Largest Number Finder

A C program that finds the maximum value in an array and identifies all indices where this maximum occurs.

Algorithm Overview

The program processes multiple test cases, where each test case:

1. Reads `m` (number of test cases)
2. For each test case:
   - Reads `n` (array size)
   - Reads `n` integers into array `a[]`
   - Finds the maximum value in the array
   - Prints the maximum value
   - Prints all indices where the maximum value appears
   - Continues to the next test case

Example

Input:

```
2
5
1 2 3 2 3
4
10 10 5 10
```

Output:

```
3
2 4
10
0 1 3
```

Explanation:

- **Test case 1**: Maximum = 3, found at indices 2 and 4
- **Test case 2**: Maximum = 10, found at indices 0, 1, and 3

Implementation

```c
#include <stdio.h>

int main() {
    int m;
    scanf("%d", &m);
    while (m--) {
        int n;
        scanf("%d", &n);
        int a[n];
        for (int i = 0; i < n; i++)
            scanf("%d", &a[i]);

        int max = a[0];
        for (int i = 1; i < n; i++)
            if (a[i] > max)
                max = a[i];

        printf("%d\n", max);
        for (int i = 0; i < n; i++)
            if (a[i] == max)
                printf("%d ", i);
        printf("\n");
    }
    return 0;
}
```

Code Explanation

Step-by-Step Breakdown

1. Reading Test Cases

```c
int m;
scanf("%d", &m);
while (m--) {
```

- Reads the number of test cases `m`

- Uses `while (m--)` loop to process each test case, decrementing `m` until it reaches 0

2. Array Input

```c
int n;
scanf("%d", &n);
int a[n];
for (int i = 0; i < n; i++)
    scanf("%d", &a[i]);
```

- Reads array size `n`
- Declares a variable-length array `a[n]` (C99 feature)
- Fills the array with `n` integers using a for loop

3. Finding Maximum Value

```c
int max = a[0];
for (int i = 1; i < n; i++)
    if (a[i] > max)
        max = a[i];
```

- Initializes `max` with the first element `a[0]`
- Iterates through remaining elements starting from index 1
- Updates `max` whenever a larger element is found
- After this loop, `max` contains the largest value in the array

4. Output Results

```c
printf("%d\n", max);
for (int i = 0; i < n; i++)
    if (a[i] == max)
        printf("%d ", i);
printf("\n");
```

- First prints the maximum value

- Then scans the entire array again to find all indices where the value equals `max`
- Prints each matching index followed by a space
- Adds a newline after all indices are printed

Algorithm Logic

The program uses a **two-pass approach**:

1. **First pass**: Find the maximum value by comparing each element
2. **Second pass**: Collect all indices where this maximum value occurs

This approach is efficient because:

- It only requires two linear scans of the array
- No additional storage needed for tracking indices during the first pass
- Simple and straightforward logic that's easy to understand and debug

Key Improvements

- **Dynamic array sizing**: Uses variable-length array `a[n]` instead of fixed `a[100]` for better memory efficiency and safety
- **Input validation**: Should consider adding constraints to prevent buffer overflow
- **Edge case handling**: Works correctly with negative numbers (maximum is still properly identified)

Time Complexity

- **O(n)** per test case for finding maximum and indices
- **O(m × n)** total for all test cases

Space Complexity

- **O(n)** for the array storage per test case