:3
```

20.WAP FOR THE MAXIMUM AND MINIMUM NUMBER IN ARRAY ✅

```
package MOCK_TEST;

import org.testng.annotations.Test;


public class Maximum_Minimum_Number_In_Array
{
	@Test

	public void Maximum()
	{
		int[]a= {4,5,3,2,7,1};
		int max=a[0]; //4  //5   //5  //5  //7  //7
		int min=a[0]; //4  //4   //3  //2  //2 //1

		for(int i=0; i<a.length;i++)


		{
			if(max<a[i])
			{
				max=a[i];
			}
			else if (min>a[i])
			{
				min=a[i];
			}
		}
		System.out.println(max);
		System.out.println(min);
	}

}
```

OUTPUT:7,1

21.WAP TO FIND THE OCCURANCE OF EACH CHARACTER IN STRING ✅

```
package MOCK_TEST;

import org.testng.annotations.Test;

public class Occurance_oF_Each_Character_In_String
{
	@Test
	public void Occurance()
	{
		String s1="Welcome";
		String s = s1.toLowerCase();

		for(int i=0; i<s.length(); i++)
		{
			int count=0;
		for(int j=0; j<s.length(); j++)


		{
			if(s.charAt(i)==s.charAt(j))
			{
				if(i>j)
				{
					break;
				}

				else
				{
					count++;
				}
			}
		}

				if(count>=1)
				{
					System.out.print(s.charAt(i) +"="+count +" "+"");
				}

		}
	}

	}
```

OUTPUT: w=1 e=2 l=1 c=1 o=1 m=1

22.WAP TO REVERSE A STRING AT SAME POSITION ✅

```
package MOCK_TEST;

public class ReverseString
{
public static void main(String[] args)
{
      String s="my name is kiran";
      String[] st = s.split(" ");

      for(int i=0; i<st.length; i++)
      {
            String str=st[i];

            for(int j=str.length()-1; j>=0; j--)
            {
                  System.out.print(str.charAt(j));
            }
            System.out.print(" ");
      }
   }
}
```

OUTPUT: ym eman si narik

23.

INPUT IS I LOVE INDIA

OUTPUT IS A IDNI EVOLI

```
package MOCK_TEST;
import org.testng.annotations.Test;


public class String1
{
	@Test

	public void test()
	{
		String s="I LOVE INDIA";
		String s1=s.replaceAll(" ",""); // ILOVEINDIA
		int count=s1.length()-1;
		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)!=' ')
			{
				System.out.print(s1.charAt(count--));
			}
			else
			{
				System.out.print(s.charAt(i));
			}
		}
	}

}
```

OUTPUT: A IDNI EVOLI

24.WAP TO PRINT UNIQUE OR REMOVE DUPLICATE CHARACTER IN A STRING

```
package MOCK_TEST;


import org.apache.commons.collections4.map.HashedMap;
import org.testng.annotations.Test;

public class Unique_RemoveDuplicate_Character_In_String_Using_HashMap
{
	@Test
	public void Occurance()
	{
		String s="Welcome";
	    String s1 = s.toLowerCase();
 HashMap<Character, Integer> map = new HashMap<Character, Integer>();
	    for(int i=0; i<s1.length();i++)
	    {
	     if(map.containsKey(s1.charAt(i)))
	     {
			map.put(s1.charAt(i), map.get(s1.charAt(i))+1);

	     }
	     else
	     {
			map.put(s1.charAt(i), 1);
	     }
	    }
	    for(Entry<Character, Integer> m:map.entrySet())
	    {
	     if(m.getValue()==1)
	     {
			System.out.print(m.getKey()+"="+m.getValue()+" "+"");
	     }
	    }
	}

}
```

OUTPUT: c=1 w=1 l=1 m=1 o=1

25.WAP TO PRINT UNIQUE OR REMOVE DUPLICATE CHARACTER IN A INPUT STRING ✅

Input is **Selenium**
OUTPUT IS **Selnium**

```
import org.testng.annotations.Test;

public class Unique_RemoveDuplicate_Character_In_String
{
	@Test
	public void Unique()
	{
		String s1=”Selenium”;
		String s = s1.toLowerCase();
		String rem="";

		for(int i=0; i<s.length(); i++)
		{
			int count=0;
		for(int j=0; j<s.length(); j++)
		{
			if(s.charAt(i)==s.charAt(j))
			{
				if(i>j)
				{
					break;
				}

				else
				{
					count++;
				}
			}
		}

			if(count>=1)
			{
				System.out.print(s.charAt(i)+””);
			}
		}
	}

	}
```

OUTPUT: selnium

STRING PROGRAMS

26.WAP TO PRINT UNIQUE CHARACTER IN STRING ✅

```
package string_program;

import org.testng.annotations.Test;

public class Unique_RemoveDuplicate_Character_In_String
{
	@Test
	public void Unique()
	{
		String s1="Tester";
		String s = s1.toLowerCase();

		for(int i=0; i<s.length(); i++)
		{
			int count=0;
		for(int j=0; j<s.length(); j++)
		{
			if(s.charAt(i)==s.charAt(j))
			{
				if(i>j)
				{
					break;
				}

				else
				{
					count++;
				}
			}
		}

			if(count==1)
			{
				System.out.print(s.charAt(i)+"="+count+" "+"");
			}
		}
	}
	}
```

OUTPUT: s=1 r=1

27.WAP TO REMOVE UNIQUE OR REMOVE DUPLICATE CHARACTER IN STRING USING LINKEDHASHSET ✅

```
package string_program;

import java.util.LinkedHashSet;

import org.testng.annotations.Test;

public class
Unique_RemoveDuplicate_Character_In_String_Using_LinkedHashSet
{
	@Test
	public void test()
	{
		String s="Tester";
		String s1=s.toLowerCase();
	LinkedHashSet<Character> set = new LinkedHashSet<Character>();

		for(int i=0; i<s1.length(); i++)
		{
			set.add(s1.charAt(i));
		}

		for(Character ch:set)
		{
			int count =0;
			for(int i=0; i<s1.length(); i++)
			{
				if(ch==s1.charAt(i))

				{
					count++;
				}
			}

			if(count==1)
			{
			System.out.println(ch+"="+count);
			}

		}
	}

}
```

OUTPUT: **s=1 r=1**

28.WAP TO REMOVE UNIQUE OR REMOVE DUPLICATE CHARACTER IN STRING USING MAP

```
package string_program;

import java.util.HashMap;
import java.util.Map.Entry;

import org.apache.commons.collections4.map.HashedMap;
import org.testng.annotations.Test;

public class Unique_RemoveDuplicate_Character_In_String_Using_HashMap
{
      @Test
      public void Occurance()
      {
          String s="Tester";
          String s1 = s.toLowerCase();
 HashMap<Character, Integer> map = new HashMap<Character, Integer>();
          for(int i=0; i<s1.length();i++)
          {
            if(map.containsKey(s1.charAt(i)))
            {
                  map.put(s1.charAt(i), map.get(s1.charAt(i))+1);

            }
            else
            {
                  map.put(s1.charAt(i), 1);
            }
          }
          for(Entry<Character, Integer> m:map.entrySet())
          {
            if(m.getValue()==1)
            {
                  System.out.println(m.getKey()+" "+m.getValue());
            }
          }
      }

}
```

OUTPUT: **s=1 r=1**

29.WAP TO PRINT OCCURANCE OF EACH CHARACTER IN STRING ✅

```java
import org.testng.annotations.Test;

public class Occurance_oF_Each_Character_In_String
{
	@Test
	public void Occurance()
	{
		String s1="Tester";
		String s = s1.toLowerCase();

		for(int i=0; i<s.length(); i++)
		{
			int count=0;
		for(int j=0; j<s.length(); j++)


		{
			if(s.charAt(i)==s.charAt(j))
			{
				if(i>j)
				{
					break;
				}

				else
				{
					count++;
				}
			}
		}

			if(count>=1)
			{
				System.out.println(s.charAt(i)+""+count);

			}
		}
	}

	}
```

OUTPUT: t=2 e=2 s=1 r=1

30.WAP TO PRINT OCCURANCE OF EACH CHARACTER IN STRING USING LINKEDHASHSET ✅

```
import java.util.LinkedHashSet;

import org.testng.annotations.Test;

public class Occurance_Of_EachCharacter_Using_LinkedHashSet
{
	@Test
	public void t()
	{
		String s="Tester";
		String s1=s.toLowerCase();
	LinkedHashSet<Character> set = new LinkedHashSet<Character>();

		for(int i=0; i<s1.length(); i++)
		{
			set.add(s1.charAt(i));
		}

		for(Character ch:set)
		{
			int count =0;
			for(int i=0; i<s1.length(); i++)
			{
				if(ch==s1.charAt(i))
				{
					count++;
				}
			}

			if(count>=1)
			{
			System.out.println(ch+" "+count);
			}

		}
	}

}
```

OUTPUT: **t=2 e=2 s=1 r=1**

31.WAP TO OCCURANCE OF EACH CHARACTER IN STRING USING USING MAP

```
package string_program;

import java.util.HashMap;
import java.util.Map.Entry;

import org.apache.commons.collections4.map.HashedMap;
import org.testng.annotations.Test;

public class Occurance_Of_EachCharacter_Using_HashMap
{
      @Test
      public void Occurance()
      {
            String s="Tester";
          String s1 = s.toLowerCase();
HashMap<Character, Integer> map = new HashMap<Character, Integer>();
          for(int i=0; i<s1.length();i++)
          {
            if(map.containsKey(s1.charAt(i)))
            {
                  map.put(s1.charAt(i), map.get(s1.charAt(i))+1);

            }
            else
            {
                  map.put(s1.charAt(i), 1);
            }
          }
          for(Entry<Character, Integer> m:map.entrySet())
          {
            if(m.getValue()>=1)
            {
                  System.out.println(m.getKey()+" "+m.getValue());
            }
          }
      }

}
```

OUTPUT: **t=2 e=2 s=1 r=1**

32.WAP TO PRINT ONLY DUPLICATE CHARACTER IN STRING ✅

```
import org.testng.annotations.Test;

public class Occurance_oF_Each_Character_In_String
{
	@Test
	public void Occurance()
	{
		String s1="Tester";
		String s = s1.toLowerCase();

		for(int i=0; i<s.length(); i++)
		{
			int count=0;
		for(int j=0; j<s.length(); j++)


		{
			if(s.charAt(i)==s.charAt(j))
			{
				if(i>j)
				{
					break;
				}

				else
				{
					count++;
				}
			}
		}
			if(count>1)
			{
				System.out.println(s.charAt(i)+""+count);

			}
		}
	}

	}
```

OUTPUT:t=2
e=2

33.WAP TO PRINT ONLY DUPLICATE CHARACTER IN STRING USING LINKEDHASHSET ✅

```
import java.util.LinkedHashSet;

import org.testng.annotations.Test;

public class
Print_Only_Duplicate_Character_InString_Using_LinkedHashSet
{
	@Test
	public void t()
	{
		String s="Tester";
		String s1=s.toLowerCase();
	LinkedHashSet<Character> set = new LinkedHashSet<Character>();

		for(int i=0; i<s1.length(); i++)
		{
			set.add(s1.charAt(i));
		}

		for(Character ch:set)
		{
			int count =0;
			for(int i=0; i<s1.length(); i++)
			{
				if(ch==s1.charAt(i))
				{
					count++;
				}
			}

			if(count>1)
			{
			System.out.println(ch+"="+count);
			}

		}
	}

}
```

OUTPUT:t=2
e=2

34.WAP TO PRINT ONLY DUPLICATE CHARACTER IN STRING USING MAP

```
package string_program;

import java.util.HashMap;

import java.util.Map.Entry;

import org.apache.commons.collections4.map.HashedMap;
import org.testng.annotations.Test;


public class Print_Only_Duplicate_Character_InString_Using_HashMap
{
	@Test

	public void Occurance()
	{
		String s="Tester";
	    String s1 = s.toLowerCase();
HashMap<Character, Integer> map = new HashMap<Character, Integer>();
	    for(int i=0; i<s1.length();i++)
	    {
	      if(map.containsKey(s1.charAt(i)))
	      {
			map.put(s1.charAt(i), map.get(s1.charAt(i))+1);

	      }
	      else
	      {
			map.put(s1.charAt(i), 1);
	      }
	    }
	    for(Entry<Character, Integer> m:map.entrySet())
	    {
	      if(m.getValue()>1)
	      {
			System.out.println(m.getKey()+"="+m.getValue());
	      }
	    }
	}

}
```

OUTPUT:t=2
e=2

ASSIGNMENT

35.WAP TO REMOVE REPEATED(DUPLICATE) OR UNIQUE NUMBERS IN GIVEN ARRAY USING LINKEDHASHSET ✅

```
import java.util.LinkedHashSet;
import org.testng.annotations.Test;
public class Assignment_Collection_Arrays_Prgm
{
	public void Uniquenumber()
	{
		int a[]= {5,4,4,2,5,4,2,1};
		LinkedHashSet<Integer> set=new LinkedHashSet<Integer>();
		for (int i = 0; i < a.length; i++)
		{
			set.add(a[i]);
		}
		System.out.println("unique numbers");
		for (Integer num :set)
		{
			int count=0;
			for(int  i=0;i<a.length;i++)
			{
				if(num==a[i])
				{
					count++;
				}
			}
			if(count==1)
			{
				System.out.println(num+"   "+count);
			}
		}
	}
```

OUTPUT: unique numbers
1=1

36.WAP TO PRINT ONLY DUPLICATE NUMBERS IN ARRAY USING LINKEDHASHSET

```
public class Assignment_Collection_Arrays_Prgm
{
	@Test
	public void OnlyDuplicatesnumber()
{
		int a[]= {5,4,4,2,5,4,2,1};
		LinkedHashSet<Integer> set=new LinkedHashSet<Integer>();
		for (int i = 0; i < a.length; i++)
		{
			set.add(a[i]);
		}
		System.out.println("Only Duplicates number");
		for (Integer num :set)
		{
			int count=0;
			for(int  i=0;i<a.length;i++)
			{
				if(num==a[i])
				{
					count++;
				}
			}
			if(count>1)
			{
				System.out.println(num+"  "+count);
			}
		}
	}
```

OUTPUT:
```
Only Duplicates number
5=2
4=3
2=2
```

37.WAP TO FIND THE OCCURANCE OF EACH NUMBER IN ARRAY USING LINKEDHASHSET ✅

```
public class Assignment_Collection_Arrays_Prgm
{
	@Test
	public void OrderofEachNumberInArray()
	{
		int a[]= {5,4,4,2,5,4,2,1};
		LinkedHashSet<Integer> set=new LinkedHashSet<Integer>();
		for (int i = 0; i < a.length; i++)
		{
			set.add(a[i]);
		}
		System.out.println("Order of Each Number In the Array");
		for (Integer num :set)
		{
			int count=0;
			for(int  i=0;i<a.length;i++)
			{
				if(num==a[i])
				{
					count++;
				}
			}
			if(count>=1)
			{
				System.out.println(num+"   "+count+" times");
			}
		}
	}
}
```

OUTPUT:
```
Occurance of Each Number In the Array
	5=2 times
	4=3 times
	2=2 times
	1=1 times
```

38.WAP TO REMOVE REPEATED(DUPLICATE) OR UNIQUE NUMBERS IN GIVEN ARRAY USING LINKEDHASHSET ✅

```
import java.util.LinkedHashSet;
import org.testng.annotations.Test;
public class Assignment_Collection_Srting_Programs
{
	@Test
	public void RevomeRepeatedWordsORUNIQUEInTheString()
	{
		String s="hi hello hi hello welcome";
		String[] str = s.split(" ");
		LinkedHashSet<String> set=new LinkedHashSet<String>();

		for (int i = 0; i < str.length; i++)
		{
			set.add(str[i]);
		}

		System.out.println("Revome Repeated Words In The String");
		for (String word :set)
		{
			int count=0;
			for(int  i=0;i<str.length;i++)
			{
				if(word==(str[i]))
				{
					count++;
				}
			}
			if(count==1)
			{

			System.out.println(word+"="+count);
		}
		}
	}
```

OUTPUT:

```
Revome Repeated Words In The String
hi=1
hello=1
welcome=1
```

39.WAP TO PRINT DUPLICATE OR UNIQUE NUMBERS IN GIVEN ARRAY USING LINKEDHASHSET ✅

```
public class Assignment_Map_String_Program
{
@Test

	public void OnlyDuplicatesWords() {
		String s="hi hello hi hello welcome";
		LinkedHashSet<String> set=new LinkedHashSet<String>();
		String[] str = s.split(" ");
		for (int i = 0; i < str.length; i++) {
			set.add(str[i]);
		}
		System.out.println("Only duplicates words");
		for (String word :set) {
			int count=0;
			for(int  i=0;i<str.length;i++) {
				if(word.equals(str[i])) {
					count++;
				}
			}
			if(count>1)
			{
				System.out.println(word+"="+count);
			}
		}
	}
```

OUTPUT:

```
Only duplicates words
hi=2
hello=2
```

40.WAP TO PRINT THE ORDER OF OCUURANCE IN A GIVEN STRING USING LINKEDHASHSET ✅

```
public void OrderofOccurenceofEachWords()
	{
		String s="hi hello hi hello welcome";
		String[] str = s.split(" ");
		LinkedHashSet<String> set=new LinkedHashSet<String>();

		for (int i = 0; i < str.length; i++)
		{
			set.add(str[i]);
		}
		System.out.println("Order of Occurence of Each Word");
		for (String word :set)
		{
			int count=0;
			for(int  i=0;i<str.length;i++)
			{
				if(word.equals(str[i]))
				{
					count++;
				}
			}
			if(count>=1)
			{
				System.out.println(word+"="+count+" times");
			}
		}
	}
}
```

OUTPUT:

```
Order of Occurence of Each Word
hi=2 times
hello=2 times
welcome=1 times
```

41.WAP TO PRINT DUPLICATE NUMBERS IN GIVEN ARRAY USING MAP

```
public class Assignment_Map_With_Arrays_Program
{

@Test

public void OnlyDuplicateNum()
{
	int a[]= {5,4,4,2,5,4,2,1};
	HashMap<Integer, Integer> map=new HashMap<Integer, Integer> ();

	for (int i = 0; i < a.length; i++)
	{
		if(map.containsKey(a[i]))
		{
			map.put(a[i],map.get(a[i])+1);

		}
		else
		{
			map.put(a[i],1);
		}

	}

	System.out.println("only Dupicates num");
	for(Entry<Integer, Integer>m:map.entrySet())
	{
		if(m.getValue()>1)
		{
			System.out.println(m.getKey()+"  "+m.getValue());
		}
	}
}
```

OUTPUT:

```
only Dupicates num
2=2
4=3
5=2
```

42.WAP TO PRINT OCCURANCE OF EACH NUMBER IN ARRAY USING MAP

```
public class Assignment_Map_With_Arrays_Program
{

@Test

public void OccurenceofEachNumber()
{
	int a[]= {5,4,4,2,5,4,2,1};
	HashMap<Integer, Integer> map=new HashMap<Integer, Integer> ();

	for (int i = 0; i < a.length; i++)
	{
		if(map.containsKey(a[i]))
		{
			map.put(a[i],map.get(a[i])+1);

		}
		else
		{
			map.put(a[i],1);
		}

	}

	System.out.println("Occurence of each Number");
	for(Entry<Integer, Integer>m:map.entrySet())
	{
		if(m.getValue()>=1)
		{
			System.out.println(m.getKey()+"  "+m.getValue());
		}
	}
}
```

OUTPUT:

```
Occurence of each Number
1=1
2=2
4=3
5=2
```

43.WAP TO REMOVE REPEATED(DUPLICATE) OR UNIQUE NUMBERS IN GIVEN ARRAY USING MAP

```
public class Assignment_Map_With_Arrays_Program
{

public void RemoveDuplicatesORUNIQUE()
{
      int a[]= {5,4,4,2,5,4,2,1};
      HashMap<Integer, Integer> map=new HashMap<Integer, Integer> ();

      for (int i = 0; i < a.length; i++)
      {
            if(map.containsKey(a[i]))
            {
                  map.put(a[i],map.get(a[i])+1);

            }
            else
            {
                  map.put(a[i],1);
            }

      }

      System.out.println("revome duplicates Number");

         for(Entry<Integer, Integer>m:map.entrySet())

          if(m.getValue()==1)
          {
                  System.out.println(m.getKey()+"  "+m.getValue());
            }

}
}
```

OUTPUT:

```
revome duplicates Number
1=1
```

44.WAP TO PRINT DUPLICATE NUMBERS IN GIVEN STRING USING MAP

```
public class Assignment_Map_With_Arrays_Program
{
@Test

	public void DuplicateOnlyWithCount()
	{
		String s="hi hello hi hello welcome";
	HashMap<String, Integer> map=new HashMap<String, Integer> ();
		String[] str = s.split(" ");
		for (int i = 0; i < str.length; i++)
		{
			if(map.containsKey(str[i]))
			{
				map.put(str[i],map.get(str[i])+1);

			}
			else
			{
				map.put(str[i],1);
			}

		}

		System.out.println("Duplicate Only With Count");
		for(Entry<String, Integer>m:map.entrySet())
		{
			if(m.getValue()>1)
			{
			System.out.println(m.getKey()+"="+m.getValue());
			}
		}
	}

}
```

OUTPUT:

```
Duplicate Only with Count
hi=2
hello=2
```

45.WAP TO PRINT OCCURANCE OF EACH NUMBER IN ARRAY USING MAP

```
public class Assignment_Map_With_Arrays_Program
{

@Test

	public void OrderOFOccurenceEachWord()
	{
		String s="hi hello hi hello welcome";
		String[] str = s.split(" ");
	HashMap<String, Integer> map=new HashMap<String, Integer> ();

		for (int i = 0; i < str.length; i++)
		{
			if(map.containsKey(str[i]))
			{
				map.put(str[i],map.get(str[i])+1);

			}
			else
			{
				map.put(str[i],1);
			}

		}
	System.out.println("Order Of Occurence Each Word with count");
		for(Entry<String, Integer>m:map.entrySet())
		{
			if(m.getValue()>=1)
			{
			System.out.println(m.getKey()+"="+m.getValue());
			}
		}
	}
```

OUTPUT:

```
Order Of Occurence Each Word with count
hi=2
hello=2
welcome=1
```

46.WAP TO REMOVE REPEATED(DUPLICATE) OR UNIQUE NUMBERS IN GIVEN ARRAY USING MAP

```
public class Assignment_Map_String_Program
{
	@Test

	public void UniquesWords()
	{
		String s="hi hello hi hello welcome";
		String[] str = s.split(" ");
	HashMap<String, Integer> map=new HashMap<String, Integer> ();

		for (int i = 0; i < str.length; i++)
		{
			if(map.containsKey(str[i]))
			{
				map.put(str[i],map.get(str[i])+1);

			}
			else
			{
				map.put(str[i],1);
			}

		}
		System.out.println("Only uniques words");
		for(Entry<String, Integer>m:map.entrySet())
		{
			if(m.getValue()==1)
			{
			System.out.println(m.getKey()+"="+m.getValue());
			}
		}
	}
```

OUTPUT:

```
Only uniques words
welcome=1
```

47.WAP TO PRINT COUNT OF VOWELS AND CONSONANT

```
package string_program;

public class Count_Vowels_Consonant
{
	public static void main(String[] args)
	{

		String s="My name is ankur";
		s=s.toLowerCase().replace(" ","");
		int vowcount=0;
		int conscount=0;
		String ch="";
		String ch1="";

		for(int i=0; i<s.length();i++)
		{
		    if(s.charAt(i)=='a' || s.charAt(i)=='e' ||
s.charAt(i)=='i' || s.charAt(i)=='o' || s.charAt(i)=='u')
			{
				vowcount++;
				ch=ch+s.charAt(i);
			}
			else
			{
				conscount++;
				ch1=ch1+s.charAt(i);
			}
		}

		System.out.println(vowcount+" "+ "vowels");
		System.out.println(ch);

		System.out.println(conscount+" "+ "consonant");
		System.out.println(ch1);

		}
}
```

OUTPUT:

```
6 vowels
aeiuee
8 consonant
mynmsmjb
```

48.WAP TO FIND THE MAXIMUM LENGTH VALUES IN STRING

```
package string_program;

import org.testng.annotations.Test;


public class Find_Maximum_LengthValues_In_String_Array
{
      @Test

      public void MaxLength()
      {
        String []s= {"ab","abc","abcd","ef","edcf"};
         String maxLength=s[0];
         For (int i=0; i<s.length; i++)

            {
                  if(maxLength.length()<s[i].length())
                  {
                        maxLength=s[i];
                  }
            }

         for(int i=0; i<s.length; i++)
         {
            if(maxLength.length()==s[i].length())
            {
                  System.out.println(s[i]+" ");

            }
         }

      }
}
```

OUTPUT:

abcd
edcf

49.WAP TO FIND THE MAXIMUM LENGTH VALUES IN ARRAY

```
package string_program;

import org.testng.annotations.Test;


public class Find_Maximum_LengthValues_In_String_Arrays1
{
	@Test

	public void MinLength()
	{
	  String []s= {"12","123","1234","45","6789"};
	   String maxLength=s[0];
	   for(int i=0; i<s.length; i++)

	      {
	            if(maxLength.length()<s[i].length())
	            {
	                  maxLength=s[i];
	            }
	      }

	   for(int i=0; i<s.length; i++)
	   {
	      if(maxLength.length()==s[i].length())
	      {
	            System.out.println(s[i]+" ");

	      }
	   }

	}

}
```

OUTPUT:

1234
6789

50.WAP TO FIND THE MINIMUM LENGTH VALUES IN STRING

```
package string_program;

import org.testng.annotations.Test;


public class Find_Minimunm_LengthValues_In_String_Array
{
	@Test

	public void MinLength()
	{
		String []s= {"ab","abc","abcd","ef","edcf"};
	   String minLength=s[0];
	   for(int i=0; i<s.length; i++)

		{
			if(minLength.length()>s[i].length())
			{
				minLength=s[i];
			}
		}

	   for(int i=0; i<s.length; i++)
	   {
		   if(minLength.length()==s[i].length())
		   {
			   System.out.println(s[i]+" ");

		   }
	   }

	}

}
```

OUTPUT:

```
ab
ef
```

51.WAP TO FIND THE MINIMUM LENGTH VALUES IN ARRAY

```
package string_program;

import org.testng.annotations.Test;


public class Find_Minimunm_LengthValues_In_String_Arrays1
{
	@Test

	public void MinLength()
	{
		String []s= {"12","123","1234","45","6789"};
	   String minLength=s[0];
	   for(int i=0; i<s.length; i++)

		{
			if(minLength.length()>s[i].length())
			{
				minLength=s[i];
			}
		}

	   for(int i=0; i<s.length; i++)
	   {
		if(minLength.length()==s[i].length())
		{
			System.out.println(s[i]+" ");

		}
	   }

	}

}
```

OUTPUT:

12
45

52.WAP TO PRINT CONSONANT ✅

```
public class Print_Consonant
{

	@Test

	public void Consonant()
	{
		String st="india";
		char[] ch = st.toCharArray();
		int count=0;
		for(int i=0; i<ch.length; i++)
		{
			if(!(ch[i]=='a' || ch[i]=='e'|| ch[i]=='i' ||
ch[i]=='o'|| ch[i]=='u'))
			{
				count++;

				System.out.println(ch[i]);

			}

		}
	}

}
```

OUTPUT:

n
d

53.WAP TO PRINT VOWELS WITH DUPLICATE ✅

```
package string_program;

import org.testng.annotations.Test;


public class Print_Vowels_With_Duplicate
{
	@Test

	public void VowelsDuplicate()
	{
		String st="india";
		char[] ch = st.toCharArray();

		for(int i=0; i<ch.length; i++)
		{
			if(ch[i]=='a' || ch[i]=='e'|| ch[i]=='i' ||
ch[i]=='o'|| ch[i]=='u')
			{


				System.out.println(ch[i]);

			}
		}
	}

}
```

OUTPUT:

```
i
i
a
```

54.WAP TO PRINT VOWELS WITHOUT DUPLICATE USING MAP ✅

```
package string_program;

import java.util.HashMap;
import java.util.Map.Entry;

import org.testng.annotations.Test;
import org.testng.reporters.jq.Main;

public class Print_Vowels_Without_Duplicate_Using_HashMap
{

	public static void main(String[] args)
	{
		String s="india";
HashMap< Character, Integer> map=new HashMap<Character, Integer>();
		for (int i = 0; i < s.length(); i++)
		{

	if(s.charAt(i)=='a'||s.charAt(i)=='e'||s.charAt(i)=='i'||s.charAt
(i)=='o'||s.charAt(i)=='u')
			{
				if(map.containsKey(s.charAt(i)))
				{

			map.put(s.charAt(i),map.get(s.charAt(i))+1);

				}
				else
				{
					map.put(s.charAt(i),1);
				}
			}
		}
		for(Entry<Character, Integer>m:map.entrySet())
		{

			System.out.println(m.getKey());

		}
	}
}
```

OUTPUT: a
i

55.WAP TO PRINT VOWELS WITHOUT DUPLICATE USING LINKEDHASHSET

```
import java.util.LinkedHashSet;

import org.testng.annotations.Test;


public class Print_Vowels_WithOut_Duplicate_Using_LInkedHashSet
{
	@Test

	public void VowelsWoDuplicate()
	{
	@Test
	public void test()
	{
		String s="india";
	LinkedHashSet<Character> set = new LinkedHashSet<Character>();
		for(int i=0; i<s.length(); i++)
		{
			set.add(s.charAt(i));
		}
		for(Character ch:set)
		    {
			int count=0;
			if(ch=='a' || ch=='e'|| ch=='i' || ch=='o'|| ch=='u')
			{
				count++;

				System.out.println(ch);

			}
		}
	}

}
```

OUTPUT: i
a

56.WAP TO REVERSE AND PRINT ONLY INTEGER VALUE IN A COMBINATION OF STRING AND INTEGER ✅

```
package string_program;

import org.testng.annotations.Test;


public class Reverse_And_Print_Only_IntegerValues
{
	@Test

	public void ReverseInteger()
	{
		String s="Aabb12345dd45ee1ff44";
		String []st=s.split("[A-Z a-z]");

		for(int i=st.length-1; i>=0; i--)
		{
			System.out.print(st[i]+" ");
		}
	}
}
```

OUTPUT: 44 1 45 12345

```
public class Reverse_And_Print_Only_IntegerValues
{
	@Test

	public void ReverseInteger1()
	{
	String s="Tekion52inter54view234";
	String []st=s.split("[A-Z a-z]");

	for(int i=st.length-1; i>=0; i--)
	 {
		System.out.print(st[i]+" ");
	 }
	}

}
```

OUTPUT: 234 54 52

57.WAP TO REVERSE EACH WORD IN A STRING WITHOUT CHANGING PLACE OF THAT WORD ✅

```
public class Reverse_Each_Word_In_String
{
	@Test
	public void ReverseWord1()
	{
	String s="i am software engineer";
	String[] word=s.split(" ");
	String reverseword="";

	for(String w:word)
	{
	    StringBuilder sb = new StringBuilder(w);
	    sb.reverse();
	    reverseword=reverseword+sb.toString()+" ";
	}
	System.out.print(reverseword);
	}
}
```

OUTPUT: i ma erawtfos reenigne

```
public class Reverse_Each_Word_In_String
{
	@Test
	public void ReverseWord2()
	{
		String s="i am software engineer";
		String[] st = s.split(" ");

		for(int i=0; i<st.length; i++)
		{
			String str=st[i];

			for(int j=str.length()-1; j>=0; j--)
			{
				System.out.print(str.charAt(j));
			}
			System.out.print(" ");
		}
	}
}
```

OUTPUT: i ma erawtfos reenigne

58.WAP TO REVERSE EACH WORD ✅

```
public class Reverse_String_MoreThan_3Ways
{
	@Test
	public void Reversestring1()
	{
		String s="Mujeeb";
		for(int i=s.length()-1;i>=0; i--)
		{
			System.out.print(s.charAt(i));
		}
	}
}
```

OUTPUT: beejuM

```
public class Reverse_String_MoreThan_3Ways
{
    @Test
    public void Reversestring2()
    {
		String s="Software";
		char[] ch = s.toCharArray();
		for(int i=ch.length-1; i>=0; i--)
		{
			System.out.print(ch[i]);
		}

    }
}
```

OUTPUT: erawtfoS

59.WAP TO REVERSE EACH WORD ✅

```
public class Reverse_String_MoreThan_3Ways
{
    @Test
    public void Reversestring3()
    {
           String s="softwares";
           String s1="";

           for(int i=0; i<s.length(); i++)
           {
                 s1=s.charAt(i)+s1;
           }
           System.out.println(s1);
     }
}
```

OUTPUT: serawtfoS

```
public class Reverse_String_MoreThan_3Ways
{
    @Test
    public void Reversestring4()
    {

           String st="softwares";
           char[]s = st.toCharArray();
        int count=0;
        for(char c : s)
           {
                 count++;
           }

        for(int i=count-1; i>=0; i--)
        System.out.println(s[i]);
     }
}
```

OUTPUT: serawtfoS

60.WAP TO REVERSE STRING ✅

```java
package string_program;

import org.testng.annotations.Test;


public class Reverse_String
{
	@Test

	public void ReverseString()
	{
		String s="i am software engineer";
		String[] st = s.split(" ");

		for(int i=st.length-1; i>=0; i--)
		{
			System.out.print(st[i]+" ");
		}
	}
}
```

OUTPUT: engineer software am i

```java
public class Reverse_String1
{
	@Test
    public void Reversestring3()
    {

		String s="india";
		String rev="";

		for(int i=s.length()-1; i>=0; i--)
		{
			rev=rev+s.charAt(i);
		}
		System.out.println(rev);
	}
}
```

OUTPUT: aidni

61.WAP TO SEGRIGATE ALPHA, NUMERIC AND SPECIAL CHARACTER ✅

```
package string_program;

import org.testng.annotations.Test;


public class Segrigate_Alpha_Numeric_Special_Character
{
	@Test

	public void segrigate()
	{
		String s="a1c$d3R%";
		String alpha=" ";
		String num=" ";
		String spe=" ";

		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)>='a' && s.charAt(i)<='z' ||
s.charAt(i)>='A' && s.charAt(i)<='Z')
			{
				alpha=alpha+s.charAt(i);
			}
			else if (s.charAt(i)>='1' && s.charAt(i)<='9')
			{
				num=num+s.charAt(i);

			}
			else
			{
				spe=spe+s.charAt(i);
			}
		}

		System.out.println(alpha+num+spe);
	}


}
```

OUTPUT: acdR 13 $%

62.
INPUT IS---- aabbabc
OUTPUT IS---- a2b2a1b1c1

```
package string_program;

import org.testng.annotations.Test;


public class String_Imp
{
 @Test

 public void test()
 {
       String s="aabbabc";
       int count=1;
       for(int i=0; i<s.length()-1; i++)
       {
             if(s.charAt(i)==s.charAt(i+1))
             {
                   count++;
             }
             else
             {
                   System.out.print(s.charAt(i)+""+count);
                   count=1;
             }
       }
       System.out.println(s.charAt(s.length()-1)+""+count);
 }
}
```

OUTPUT:
a2b2a1b1c1

63.

INPUT IS---- a1b2c1
OUTPUT IS---- abbc

```
package Array_Practice_Program;

public class a
{
	public static void main(String[] args)
	{
		String s="a1b2c1";
		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)>='a' && s.charAt(i)<='z')
			{
			int num = Character.getNumericValue(s.charAt(i+1));

				for(int j=0; j<num; j++)
				{
					System.out.print(s.charAt(i));
				}
			}
		}

	}

}
```

OUTPUT:
abbc

64.

INPUT IS ----- I LOVE TYSS

OUTPUT IS----- S SYTE VOLI

```
package string_program;

import org.testng.annotations.Test;


public class String_Imp2
{
	@Test

	public void test()
	{
		String s="I LOVE TYSS";
		String s1=s.replaceAll(" ",""); // ILOVETYSS
		int count=s1.length()-1;
		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)!=' ')
			{
				System.out.print(s1.charAt(count--));
			}
			else
			{
				System.out.print(s.charAt(i));
			}
		}
	}

}
```

OUTPUT:

S SYTE VOLI

65.WAP TO CHECK THE STRING IS PALINDROME OR NOT ✅

```
package string_program;

import org.testng.annotations.Test;


public class String_Palindrome
{
	@Test

	public void palindrome()
	{
		String s="dad";
		String s1="";

		for(int i=s.length()-1; i>=0; i--)
		{
			s1=s1+s.charAt(i);
		}
		if(s1.equals(s))
		{
			System.out.println("String is palindrome");
		}

		else
		{
			System.out.println("String is not a Palindrome");
		}

	}

}
```

OUTPUT: String is palindrome

66.WAP TO PRINT THE SUM OF DIGITS IN STRING ✅

```java
package string_program;

import org.testng.annotations.Test;


public class Sum_Of_Digits_In_String
{
	@Test

	public void SumOfDigitsInString()
	{
		String s="a2b4c6";
		int sum=0;
		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)>='0' &&  s.charAt(i)<='9')
			{
				int n=s.charAt(i)-48; //Asci value of 2 is 50...
				sum=sum+n;
			}
		}

		System.out.println(sum);
	}

}
```

OUTPUT: 12

67.WAP TO SWAP 1ST AND LAST CHARACTER ✅

```java
package string_program;

import org.testng.annotations.Test;


public class Swap_1st_And_LastWord_InString
{
	@Test

	public void Swap1stAndLastFromString()
	{
		String s="i am software engineer";
		String[] st = s.split(" ");

		//st[0]=i  st[1]=am  st[2]=software  st[3]=engineer

		String temp=st[0];
		st[0]=st[st.length-1];
		st[st.length-1]=temp;

		for(int i=0; i<st.length; i++)
		{
			System.out.print(st[i]+" ");
		}

	}

}
```

OUTPUT: engineer am software i

68.WAP TO REVERSE STRING USING RECURSION

```
package Recursion;

public class Reverse_String_using_Recursion
{
	public static void main(String[] args)
	{
		String s="india";
		rev(s,s.length()-1); //india,4

	}
	public static void rev(String s,int len) //india,4
	{
		if(len>=0)
		{
			System.out.print(s.charAt(len)); //a
			len--;
			rev(s,len);//india,4
		}
	}

}
```

OUTPUT: aidni

69.WAP TO REVERSE AND SPLIT EACH WORD

```
package string_program;

import org.testng.annotations.Test;


public class Reverse_Split_Each_Word
{

	@Test

	public void test()
	{
		String s="I am from Bangalore";
		String[] str = s.split(" ");

		for (int i=0; i<str.length; i++)
		{
			String temp = str[0];

			for(int j=1; j<str.length; j++)
			{
				str[j-1]=str[j];
				System.out.print(str[j]+" ");
			}

			str[str.length-1]=temp;
			System.out.println(temp);
		}
	}
}
```

OUTPUT:

```
am from Bangalore I
from Bangalore I am
Bangalore I am from
I am from Bangalore
```

70.WAP TO CHECK THE STRING IS ANAGRAM OR NOT ✅

```
package string_program;
import java.util.Arrays;

public class Anagram
{
	public static void main (String[] args)
	{

		boolean status=false;
		String s1="Listen";
		String s2="Silent";
		if(s1.length()==s2.length())
		{
			char ch1[]=s1.toLowerCase().toCharArray();
			char ch2[]=s2.toLowerCase().toCharArray();
			Arrays.sort(ch2);
			Arrays.sort(ch1);
			if(Arrays.equals(ch1, ch2))
			{
				System.out.println("anagram");
			}
			else
			{
				System.out.println("not anagram");
			}

		}
		else
		{
			System.out.println("not anagram");
		}

	}
}
```

OUTPUT:

Anagram

71.WAP TO FIND THE COUNT OF UPPERCASE LETTERS IN A WORD ✅

```
package string_program;

import org.testng.annotations.Test;


public class Count_Of_Upper_Case
{
	@Test

	public void test()
	{
		String s="JaNuArY";
		int count=0;

		for(int i=0; i<s.length(); i++)
		{
			if(s.charAt(i)>='A' && s.charAt(i)<='Z')
			{
				count++;
			}

		}
			System.out.println(count);

	}

}
```

OUTPUT:

4

72.WAP TO FIND THE FACTORIAL

```
package Number_Series_programs;

import org.testng.annotations.Test;


public class Factorial_Series
	{
	@Test


		public void Factorial()
		{
			int i=1, fact=1;
			while(i<=5)
			{
			fact=fact*i;
			i++;
			}
			System.out.println(fact);
		}
	}
```

OUTPUT:

120

73.WAP TO FIND THE FIBONACCI SERIES

```
package Number_Series_programs;

import org.testng.annotations.Test;


public class Fibonacci_Series
	{
	@Test


		public void Fibonacci()
		{
			int n1=0,n2=1,sum=0;
			int i=1;

			while(i<=10)
			{
				sum=n1+n2;
				n1=n2;
				n2=sum;
			System.out.println(sum);
			i++;
			}
		}
	}
```

OUTPUT:

```
1
2
3
5
8
13
21
34
55
```

74.WAP TO FIND THE INTEGER IS PALINDROME OR NOT

```
package Number_Series_programs;

import org.testng.annotations.Test;


public class Integer_Palindrome
{
	@Test

	public  void Palindrome()
		{
			int num=121,sum=0,rem=0,temp=num;
			while(num>0)
			{
				rem=num%10;
				sum=(sum*10)+rem;
				num=num/10;
			}
			if(temp==sum)
			{
			System.out.println("its a palindrome"+" "+ sum);
			}
			else
			{
			System.out.println("its not a palindrome"+" "+sum);
			}

		}
	}
```

OUTPUT:

```
its a palindrome 121
```

75.WAP TO FIND THE PRIME NUMBER IN ARRAY

```
package Number_Series_programs;

import org.testng.annotations.Test;


public class Prime_No_In_Array
{
	@Test

	public  void PrimeInArray()
		{
		  int []a= {7,4,2,1,5,9,8,3};
		for(int i=0 ;i<a.length; i++)
			{
				int count=0;
				for(int j=1; j<a.length; j++)
				{
					if(i%j==0)
					{
						count++;
					}
				}

				if(count==2)
				{
				System.out.println(i);
				}

				}

			}
		}
```

OUTPUT:

```
2
3
5
7
```

76.WAP TO FIND THE PRIME NUMBER FROM 1 TO 30

```
package Number_Series_programs;

import org.testng.annotations.Test;
public class Prime_Number_From_1_30
{
	@Test
	public void Prime()
	{
		int a=1;
		int b=30;

		for(int i=a; i<=b; i++)
		{
			int count=0;
			for(int j=1; j<=i; j++)
			{
				if(i%j==0)
				{
					count++;
				}
			}
			if(count==2)
			{
				System.out.println(i);
			}

		}
	}
}
```

OUTPUT:

```
2
3
5
7
11
13
17
19
23
29
```

77.WAP TO FIND THE PRIME NUMBER

```
package Number_Series_programs;

public class Prime_Number
{
	public static void main(String[] args)
	{
		int num=7, count=0;
		for(int i=1;i<=num;i++)
		{
			if(num%i==0)
			{
				count++;
			}
		}
		if(count==2)
		{
		System.out.println(num+" "+ "is a prime number");
		}
		else
		{
		System.out.println(num+" "+ "is not a prime number");
		}

	}
}
```

OUTPUT:

```
7 is a prime number
```

78.WAP TO FIND THE SMALLEST EVEN NUMBER IN ARRAY

```
package Number_Series_programs;

import java.util.Scanner;

import org.testng.annotations.Test;

public class Smallest_Even_No_In_Array
{
	@Test

	public void SmaEvenInArray()
	{
		int[]a= {5,3,2,1,7,8,4};
		int temp=a[0];


		 for(int i=0; i<a.length; i++)
		 {
			if(a[i]%2==0)
			{
				if(temp>a[i])
				{
					temp=a[i];

				}
			}
		 }
		 System.out.println(temp +" "+ "is the smallest Even No");

	}
}
```

OUTPUT:

2 is the smallest Even No

INTERVIEW PROGRAMS

79. WAP TO PRINT THE NUMBER CONTAINS 1

```
package Array_Practice_Program;

public class Print_numbers_Contains_1
{
	public static void main(String[] args)
	{
		String []a = {"12","34","81","21","71"};
		for(int i=0;i<a.length;i++)
		{
			if(a[i].contains("1"))
			{
				System.out.print(a[i]+" ");
			}
		}

	}
}
```

OUTPUT:

12 81 21 71

80. WAP TO PRINT REPEATED NUMBERS WITH POSITION

```
public class Print_Repeated_Numbers_With_Position
{
    public static void main(String[] args)
    {
       int[] a={10,20,30,40,10,20,60,70,60,30,10,20,10};
       boolean[] b=new boolean[a.length];
       for(int i=0;i<a.length;i++)
       {
           if(b[i]==false)
           {
               int count=0;
               String repeat="";
               for(int j=i+1;j<a.length;j++)
               {
                   if(a[i]==a[j])
                   {
                       count++;
                       b[j]=true;
                       repeat=repeat+(j+1)+",";
                   }
               }
               if(count>0)
               {
                 System.out.println(a[i]+" is repeated at "+repeat+ "
positions");
               }
           }
       }
    }
}
```

OUTPUT:

```
10 is repeated at 5,11,13, positions
20 is repeated at 6,12, positions
30 is repeated at 10, positions
60 is repeated at 9, positions
```

81. WAP TO REVERSE CHARACTER AND SPECIAL CHARACTER, INTEGER POSITION SAME

APPROACH 1

```
package string_program;

import org.testng.annotations.Test;

public class Reverse_Character_Special_Position_Same
{
	public static void main(String[] args)
	{
		String s="ab@s%y%m";
		String s1="";
		for(int i=0;i<s.length();i++)
		{
			if(s.charAt(i)>='a'&&s.charAt(i)<='z')
			{
				s1=s1+s.charAt(i);

			}
		}
		int count=s1.length()-1;
		for(int i=0;i<s.length();i++)
		{
			if((s.charAt(i)>='a'&&s.charAt(i)<='z'))
			{
				System.out.print(s1.charAt(count--));
			}
			else
			{
				System.out.print(s.charAt(i));

			}
		}
	}
}
```

OUTPUT:
my@s%b%a

APPROACH 2

```
public class Reverse_Character_Special_Position_Same
{
	@Test

	public void test()
	{
		String s="ab@s%y%m";
		String s1=s.replaceAll("[^a-zA-Z]", "");
		int count=s1.length()-1;
		for(int i=0;i<s.length();i++)
		{
	if(s.charAt(i)>='A'&& s.charAt(i)<='Z'|| s.charAt(i)>='a'&&
s.charAt(i)<='z')
			{
				System.out.print(s1.charAt(count--));
			}
			else
			{
				System.out.print(s.charAt(i));
			}
		}
	}
}
```

OUTPUT:

my@s%b%a

82. WAP TO REVERSE CHARACTER, INTEGER AND SPECIAL CHARACTER POSITION REMAIN SAME

```
public class
Reverse_Character_Integer_And_Special_Character_Position_Same
{

public void test()
      {
            String s="ab@s%4%m";
           String s1 = s.replaceAll("[^a-z A-Z 0-9]", "");
            int count=s1.length()-1;
            for(int i=0;i<s.length();i++)
            {
                  if(s.charAt(i)>='A'&& s.charAt(i)<='Z'||
s.charAt(i)>='a'&& s.charAt(i)<='z'|| s.charAt(i)>='0' &&
s.charAt(i)<='9')
                  {
                        System.out.print(s1.charAt(count--));
                  }
                  else
                  {
                        System.out.print(s.charAt(i));
                  }
            }
      }
}
```

OUTPUT:

m4@s%b%a

83. WAP TO REVERSE STRING

Approach 1

```
public class Reverse
@Test
public void ReverseWord3_Approach1()
{
	String s="i am software engineer";
	String[] st = s.split(" ");

	for(int i=st.length-1; i>=0; i--)
	{
		String str=st[i];
		for(int j=str.length()-1; j>=0; j--)
		{
			System.out.print(str.charAt(j));
		}
		System.out.print(" ");
	}
}
```

OUTPUT:

```
reenigne erawtfos ma i
```

Approach 2

```
public class Reverse

@Test
public void ReverseWord3_Approach2()
{
	String s="i am software engineer";

	for(int i=s.length()-1; i>=0; i--)
	{
		System.out.print(s.charAt(i));
	}
}
}
```

OUTPUT:

```
reenigne erawtfos ma i
```

84. WAP TO REVERSE STRING

```
public class Reverse
	@Test
	 public void ReverseWord2()
	 {
		String s="i am software engineer";
		String[] st = s.split(" ");

		for(int i=0; i<st.length; i++)
		{
			String str=st[i];

			for(int j=str.length()-1; j>=0; j--)
			{
				System.out.print(str.charAt(j));
			}
			System.out.print(" ");
		}

	 }
```

OUTPUT:

```
i ma erawtfos reenigne
```

85. WAP TO PRINT REPEATED NUMBERS WITH POSITION

```
public class Print_Repetaed_Numbers_With_Position
{
@Test
public void position()
  {
        int []a= {1,1,3,1,3,4};
        ArrayList<Object> list = new ArrayList<Object>();
        for(int i=0; i<a.length; i++)
        {
              if(list.contains(a[i]))
              {
           System.out.println(a[i]+" is repeating at position "+(i+1));
              }
              else
              {
                    list.add(a[i]);
              }
        }
  }
}
```

OUTPUT:

```
1 is repeating at position 2
1 is repeating at position 4
3 is repeating at position 5
```

86. WAP TO REVERSE AND PRINT ONLY INTEGER VALUE IN A COMBINATION OF STRING AND INTEGER

```
package string_program;

import org.testng.annotations.Test;


public class Reverse_And_Print_Only_IntegerValues_With_Comma
{

@Test

public void ReverseInteger2()
{

	String s="Aabb12345dd45ee1ff44";
	String []st=s.split("[A-Z a-z]");
	System.out.print(st[st.length-1]);

	for(int i=st.length-2; i>=0; i--)
	{
		if(st[i].equals(""))
		{

		}
		else
		{
			System.out.print(","+st[i]);

		}
	}
}
```

OUTPUT:
44,1,45,12345

87. WAP TO REVERSE AND PRINT ONLY INTEGER VALUE IN A COMBINATION OF STRING AND INTEGER

```
package string_program;

import org.testng.annotations.Test;


public class Print_Only_IntegerValues_With_Comma
{

@Test

public void ReverseInteger3()
{

	String s="Aabb12345dd45ee1ff44";
	String []st=s.split("[A-Z a-z]");

	for(int i=0; i<st.length-1; i++)
	{
		if(st[i].equals(""))
		{

		}
		else
		{
			System.out.print(st[i]+",");

		}
	}
	System.out.print(st[st.length-1]);
}
}
```

OUTPUT:
12345,45,1,44

88.WAP to find out how many 1 and 0 in number= 100110011

```
String s = "100110011";

HashMap<Character,Integer> map=new HashMap<Character,Integer>();
for(int i=0;i<s.length();i++)
{
    if(map.containsKey(s.charAt(i)))
    {
        map.put(s.charAt(i), map.get(s.charAt(i))+1);
    }
    else
    {
        map.put(s.charAt(i),1);
    }
}
System.out.println(map);
```

89. WAP to find the consecutive numbers in a string

```
package String;

public class ConsecutiveNumber {

      public static void main(String[] args)
    {
        String s = "9975612389";
        String s1 = "";
        for (int i = 0; i < s.length(); i++) {
            if (i < s.length()-1 && s.charAt(i + 1)-s.charAt(i) == 1)
               {
                if (s1.length() == 0)
                {
                    s1 = s1 + s.charAt(i) + s.charAt(i + 1);
                }
               else
                {
                    s1 = s1 + s.charAt(i + 1);
                }
            }
            else
            {
                System.out.println(s1);
                s1 = "";
            }
        }
    }

}
```

90. I/P: a2b2c3d1
O/P: aabbcccd

```
package String;

public class aabbcccd {

	public static void main(String[] args) {
		String s = "a2b2c3d1";
		// o/p = aabbaaac;
		for (int i = 0; i < s.length(); i++) {
			char ch = s.charAt(i);
			if (ch >= '0' && ch <= '9') {
				int n = (ch - 48);
				for (int j = 0; j < n; j++) {
					System.out.print(s.charAt(i-1));
				}
			}
		}
	}
}
```

Aabbcccd

COMBINATION OF NUMBERS

```
public class CombinationOfNumber {

	public static void main(String[] args) {
		int[] a = { 8, 4, 7, 3, 9, 2, 5, 6 };
		for (int i = 0; i < a.length; i++) {
			for (int j = i + 1; j < a.length; j++) {
				if (a[i] + a[j] == 11) {
		System.out.println(a[i] + " + " + a[j] + " = "+(a[i]+a[j]));
				}
			}
		}
	}
}
```

FIND POSITION OF AN ARRAY

```
public class FindPosOfAnArray {

	public static void main(String[] args) {
		int[] a = {2,5,6,10,3,7,7,10,5};

		for (int i = 0; i < a.length; i++) {
			if(a[i] == 10)
			System.out.println(a[i]+"---"+(i+1));
		}
	}
}
```

SUM OF DYNAMIC ARRAY

```java
public class InfiniteAndDynamicArray {

	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the size of array : ");

		int size = sc.nextInt();
		System.out.println("enter the array : ");

		int a[] = new int[size];
		int sum = 0;

		for (int i = 0; i < a.length; i++) {
			a[i] = sc.nextInt();
			sum = sum + a[i];
		}
		System.out.println("the sum is : "+sum);
	}
}
```

MERGE TWO ARRAYS AND FIND MAX

```java
public class MergeArrayAndFindMax {

	public static void main(String[] args) {
		int[] a = { 7, 8, 5, 6 };
		int[] b = { 17, 18, 15, 16 };
		int[] c = new int[a.length + b.length];
		for (int i = 0; i < a.length; i++) {
			c[i] = a[i];
		}
		for (int j = 0; j < b.length; j++) {
			c[a.length + j] = b[j];
		}
		for (int i = 0; i < c.length; i++) {
			Arrays.sort(c);
			System.out.print(c[i] + " ");

		}
		System.out.println();
		System.out.println("maximum is : " + c[c.length - 1]);
	}
}
```

REVERSE AN ARRAY TILL A PARTICULAR POSITION

```java
public class ReverseAnArrayTillParticularPos1 {

	public static void main(String[] args) {
		int a[] = { 1, 2, 3, 4, 5, 6, 3, 2, 4 };
		int pos = 5;
		for (int i = 0; i < a.length; i++) {
			if (pos > 0) {
				System.out.print(a[--pos] + " ");
			} else {
				System.out.print(a[i] + " ");
			}
		}
	}
}
```