📝 Complete Programming Problem Package

Print Even Numbers from Arrays

Table of Contents

Problem Statement

📌 Problem Description

You are given multiple arrays. For each array, print all the even numbers present in the array in the order they appear.

📥 Input Format

- The first line contains an integer `m` — the number of test cases.
- For each test case:
  - The first line contains an integer `n` — the size of the array.
  - The second line contains `n` space-separated integers — the elements of the array.

📤 Output Format

For each test case, print all even numbers from the array separated by spaces. Print a newline after each test case.

❗ Constraints

- $1 \leq m \leq 100$
- $1 \leq n \leq 1000$
- Each element of the array is an integer between -10^6 and 10^6.

🛠️ Notes

- If there are no even numbers in a test case, print an empty line.
- Negative even numbers should also be considered (e.g., -2, -4, etc.).
- Zero (0) is considered an even number.

🎯 Learning Objectives

- Array traversal and filtering
- Conditional logic implementation
- Input/output handling for multiple test cases
- Understanding even number identification

📊 Difficulty Level

Beginner - Suitable for those learning basic programming concepts and array manipulation.

Sample Input/Output

🧪 Sample Input

```
2
5
1 2 3 4 5
4
10 11 12 13
```

✅ Sample Output

```
2 4
10 12
```

💡 Explanation

- For the first array `[1, 2, 3, 4, 5]`, the even numbers are `2` and `4`.

- For the second array `[10, 11, 12, 13]`, the even numbers are `10` and `12`.

Solutions

C++ Solution

cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int m;
    cin >> m;

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

        vector<int> arr(n);
        for (int j = 0; j < n; j++) {
            cin >> arr[j];
        }

        // Print even numbers
        bool first = true;
        for (int j = 0; j < n; j++) {
            if (arr[j] % 2 == 0) {
                if (!first) cout << " ";
                cout << arr[j];
                first = false;
            }
        }
        cout << endl;
    }

    return 0;
}

/*
Time Complexity: O(m * n) where m is number of test cases and n is array size
Space Complexity: O(n) for storing each array

Algorithm:
1. Read number of test cases m
2. For each test case:
   - Read array size n
   - Read n elements into array
   - Iterate through array and print even numbers with spaces
   - Print newline after each test case
```

```
*3. Handle edge case of no even numbers (prints empty line)*
*/
```

Python Solution

python

```python
def solve():
    m = int(input())

    for _ in range(m):
        n = int(input())
        arr = list(map(int, input().split()))

        # Find and print even numbers
        even_numbers = [num for num in arr if num % 2 == 0]

        if even_numbers:
            print(' '.join(map(str, even_numbers)) + ' ')
        else:
            print()  # Empty line if no even numbers

# Alternative solution using direct iteration
def solve_alternative():
    m = int(input())

    for _ in range(m):
        n = int(input())
        arr = list(map(int, input().split()))

        # Print even numbers directly
        even_nums = []
        for num in arr:
            if num % 2 == 0:
                even_nums.append(str(num))

        print(' '.join(even_nums) + ' ')

if __name__ == "__main__":
    solve()

"""
Time Complexity: O(m * n) where m is number of test cases and n is array size
Space Complexity: O(n) for storing arrays and even numbers list

Algorithm:
1. Read number of test cases
2. For each test case:
   - Read array size and elements
   - Filter even numbers using list comprehension
```

```
    - Print them with proper spacing
    - Handle empty case with blank line

Python-specific optimizations:
- List comprehension for filtering
- map() for type conversion
- join() for efficient string concatenation
"""
```

Java Solution

java

```java
import java.util.*;

public class PrintEvenNumbers {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int m = scanner.nextInt();

        for (int i = 0; i < m; i++) {
            int n = scanner.nextInt();
            int[] arr = new int[n];

            // Read array elements
            for (int j = 0; j < n; j++) {
                arr[j] = scanner.nextInt();
            }

            // Print even numbers
            StringBuilder result = new StringBuilder();
            for (int j = 0; j < n; j++) {
                if (arr[j] % 2 == 0) {
                    result.append(arr[j]).append(" ");
                }
            }

            System.out.println(result.toString());
        }

        scanner.close();
    }
}

/*
Alternative approach using ArrayList for dynamic storage:

import java.util.*;

public class PrintEvenNumbersAlt {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int m = scanner.nextInt();

        for (int i = 0; i < m; i++) {
            int n = scanner.nextInt();
```

```
        ArrayList<Integer> evenNumbers = new ArrayList<>();

        // Read and filter even numbers
        for (int j = 0; j < n; j++) {
            int num = scanner.nextInt();
            if (num % 2 == 0) {
                evenNumbers.add(num);
            }
        }

        // Print even numbers
        for (int k = 0; k < evenNumbers.size(); k++) {
            System.out.print(evenNumbers.get(k));
            if (k < evenNumbers.size() - 1) {
                System.out.print(" ");
            }
        }
        System.out.println(" ");
    }

    scanner.close();
  }
}

Time Complexity: O(m * n)
Space Complexity: O(n)
*/
```

JavaScript Solution

javascript

```javascript
// Node.js solution using readline
const readline = require('readline');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

let input = [];
let lineIndex = 0;

rl.on('line', (line) => {
    input.push(line.trim());
});

rl.on('close', () => {
    solve();
});

function solve() {
    const m = parseInt(input[lineIndex++]);

    for (let i = 0; i < m; i++) {
        const n = parseInt(input[lineIndex++]);
        const arr = input[lineIndex++].split(' ').map(Number);

        // Filter and print even numbers
        const evenNumbers = arr.filter(num => num % 2 === 0);

        if (evenNumbers.length > 0) {
            console.log(evenNumbers.join(' ') + ' ');
        } else {
            console.log('');
        }
    }
}

// Alternative browser-compatible solution
function solveBrowser(inputString) {
    const lines = inputString.trim().split('\n');
    let lineIndex = 0;
    const results = [];
```

```
    const m = parseInt(lines[lineIndex++]);

    for (let i = 0; i < m; i++) {
        const n = parseInt(lines[lineIndex++]);
        const arr = lines[lineIndex++].split(' ').map(Number);

        const evenNumbers = [];
        for (let j = 0; j < arr.length; j++) {
            if (arr[j] % 2 === 0) {
                evenNumbers.push(arr[j]);
            }
        }

        results.push(evenNumbers.join(' ') + ' ');
    }

    return results.join('\n');
}

// ES6+ Functional approach
function solveFunctional(inputString) {
    const lines = inputString.trim().split('\n');
    const m = parseInt(lines[0]);

    return Array.from({ length: m }, (_, i) => {
        const n = parseInt(lines[1 + i * 2]);
        const arr = lines[2 + i * 2].split(' ').map(Number);

        return arr
            .filter(num => num % 2 === 0)
            .join(' ') + ' ';
    }).join('\n');
}

/*
Time Complexity: O(m * n) where m is test cases and n is array size
Space Complexity: O(n) for array storage

Key JavaScript features used:
- Array.filter() for functional programming approach
- Array.map() for type conversion
- Template literals for string formatting
- ES6+ arrow functions and destructuring
- Multiple solution approaches for different environments
```

```
*Browser vs Node.js considerations:
- Node.js uses readline for input handling
- Browser version works with string input
- Both handle the same core algorithm
*/
```

Test Cases

Test Case 1 (Sample)

Input:

```
2
5
1 2 3 4 5
4
10 11 12 13
```

Expected Output:

```
2 4
10 12
```

Test Case 2 (Edge Cases)

Input:

```
5
3
1 3 5
1
0
4
-4 -3 -2 -1
6
2 4 6 8 10 12
2
-100 100
```

Expected Output:

```
0
-4 -2
2 4 6 8 10 12
-100 100
```

Test Case 3 (Mixed Numbers)

Input:

```
3
7
-10 -5 0 5 10 15 20
5
1 1 1 1 1
8
-6 -4 -2 0 2 4 6 8
```

Expected Output:

```
-10 0 10 20

-6 -4 -2 0 2 4 6 8
```

Test Case 4 (Large Numbers)

Input:

```
2
6
999999 1000000 -999999 -1000000 500000 -500000
4
123456 234567 345678 456789
```

Expected Output:

```
1000000 -1000000 500000 -500000
123456 345678
```

Test Case 5 (Single Elements)

Input:

```
4
1
2
1
3
1
-8
1
0
```

Expected Output:

```
2


-8
0
```

Test Case 6 (Maximum Constraints)

Input:

```
3
10
1 2 3 4 5 6 7 8 9 10
8
-8 -7 -6 -5 -4 -3 -2 -1
12
0 1 0 2 0 3 0 4 0 5 0 6
```

Expected Output:

```
2 4 6 8 10
-8 -6 -4 -2
0 0 2 0 0 4 0 0 6
```

Notes for Testing:

1. **Empty result case**: Test case with all odd numbers
2. **Zero handling**: Zero should be treated as even
3. **Negative numbers**: Negative even numbers should be included
4. **Single element arrays**: Edge case for minimum array size
5. **Large numbers**: Test within constraint limits
6. **All even numbers**: Array with only even numbers
7. **Mixed positive/negative**: Combination of positive and negative even numbers

Validation Points:

- Each line should end with a space followed by newline
- Empty lines should still have a newline character
- Order of even numbers should match their appearance in input
- Handle edge cases gracefully
- Proper spacing between numbers

Editorial & Explanation

🎯 Problem Analysis

This is a fundamental array processing problem that tests:

- **Array traversal** - Iterating through elements
- **Conditional logic** - Identifying even numbers
- **Output formatting** - Proper spacing and newlines
- **Multiple test case handling** - Processing input efficiently

🔍 Key Insights

What makes a number even?

A number is even if it's divisible by 2 with no remainder:

- `n % 2 == 0` returns `true` for even numbers
- This works for positive, negative, and zero

Critical Edge Cases

1. **Empty result**: Array with no even numbers → print empty line

2. **Zero**: `0 % 2 == 0` → zero is even
3. **Negative numbers**: `-4 % 2 == 0` → negative evens count
4. **Single elements**: Handle arrays of size 1

🚀 Solution Approaches

Approach 1: Direct Filtering (Recommended)

```
For each test case:
1. Read array elements
2. Iterate through array
3. If element % 2 == 0, add to result
4. Print results with proper formatting
```

Pros: Simple, efficient, easy to understand

Cons: None for this problem size

Approach 2: Two-Pass Solution

```
Pass 1: Count even numbers
Pass 2: Print even numbers
```

Pros: Can pre-allocate result array

Cons: Unnecessary complexity for this problem

💡 Implementation Tips

1. Output Formatting

- Each line should end with a space followed by newline
- Empty lines still need newline character
- Maintain original order of even numbers

2. Modulo Operation

```cpp
```

```
// Correct way to check even
if (num % 2 == 0) {
    // num is even
}

// Note: In some languages, negative modulo behavior varies
// But for even/odd checking, num % 2 == 0 works universally
```

3. Input/Output Efficiency

- Use fast I/O methods for large inputs
- StringBuilder/StringBuffer for string concatenation
- Read entire input at once when possible

🔧 Language-Specific Optimizations

C++

cpp

```
// Fast I/O
ios_base::sync_with_stdio(false);
cin.tie(NULL);

// Use vectors for dynamic sizing
vector<int> evenNums;
```

Python

python

```
# List comprehension for filtering
even_nums = [x for x in arr if x % 2 == 0]

# sys.stdin for faster input (if needed)
import sys
input = sys.stdin.readline
```

Java

java

```
// StringBuilder for string building
StringBuilder result = new StringBuilder();

// Enhanced for loop
for (int num : arr) {
    if (num % 2 == 0) {
        result.append(num).append(" ");
    }
}
```

📊 Complexity Analysis

Time Complexity: O(m × n)

- m = number of test cases
- n = average array size
- Must examine each element once

Space Complexity: O(n)

- For storing input array
- Output space not counted in auxiliary space
- Can optimize to O(1) by processing elements as read

🐛 Common Mistakes

1. Wrong Even Check

cpp

```
// WRONG: Only works for positive numbers in some languages
if (num % 2 == 1) // This is checking odd!

// CORRECT
if (num % 2 == 0)
```

2. Output Formatting Issues

cpp

```
// WRONG: Extra space handling
cout << num << " ";
if (last_element) cout << "\n"; // Missing space at end

// CORRECT: Consistent spacing
cout << num << " ";
cout << "\n"; // Always space then newline
```

3. Ignoring Empty Case

python

```
# WRONG: No output for empty case
if even_numbers:
    print(' '.join(map(str, even_numbers)))
# Missing else case!

# CORRECT: Handle empty case
if even_numbers:
    print(' '.join(map(str, even_numbers)) + ' ')
else:
    print()  # Empty line
```

🎓 Learning Extensions

Easy Extensions

1. Print odd numbers instead
2. Count even/odd numbers
3. Find sum of even numbers

Medium Extensions

1. Print even numbers in reverse order
2. Remove duplicates while printing
3. Sort even numbers before printing

Hard Extensions

1. Find kth largest even number
2. Group even numbers by frequency

3. Print even numbers from 2D arrays

🏆 Best Practices

1. **Read problem constraints carefully** - affects choice of data types
2. **Handle edge cases explicitly** - empty arrays, single elements
3. **Use appropriate data structures** - vectors for dynamic, arrays for fixed
4. **Test with provided samples** - verify output format matches exactly
5. **Consider time limits** - O(m×n) is efficient for given constraints

📝 Template Code Structure

cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    // Fast I/O (optional)
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    int m;
    cin >> m;

    while (m--) {
        int n;
        cin >> n;

        // Process each test case
        bool first = true;
        for (int i = 0; i < n; i++) {
            int num;
            cin >> num;

            if (num % 2 == 0) {
                if (!first) cout << " ";
                cout << num;
                first = false;
            }
        }
        cout << " \n";  // Space then newline
    }

    return 0;
}
```

Learning Resources

🎯 Related Topics to Study

- **Array Processing**: Fundamental data structure operations
- **Modular Arithmetic**: Understanding remainder operations
- **Input/Output Handling**: Efficient I/O for competitive programming

- **Time Complexity Analysis**: Big O notation and algorithm analysis

📚 Next Steps

1. **Practice Similar Problems**: Array filtering, number theory basics
2. **Learn Advanced Techniques**: Two pointers, sliding window
3. **Explore Data Structures**: Vectors, lists, dynamic arrays
4. **Study Algorithms**: Sorting, searching, basic number theory

🎯 Problem Variations

- Filter numbers by different criteria (divisible by 3, prime numbers, etc.)
- Process 2D arrays or matrices
- Handle multiple conditions simultaneously
- Optimize for memory-constrained environments

📄 Document Information

- **Created**: Programming Problem Package
- **Topic**: Print Even Numbers from Arrays
- **Difficulty**: Beginner
- **Languages**: C++, Python, Java, JavaScript
- **Complete Package**: Problem statement, solutions, test cases, editorial

---

This document contains everything needed to understand, solve, and teach the "Print Even Numbers from Arrays" programming problem. Save this file with a `.md` extension and convert to PDF using any markdown-to-PDF converter.