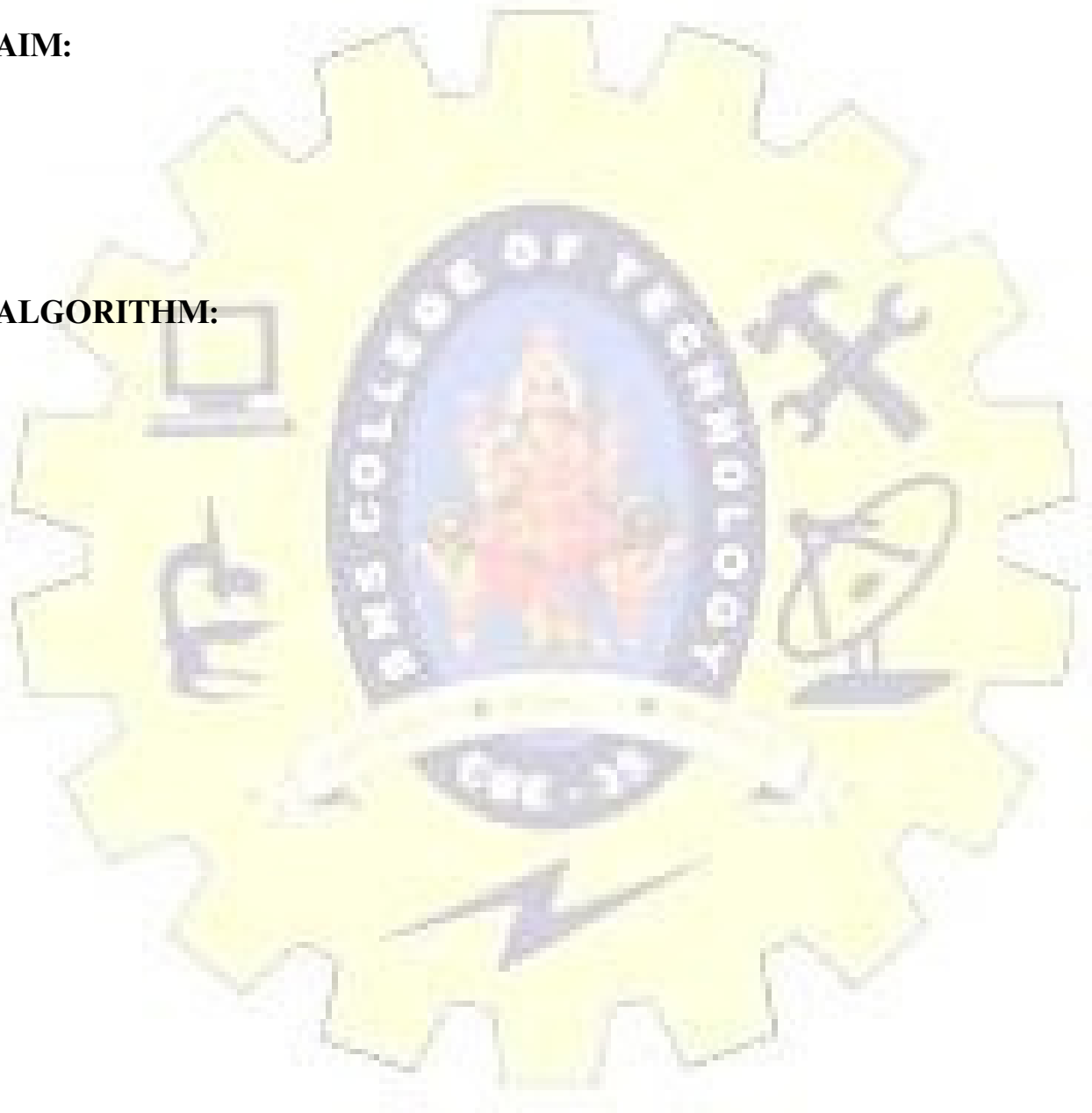


Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
import java.util.Scanner;

interface Interface1 {
    void method1();
}

interface Interface2 {
    void method2();
}

interface Interface3 {
    void method3();
}

class MultiInterfaceImplementation implements Interface1,
Interface2, Interface3 {
    public void method1() {
        System.out.println("Implementation of method1 from Interface1.");
    }

    public void method2() {
        System.out.println("Implementation of method2 from Interface2.");
    }

    public void method3() {
        System.out.println("Implementation of method3 from Interface3.");
    }
}

public class MultipleInterfacesDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        MultiInterfaceImplementation obj = new MultiInterfaceImplementation();
    }
}
```

```
System.out.println("Choose which method to execute:");
System.out.println("1: method1 from Interface1");
System.out.println("2: method2 from Interface2");
System.out.println("3: method3 from Interface3");
int choice = scanner.nextInt();
switch (choice) {
case 1:
obj.method1();
break;
case 2:
obj.method2();
break;
case 3:
obj.method3();
break;
default:
System.out.println("Invalid choice.");
}
scanner.close();
}
```

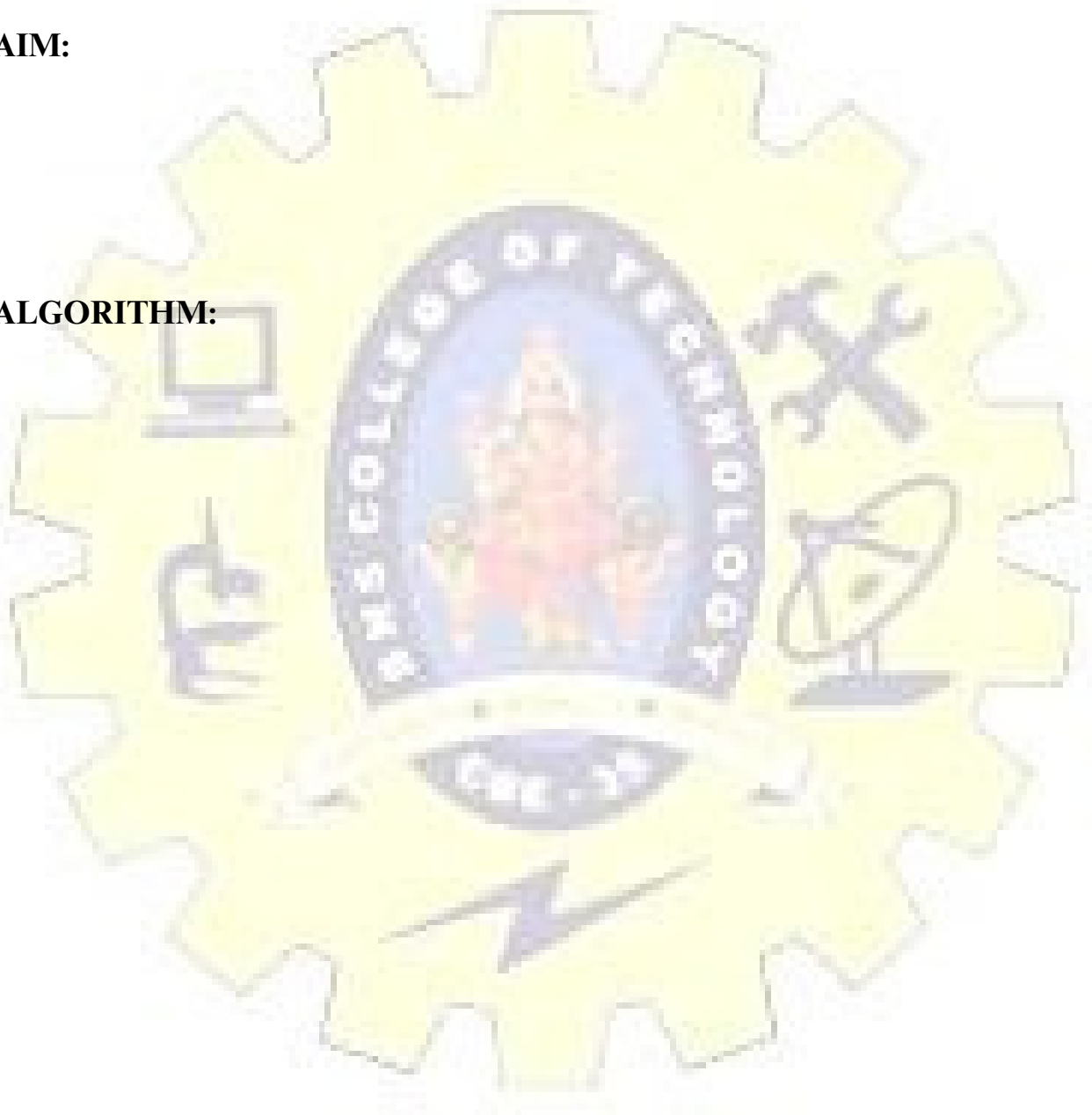
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

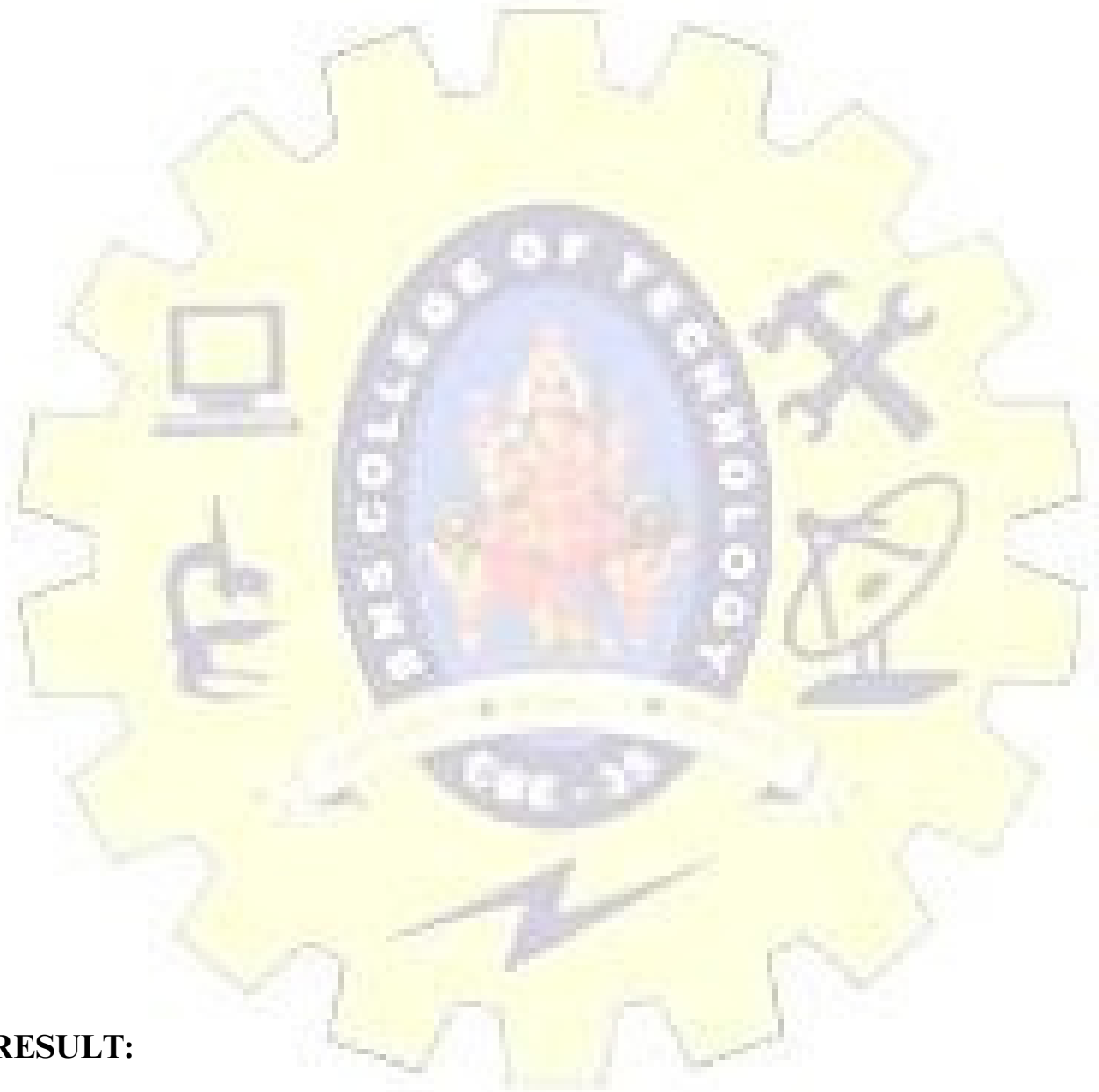
```
import java.util.Scanner;

abstract class AbstractClass {
    abstract void abstractMethod();

    void concreteMethod() {
        System.out.println("Concrete method from AbstractClass.");
    }
}

class ConcreteClass extends AbstractClass {
    void abstractMethod() {
        System.out.println("Implementation of abstractMethod in ConcreteClass.");
    }
}

public class AbstractClassDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        AbstractClass obj = new ConcreteClass();
        obj.concreteMethod();
        obj.abstractMethod();
        scanner.close();
    }
}
```



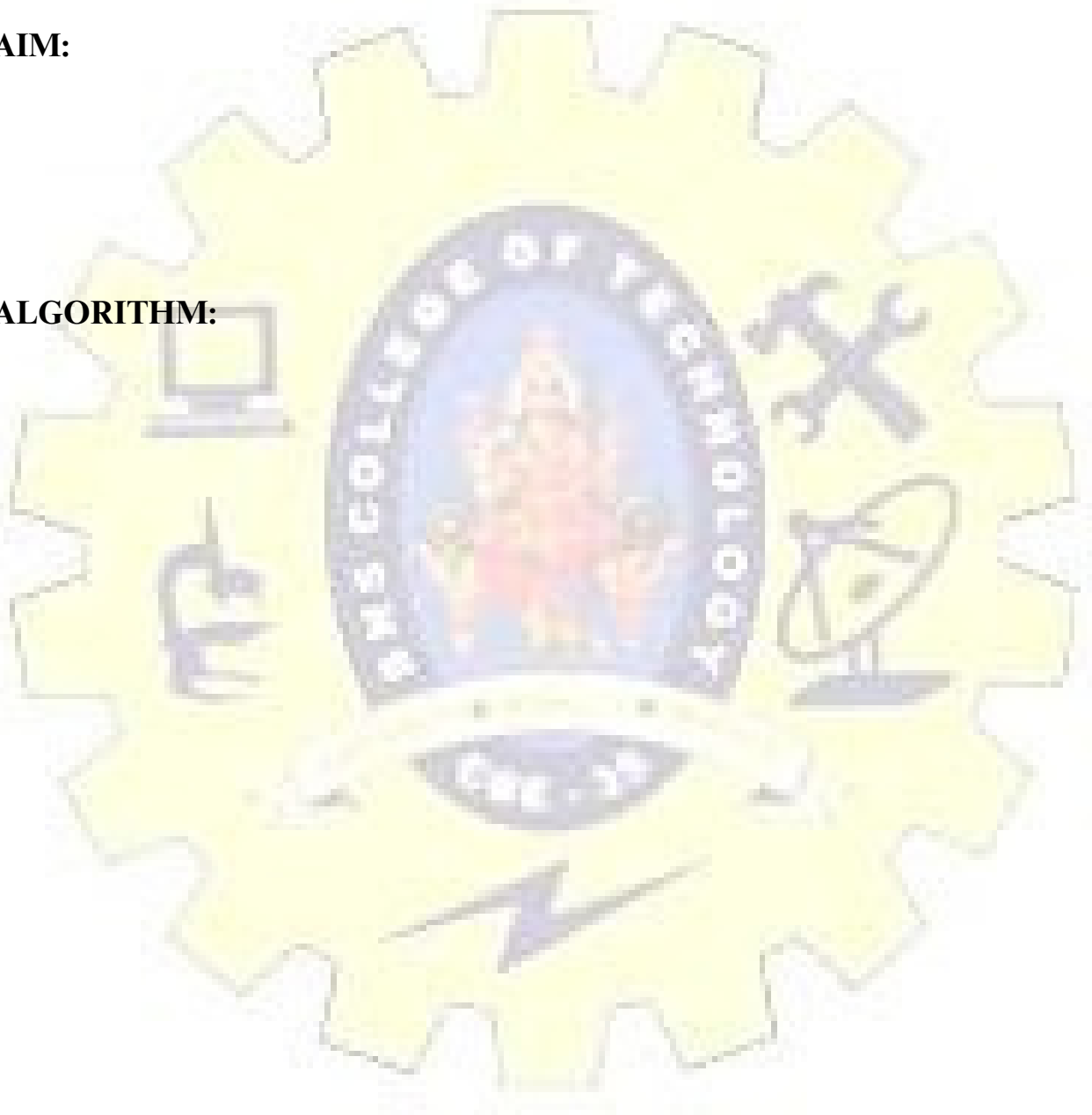
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
abstract class Animal {  
    abstract void makeSound();  
}  
  
class Dog extends Animal {  
    void makeSound() {  
        System.out.println("Woof Woof");  
    }  
}  
  
public class Main {  
    public static void main(String[] args) {  
        Animal myDog = new Dog();  
        myDog.makeSound();  
    }  
}
```

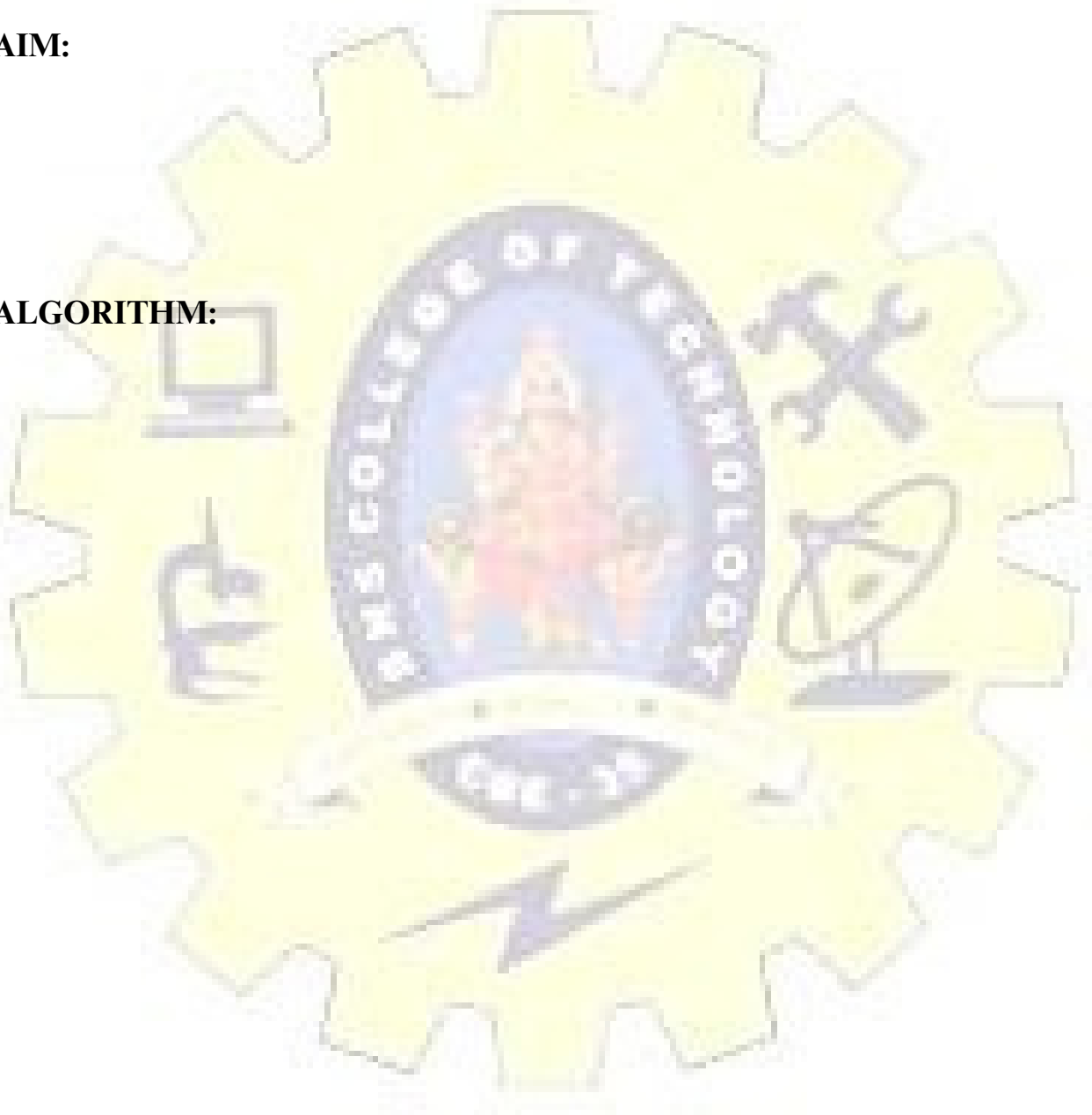
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
import java.util.Scanner;

public class ArithmeticExceptionDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        try {
            System.out.print("Enter numerator: ");
            int numerator = scanner.nextInt();
            System.out.print("Enter denominator: ");
            int denominator = scanner.nextInt();
            int result = numerator / denominator;
            System.out.println("Result: " + result);
        } catch (ArithmeticException e) {
            System.out.println("Error: Cannot divide by zero.");
        } finally {
            scanner.close();
        }
    }
}
```

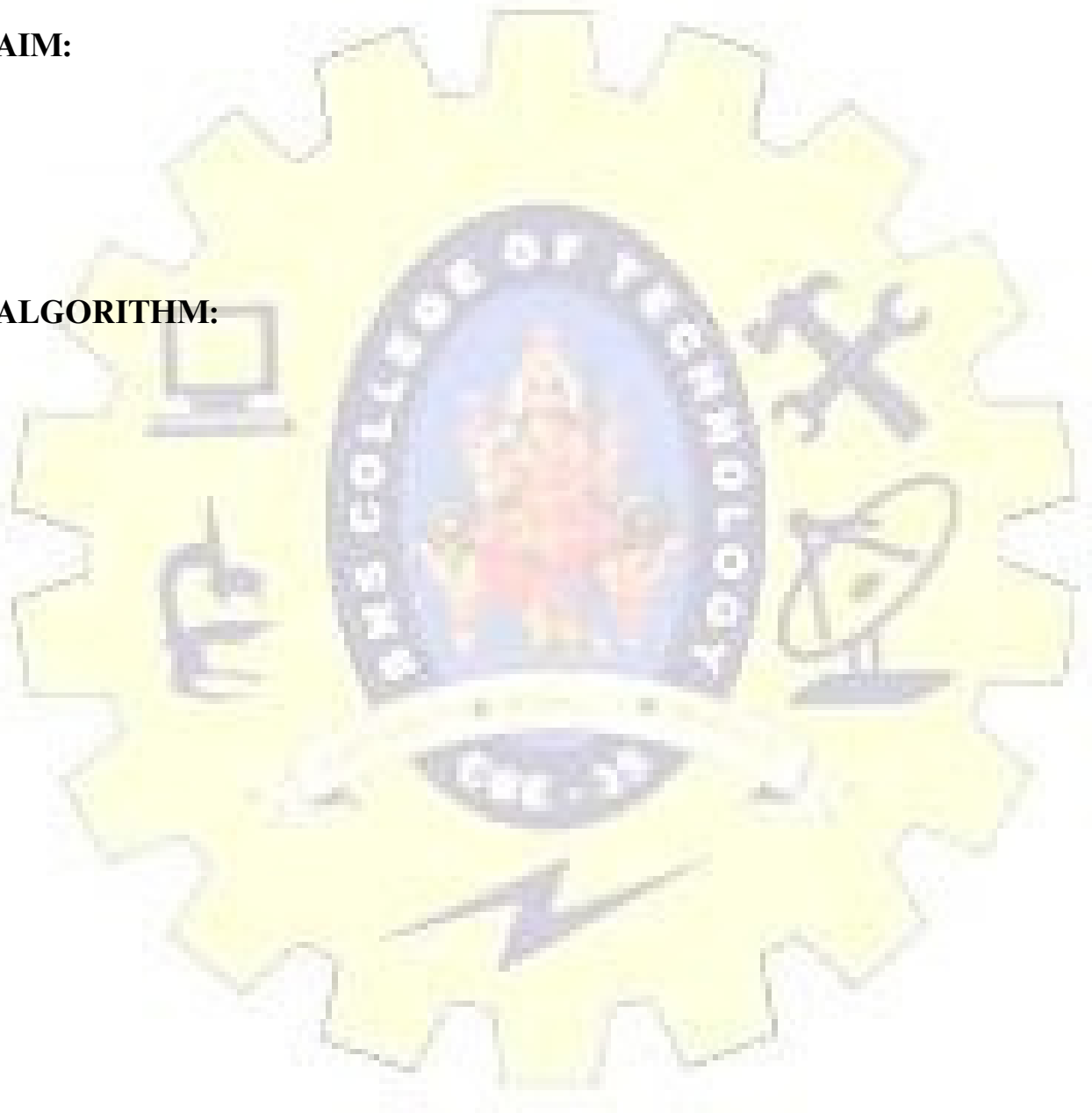
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
public class ArrayIndexExceptionDemo {  
    public static void main(String[] args) {  
        int[] numbers = {1, 2, 3, 4, 5};  
        try {  
            System.out.println(numbers[10]);  
        } catch (ArrayIndexOutOfBoundsException e) {  
            System.out.println("Error: Array index is out of bounds.");  
        }  
    }  
}
```

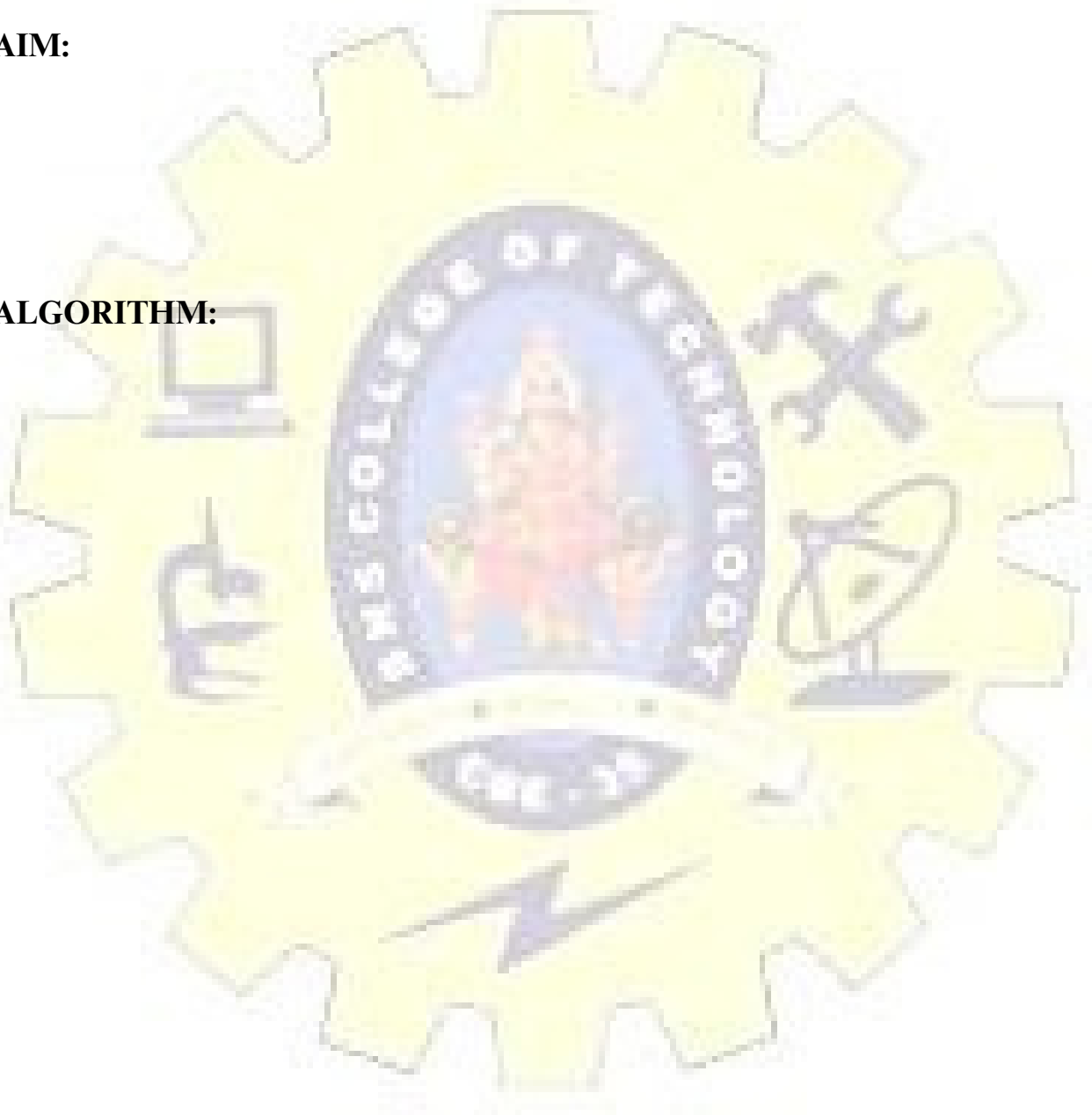
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
class MyThread extends Thread {  
    public void run() {  
        for (int i = 0; i < 5; i++) {  
            System.out.println("Thread " +  
Thread.currentThread().getId() + " is running, count: " + i);  
            try {  
                Thread.sleep(1000);  
            } catch (InterruptedException e) {  
                System.out.println(e);  
            }  
        }  
    }  
}  
  
public class ThreadExample {  
    public static void main(String[] args) {  
        MyThread thread1 = new MyThread();  
        thread1.start();  
        MyThread thread2 = new MyThread();  
        thread2.start();  
    }  
}
```

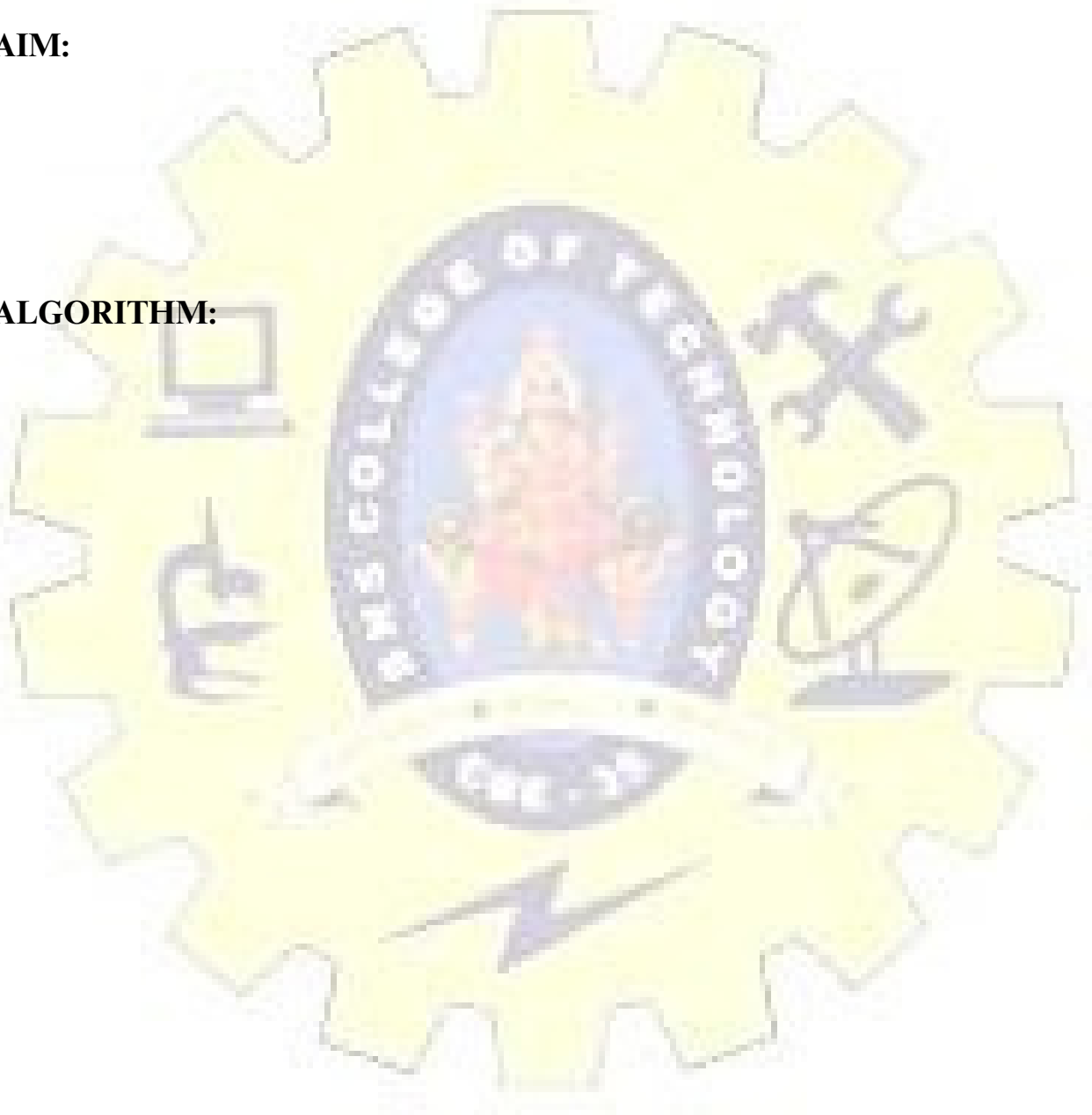
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
import java.applet.Applet;  
import java.awt.Graphics;  
  
public class HelloApplet extends Applet {  
    public void paint(Graphics g) {  
        g.drawString("Hello, Applet!", 20, 20);  
    }  
}
```

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Applet Example</title>  
</head>  
<body>  
    <h1>Applet Example</h1>  
    <applet code="HelloApplet.class" width="300" height="300">  
        Your browser does not support applets.  
    </applet>  
</body>  
</html>
```

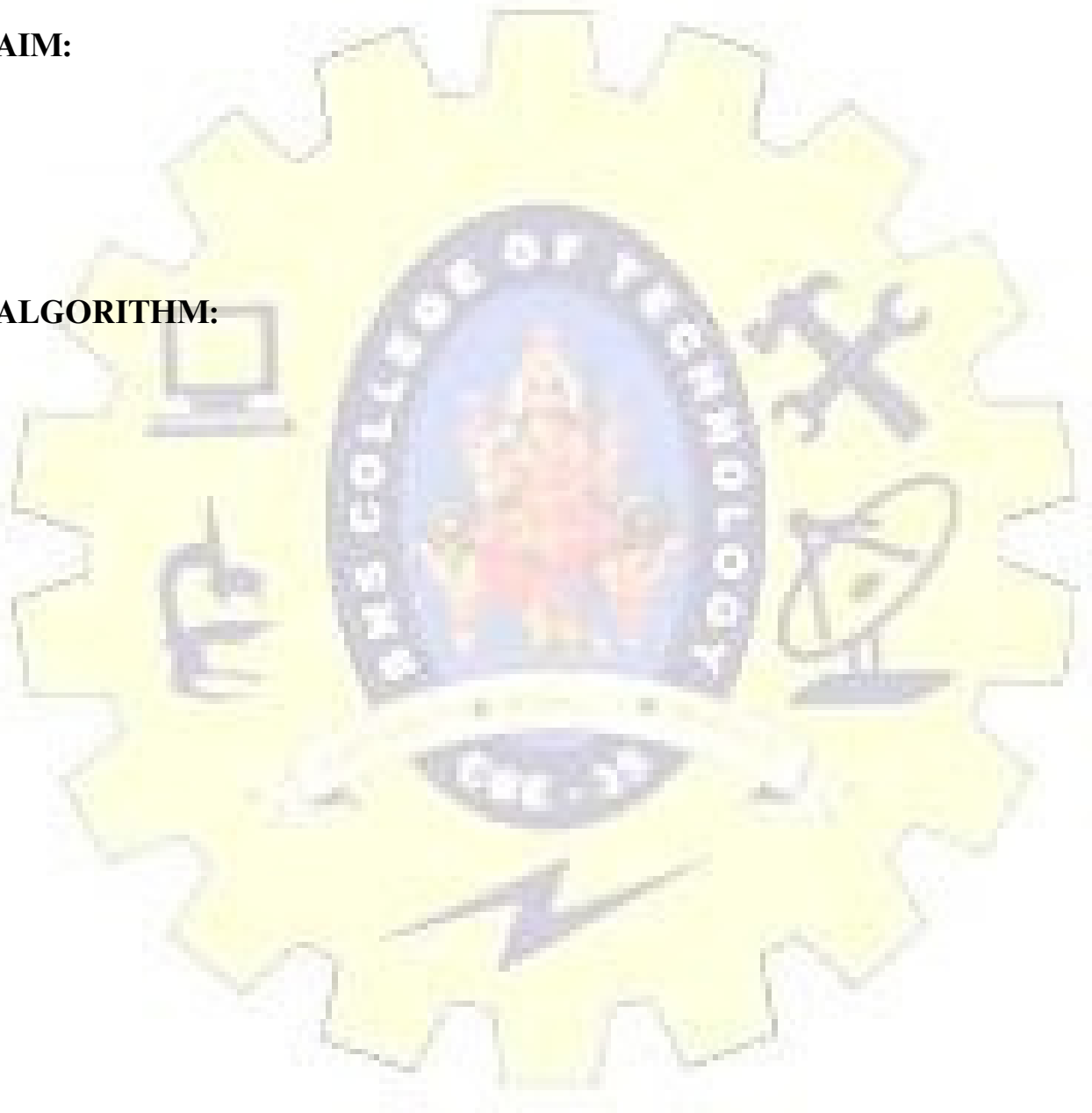
RESULT:

Ex. No. :

Date:


AIM:

ALGORITHM:



PROGRAM:

```
public class FizzBuzz {  
    public List<String> fizzBuzz(int n) {  
        List<String> result = new ArrayList<>();  
        for (int i = 1; i <= n; i++) {  
            if (i % 3 == 0 && i % 5 == 0) {  
                result.add("FizzBuzz");  
            } else if (i % 3 == 0) {  
                result.add("Fizz");  
            } else if (i % 5 == 0) {  
                result.add("Buzz");  
            } else {  
                result.add(String.valueOf(i));  
            }  
        }  
        return result;  
    }  
}
```



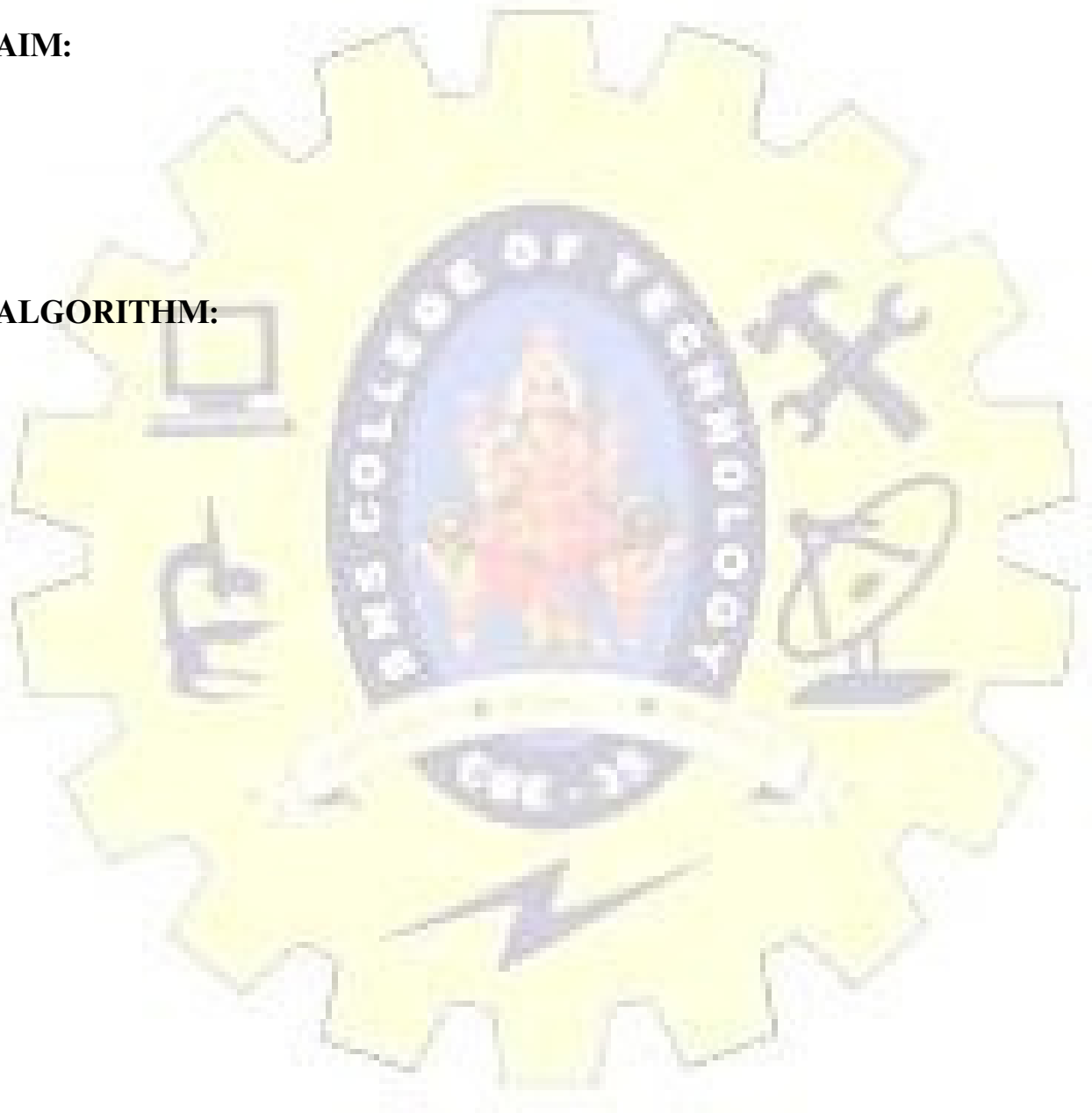
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
public class MonotonicArray {  
    public boolean isMonotonic(int[] A) {  
        boolean increasing = true, decreasing = true;  
        for (int i = 1; i < A.length; i++) {  
            if (A[i] > A[i - 1]) {  
                decreasing = false;  
            } else if (A[i] < A[i - 1]) {  
                increasing = false;  
            }  
        }  
        return increasing || decreasing;  
    }  
}
```

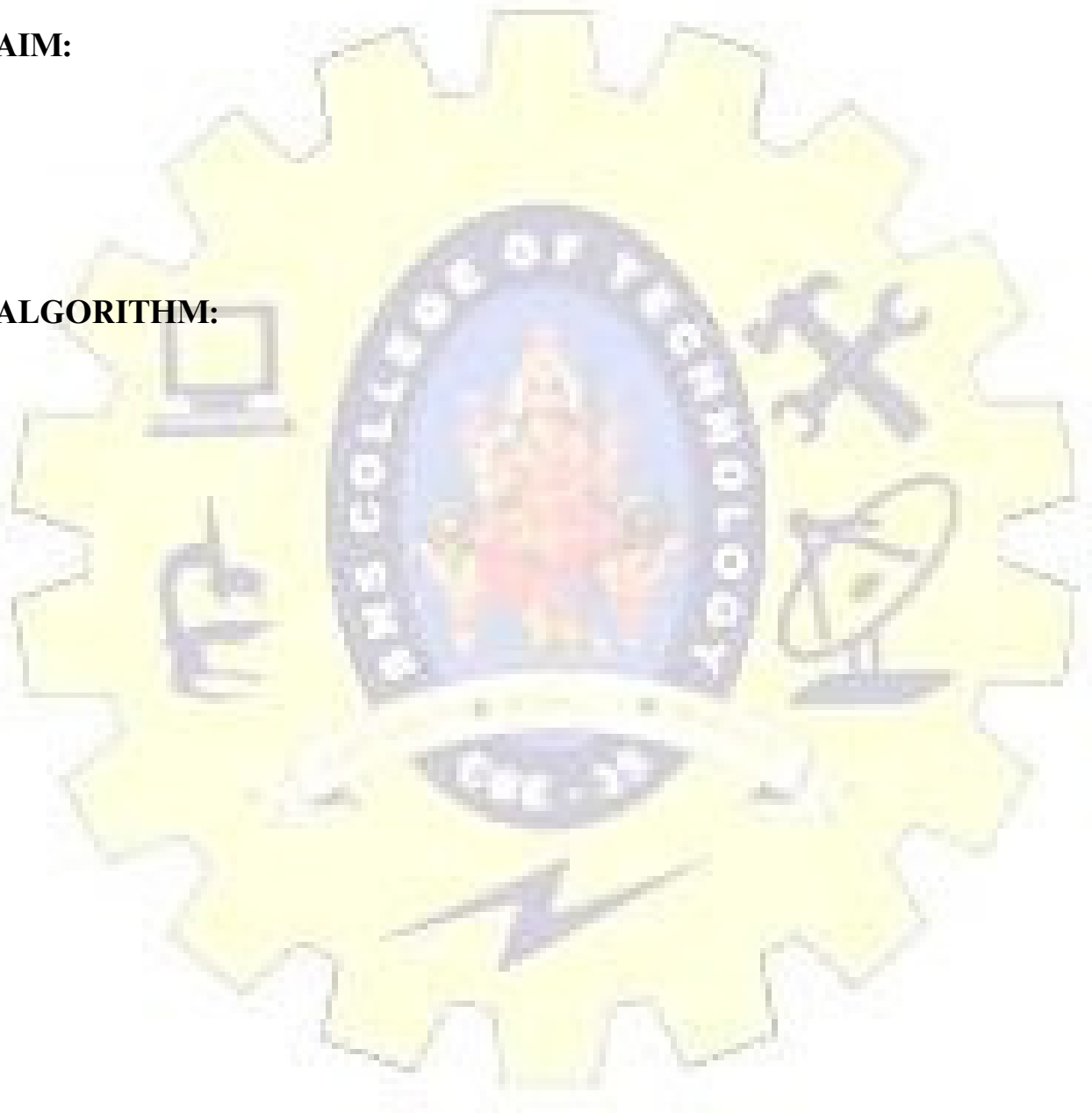
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
public class ContainerWithMostWater {  
    public int maxArea(int[] height) {  
        int left = 0, right = height.length - 1;  
        int maxArea = 0;  
  
        while (left < right) {  
            int width = right - left;  
            int minHeight = Math.min(height[left], height[right]);  
            maxArea = Math.max(maxArea, width * minHeight);  
  
            if (height[left] < height[right]) {  
                left++;  
            } else {  
                right--;  
            }  
        }  
        return maxArea;  
    }  
}
```

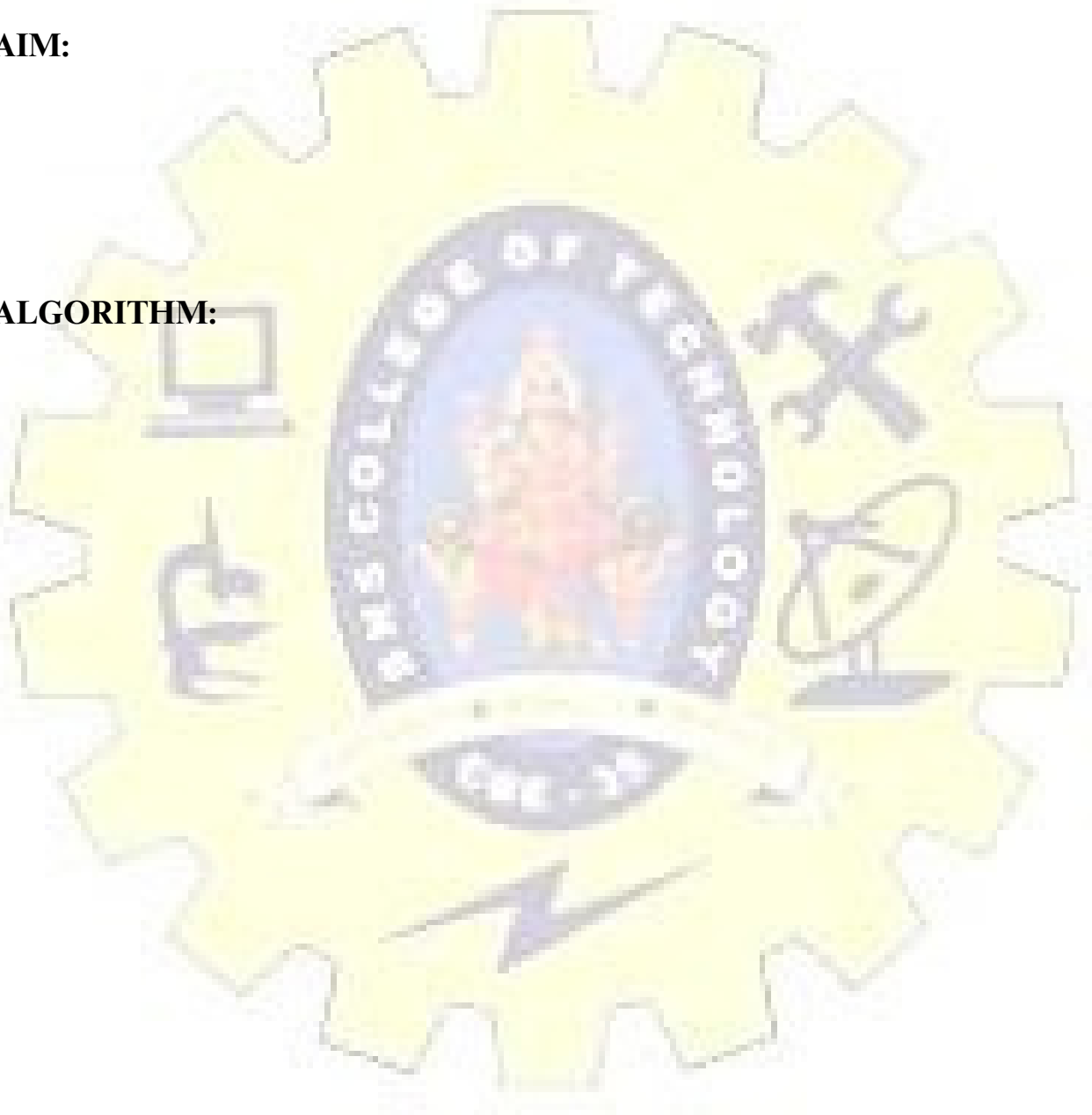
RESULT:

Ex. No. :

Date:

AIM:

ALGORITHM:



PROGRAM:

```
public class ParkingSystem {  
    private int big, medium, small;  
  
    public ParkingSystem(int big, int medium, int small) {  
        this.big = big;  
        this.medium = medium;  
        this.small = small;  
    }  
  
    public boolean addCar(int carType) {  
        if (carType == 1 && big > 0) {  
            big--;  
            return true;  
        } else if (carType == 2 && medium > 0) {  
            medium--;  
            return true;  
        } else if (carType == 3 && small > 0) {  
            small--;  
            return true;  
        }  
        return false;  
    }  
}
```

RESULT: