DSA Assignment 1

**Q1.Armstrong Number  
 Problem: Write a Java program to check if a given number is an Armstrong number.**

**>> Program :**   
package ADS\_Assignment\_1;

import java.util.Scanner;

public class ArmstrongNumber {

public static boolean isArmstrong(int num){

int original = num ;

int sum = 0 ;

int n = String.valueOf(num).length();

while (num != 0){

int digit = num % 10;

sum += Math.pow(digit,n);

num /=10;

}

return sum == original;

}

public static void main(String[] args) {

Scanner sc =new Scanner(System.in);

System.out.println("Enter a number : " );

int num = sc.nextInt();

System.out.println(isArmstrong(num));

sc.close();

}

}

**>> Flowchart:**

1.Start

2.Input the number

3.Store the number in a variable (original)

4. Count number of digits

5. Initialize sum to 0

6. For each digit of given number :

> Extract the last digit

> Raise the digit to the power of no of digits

> Add result to (sum)

> Remove the last digit from the number

7.If sum is equal to original,return TRUE otherwise return FALSE

8.End

**Time Complexity :**

>> O(d),where d is no of digits in the input number.

The program extracts each digit,raises it to the power of d , and sums the result.

**Space Complexity :**

>> O(1) : only few integer variables are used to store the result

Q2 Prime Number

Problem: Write a Java program to check if a given number is prime.

package Assignment1;

// question 2

import java.util.Scanner;

public class PrimeNo {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a number:");

int n=sc.nextInt();

boolean flag= false;

if(n==0 || n==1) {

flag=true;

}

for (int i = 2; i <= n / 2; ++i) {

if (n % i == 0) {

flag = true;

break;

}

}

if (!flag)

System.out.println(n + " is a prime number.");

else

System.out.println(n + " is not a prime number.");

sc.close();

}

}

**Flowchart**

1. Start
2. Input number n from user
3. Initialize flag to false
4. Check if n is 0 or 1
5. For loop from i=2 to n/2

Check if (n%i==0)

* If yes set flag= true and break the loop
* If no continue the loop

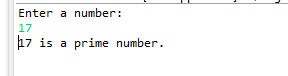
1. Chek if flag is false

If true, print n is a prime number

If false, print n is not a prime number

1. end

**Output**



Q3/// 3 .Problem: Write a Java program to compute the factorial of a given number.

//

//Test Cases:

//

//Input: 5

//Output: 120

//Input: 0

//Output: 1

**public** **class** Program {

**public** **static** **int** factorial1(**int** a) {

**int** res = 1;

**if**(a==0 || a==1)

**return** res;

**while**(a>1) {

res=res\*a;

a--;

}

**return** res;

}

**public** **static** **int** factorial(**int** a) { //

**if**(a==0 || a==1)

**return** 1;

**return** a=a\**factorial*(a-1); //recursion

}

**public** **static** **void** main(String[] args) {

**int** b=4;

System.***out***.println(*factorial*(b));

System.***out***.println(*factorial1*(b));

}

}

Time Complexity:O(n)

Space Complexity:O(1)

Q4 Fibonacci Series

Problem: Write a Java program to print the first n numbers in the Fibonacci series.

**Program**

import java.util.Scanner;

public class Fibonacci\_4 {

public static void main(String[] args) {

System.out.println("Enter no: ");

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int a = 0, b = 1;

for (int i = 0; i < n; i++) {

System.out.print(a + " ");

int next = a + b;

a = b;

b = next;

}

}

}

**Flowchart**

1. **Start**:
   * The program starts.
2. **Ask for Input**:
   * The program asks the user to enter a number (n).
3. **Get Input**:
   * The user enters a number, and the program reads it.
4. **Set Initial Values**:
   * Set the first two numbers of the Fibonacci sequence: a = 0 and b = 1.
5. **Loop Start**:
   * The program starts a loop that will run n times.
6. **Print the Number**:
   * The program prints the current value of a.
7. **Calculate the Next Fibonacci Number**:
   * Calculate the next number in the Fibonacci sequence by adding a + b.
8. **Update Values**:
   * Set a to the value of b, and set b to the new Fibonacci number (the one you just calculated).
9. **Repeat**:
   * The loop repeats until the program has printed n Fibonacci numbers.
10. **End**:
    * Once the loop finishes, the program ends.

**Output**

Enter no:

5

0 1 1 2 3

Enter no:

8

0 1 1 2 3 5 8 13

PS C:\Users\shubh>

**Time Complexity:** O(n) **Space Complexity:** O(1)

Q5. Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

**package** com.assignment;

**import** java.util.Scanner;

**public** **class** Solution5 {

**public** **static** **int** gcd(**int** a, **int** b) {

**if** (b == 0) **return** a; // Base case: if b is 0, return a

**return** *gcd*(b, a % b); // Recursive case: call gcd with (b, a % b)

}

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

**try** {

System.***out***.println("Enter two numbers:");

**int** a = sc.nextInt();

**int** b = sc.nextInt();

System.***out***.println("GCD: " + *gcd*(a, b));

} **finally** {

sc.close(); // Close the scanner

}

}

}

Enter two numbers:

24

54

GCD: 6

Enter two numbers:

17

13

GCD: 1

**Time Complexity**: O(log(min(a, b)))

**Space Complexity**: O(1)

Flowchart:  
1. Start: input a and b from the user

2. Call gcd(a,b)

3. Check if b==0: if b is 0, then base case is reached & function returns a.  
 **Yes**: If b==0, the GCD is a & the function returns the result.

**No:** If b!=0, the function calls itself recursively with the arguments gcd (b, a%b).

4. Recursion: continues until b becomes 0, and the GCD is found.

5. End

Q6

Q7

Q8. First Non-Repeated Character Problem: Write a Java program to find the first non-repeated character in a string.

**Q9 Integer Palindrome**

**Problem: Write a Java program to check if a given integer is a palindrome.**

import java.util.Scanner;

class IntPalindrome {

public static void main (String [] args){

Scanner scan = new Scanner(System.in);

System.out.println("Enter The Number To Check If It's a Palindrome or Not..");

int num = scan.nextInt();

int Revnum = 0;

int temp = num;

while (temp !=0) {

int digit = temp % 10;

Revnum = Revnum \*10 + digit;

temp = temp/10;

}

if (num == Revnum) {

System.out.println(" Yes, "+ num +" is a palindrome number");

} else {

System.out.println(" No, "+ num +" is not a palindrome number");

}

}

}

Time Complexity : *O*(log10)

Space Complexity : O(1)

**Flowchart:**

**Start:** Begin the algorithm.

**Input Number:** Prompt the user to enter a number and read the input value into the variable num.

**Initialize Variables:**

Set Revnum to 0 (this will hold the reversed number).

Set temp to num (this will be used to extract digits).

**Reverse the Number:**

While temp is not equal to 0, repeat the following steps:

Extract the last digit of temp using digit = temp % 10.

Update Revnum by multiplying it by 10 and adding the extracted digit: Revnum = Revnum \* 10 + digit.

Remove the last digit from temp by performing integer division by 10: temp = temp / 10.

Check for Palindrome:

If num is equal to Revnum, then:

Print "Yes, num is a palindrome number."

**Otherwise:**

Print "No, num is not a palindrome number."

**End:** Terminate the algorithm.

**Q10 Leap Year**

**Problem: Write a Java program to check if a given year is a leap year.**

import java.util.Scanner;

class LeapYear {

public static void main (String [] args){

Scanner scan = new Scanner(System.in);

System.out.print("Enter The Year To Check If It's Leap Year Or Not..");

int year = scan.nextInt();

//System.out.println("The Year Is : "+year);

if(year%4 == 0 || year%400 ==0){

System.out.println("The Year "+year+" is a Leap Year");

} else{

System.out.println("The Year "+year+ " is not a Leap Year");

}

}

}

Time Complexity : O(1)

Space Complexity : O(1)

**Flowchart:  
Start:** Begin the algorithm.

**Input Year:** Prompt the user to enter a year and read the input value into the variable year.

**Check Conditions:**

**Condition 1:** Check if year is divisible by 4.

If true, proceed to Condition 2.

If false, go to Step 5 (not a leap year).

**Condition 2:** If false, go to next condition (it is a leap year).

**Condition 3:** Check if year is divisible by 400.

If true, go to Step 4 (it is a leap year).

If false, go to Step 5 (not a leap year).

**Output Leap Year:** Print "The Year year is a Leap Year."

**Output Not Leap Year:** Print "The Year year is not a Leap Year."

**End:** Terminate the algorithm.

—-----------------

—--------------------------

**public** **static** **boolean** armstrong(**int** a) {

**int** num=a;

**int** sum=0;

**int** power=String.*valueOf*(a).length();

**while**(a>0) {

**int** d=a%10;

a=a/10;

sum=(**int**) (sum+Math.*pow*(d,power));

}

**return** num==sum;

}

System.out.println(armstrong(153));

ystem.out.println(armstrong(123));