

TASK 1: Access and print the element at a given index in an array.

AIM : To access and print the element present at a given index in an array.

ALGORITHM;

1. Start
2. Declare and initialize an array
3. Read the index value
4. Check if index is valid ($0 \leq \text{index} < \text{array length}$)
5. Print the element at that index
6. Stop

PROGRAM:

```
import java.util.Scanner;

public class AccessElement {

    public static void main(String[] args)

    { int[] arr = {10, 20, 30, 40, 50};

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter index: ");

    int index = sc.nextInt();

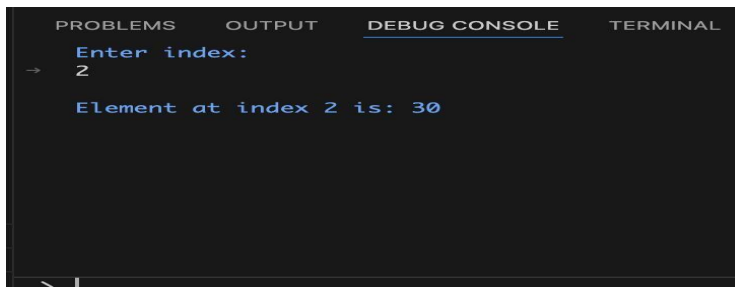
    if (index >= 0 && index < arr.length) {

        System.out.println("Element at index " + index + " is: " + arr[index]);

    } else {

        System.out.println("Invalid index");} } }
```

OUTPUT:



```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
Enter index:
→ 2
Element at index 2 is: 30
> |
```

RESULT: Thus is an java program to access and print the element present at a given index in an array has been done successfully

TASK 2: Search for a given element in a sorted array using Binary Search

AIM : To search an element in a sorted array using Binary Search technique.

ALGORITHM;

1. Start
2. Declare a sorted array
3. Read the element to search
4. Set $low = 0$ and $high = n - 1$ and Find $mid = (low + high) / 2$
5. If $element == arr[mid]$, print position
6. If $element < arr[mid]$, set $high = mid - 1$ then Else set $low = mid + 1$
7. Repeat until $low > high$ then If not found, print "Element not found"
8. Stop

PROGRAM:

```
import java.util.Scanner;

public class BinarySearch {

    public static void main(String[] args)

    { int[] arr = {5, 10, 15, 20, 25, 30};

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter element to search: ");

        int key = sc.nextInt();

        int low = 0, high = arr.length - 1;

        boolean found = false;

        while (low <= high) {

            int mid = (low + high) / 2; if (arr[mid] == key)

                { System.out.println("Element found at index: " + mid);

                    found = true;

                    break;

                } else if (key < arr[mid])

                { high = mid - 1;

                } else {

                    low = mid + 1;

                }

        }
```

```
    }

    if(!found) {


        System.out.println("Element not found");

    }

}

}
```

OUTPUT:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Enter element to search:
→ 20

Element found at index: 3

> |
```

RESULT: Thus is an java program to search an element in a sorted array using Binary Search technique.has been done successfully

TASK 3: Find the maximum element in an array of n integers

AIM : To find the maximum element in an array.

ALGORITHM;

1. Start
2. Declare and initialize array
3. Set `max = arr[0]`
4. Compare each element with `max`
5. If `element > max`, update `max`
6. Print `max` value
7. Stop

PROGRAM:

```
import java.util.Scanner;

public class MaxElement {

    public static void main(String[] args)

        { int[] arr = {12, 45, 7, 89, 34};

        int max = arr[0];

        for (int i = 1; i < arr.length; i++)

            { if (arr[i] > max) {

                max = arr[i];}

            } System.out.println("Maximum element is: " + max)

        } }
```

OUTPUT:

A screenshot of an IDE window with a dark theme. At the top, there are four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'OUTPUT' tab is selected and highlighted. Below the tabs, the text 'Maximum element is: 89' is displayed in a light blue font. At the bottom of the window, there is a prompt character '>'.

RESULT: Thus is an java program to search an element in a sorted array using Binary Search technique.has been done successfully

TASK 4: Find the Kth smallest element in an array

AIM : To Find the Kth smallest element in an array

ALGORITHM;

1. Start
2. Declare array and value of K
3. Sort the array
4. The Kth smallest element is at index $K - 1$
5. Print the element
6. Stop

PROGRAM:

```
import java.util.Arrays;

import java.util.Scanner;

public class KthSmallest {

    public static void main(String[] args)

    { int[] arr = {7, 10, 4, 3, 20, 15};

    Scanner sc = new Scanner(System.in);

    System.out.print("Enter K: ");

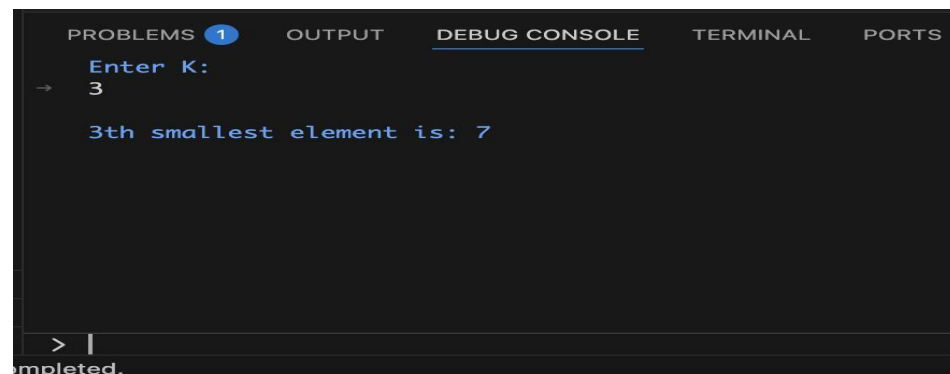
    int k = sc.nextInt();

    Arrays.sort(arr);

    if (k > 0 && k <= arr.length) { System.out.println(k + "th smallest element is: " + arr[k - 1]);

    } else {    System.out.println("Invalid value of K") } }
```

OUTPUT:

A screenshot of an IDE's DEBUG CONSOLE window. The window has tabs for PROBLEMS (1), OUTPUT, DEBUG CONSOLE (selected), TERMINAL, and PORTS. The DEBUG CONSOLE shows the program's execution: it prompts "Enter K:" and receives the input "3". It then outputs "3th smallest element is: 7". At the bottom, a prompt "> |" is visible, and the word "Completed." is partially visible at the very bottom edge.

RESULT: Thus is an java program to Find the Kth smallest element in an array.has been done successfully

TASK 5: Print all possible pairs of elements from an array

AIM : To Print all possible pairs of elements from an array

ALGORITHM;

1. Start
2. Declare and initialize array
3. Use two nested loops
4. First loop from $i = 0$ to $n-1$
5. Second loop from $j = i+1$ to $n-1$
6. Print (`arr[i], arr[j]`)
7. Stop

PROGRAM:

```
import java.util.Scanner;

public class ArrayPairs {

    public static void main(String[] args)

    { int[] arr = {1, 2, 3, 4};

    System.out.println("All possible pairs:");

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

        for (int j = i + 1; j < arr.length; j++)

            { System.out.println("(" + arr[i] + ", " + arr[j] + ")");}}

    }
```

OUTPUT:



RESULT: Thus is an java program to print all possible pairs of elements from an array.has been done successfully

TASK6: : Digit Sum opt sum of even or odd digits

AIM: To find the sum of either even digits or odd digits of a given number based on user choice.

ALGORITHM:

1. Start
2. Read a number n
3. Read option (1 for even digits, 2 for odd digits)
4. Initialize $sum = 0$
5. While $n > 0$
 - Get digit $d = n \% 10$
 - If option = 1 and d is even, add to sum
 - If option = 2 and d is odd, add to sum
 - Update $n = n / 10$
6. Print sum
7. Stop

PROGRAM:

```
import java.util.Scanner;

public class DigitSum {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number: ");

        int n = sc.nextInt();

        System.out.print("Enter option (1-Even, 2-Odd): ");

        int opt = sc.nextInt();

        int sum = 0, digit;

        while (n > 0) {

            digit = n % 10;

            if (opt == 1 && digit % 2 == 0)

                sum += digit;

            else if (opt == 2 && digit % 2 != 0)

                sum += digit;

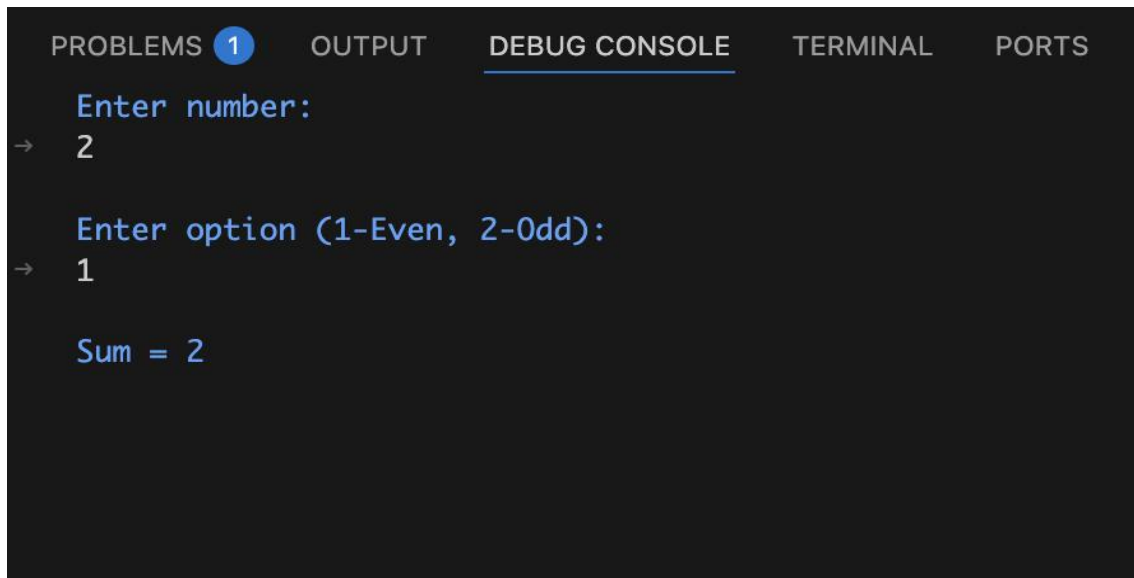
            n = n / 10;

        }

        System.out.println("Sum of " + (opt == 1 ? "even" : "odd") + " digits is: " + sum);
    }
}
```

```
    }  
  
    System.out.println("Sum = " + sum);  
  
    }  
  
}
```

OUTPUT:



The screenshot shows a dark-themed IDE window with a tab bar at the top containing 'PROBLEMS 1', 'OUTPUT', 'DEBUG CONSOLE' (which is underlined), 'TERMINAL', and 'PORTS'. The main area displays the program's output in a monospaced font. It starts with the prompt 'Enter number:' followed by the user input '2' on the next line. Then, it shows 'Enter option (1-Even, 2-Odd):' followed by the user input '1' on the next line. Finally, it displays the result 'Sum = 2'.

```
Enter number:  
→ 2  
  
Enter option (1-Even, 2-Odd):  
→ 1  
  
Sum = 2
```

RESULT: Thus is an java program to find the sum of either even digits or odd digits of a given number based on user choice has been done successfully

TASK 7 : Nth Fibonacci Number

AIM: To find the Nth Fibonacci number.

ALGORITHM:

1. Start
2. Read value of n
3. Initialize $a = 0, b = 1$
4. Repeat from 2 to n
 - $c = a + b$
 - $a = b, b = c$
5. Print Fibonacci number
6. Stop

PROGRAM :

```
import java.util.Scanner;

public class Fibonacci {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter n: ");

        int n = sc.nextInt();

        int a = 0, b = 1, c = 0;

        if (n == 0)

            System.out.println(a);

        else if (n == 1)

            System.out.println(b);

        else {

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

                c = a + b;

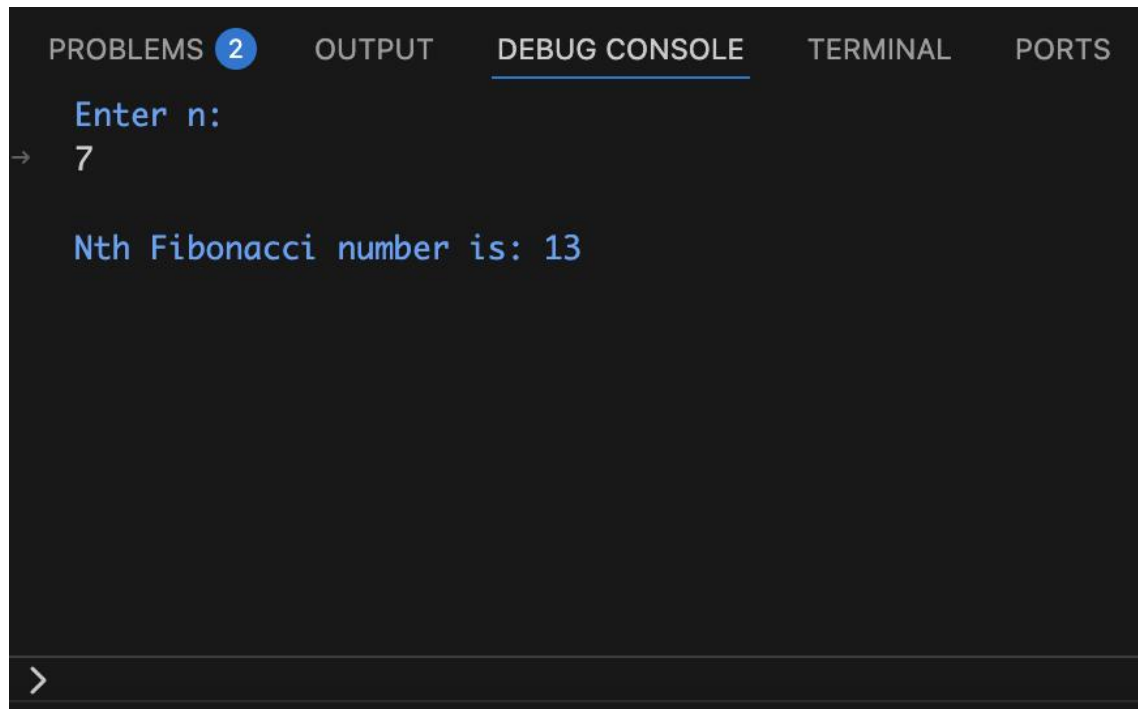
                a = b;

                b = c;

            }

            System.out.println("Nth Fibonacci number is: " + c);}}}
```

OUTPUT:



The screenshot shows a dark-themed interface with a horizontal menu at the top containing five items: 'PROBLEMS' with a blue circle containing the number '2', 'OUTPUT', 'DEBUG CONSOLE' (which is underlined), 'TERMINAL', and 'PORTS'. Below the menu, the text 'Enter n:' is displayed in a light blue font. A light blue arrow points to the number '7' on the next line. On the following line, the text 'Nth Fibonacci number is: 13' is displayed in the same light blue font. At the bottom left of the console area, there is a white right-pointing chevron '>'. The entire console area has a dark gray background.

```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Enter n:
→ 7
Nth Fibonacci number is: 13
>
```

RESULT: Thus is an java program to find the Nth Fibonacci number.
has been done successfully

TASK 8 : Check Whether a Number is Palindrome

AIM: To check whether a given number is a palindrome.

ALGORITHM:

1. Start
2. Read number n
3. Store original number
4. Reverse the number
5. Compare reversed number with original
6. Stop

PROGRAM;

```
import java.util.Scanner;

public class PalindromeNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

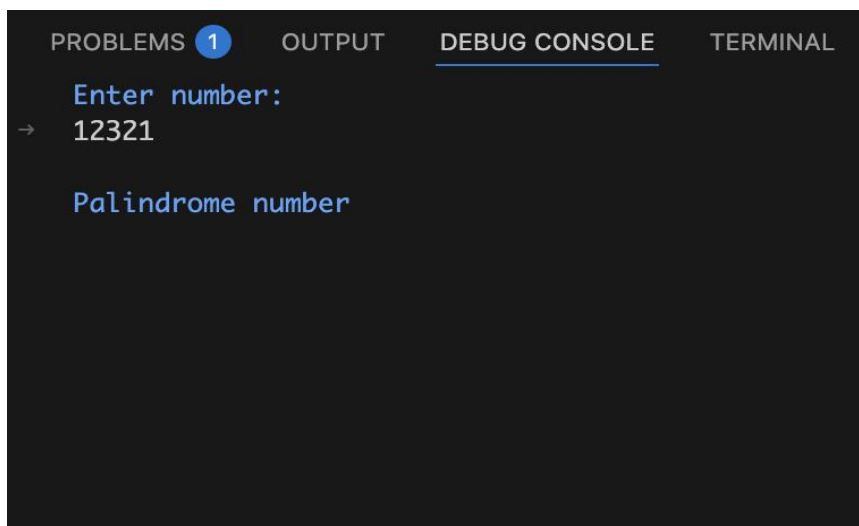
        System.out.print("Enter number: ");
        int n = sc.nextInt();

        int original = n, reverse = 0, digit;

        while (n > 0) {
            digit = n % 10;
            reverse = reverse * 10 + digit;
            n = n / 10;
        }

        if (original == reverse)
            System.out.println("Palindrome number");
        else
            System.out.println("Not a palindrome number");
    }
}
```

OUTPUT:



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

Enter number:
→ 12321

Palindrome number
```

RESULT: Thus is an java program to check whether a given number is a palindrome. has been done successfully

TASK 9 Sum of Last Digit of Two Given Numbers

AIM: To find the sum of the last digits of two given numbers.

ALGORITHM:

1. Start
2. Read two numbers a and b
3. Find last digits using % 10
4. Add last digits
5. Print sum
6. Stop

PROGRAM

```
import java.util.Scanner;

public class LastDigitSum {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

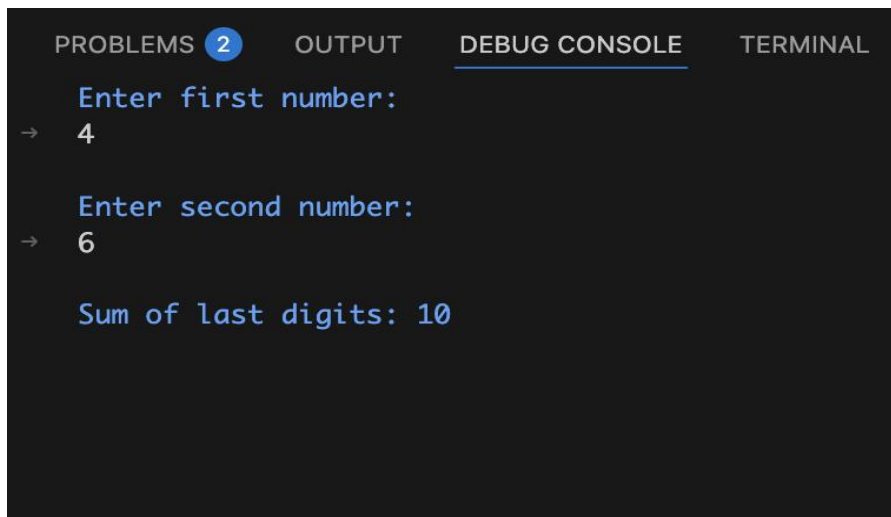
        System.out.print("Enter first number: ");
        int a = sc.nextInt();

        System.out.print("Enter second number: ");
        int b = sc.nextInt();

        int sum = (a % 10) + (b % 10);

        System.out.println("Sum of last digits: " + sum);
    }
}
```

OUTPUT



```
PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

Enter first number:
→ 4

Enter second number:
→ 6

Sum of last digits: 10
```

RESULT: Thus is an java program to find the sum of the last digits of two given numbers. has been done successfully