Binary Search Algorithms Day 1

Only one of these is entirely correct. Find it!

```
// CS 10
// Dr. White
// Aug 20, 2026

public static int versionA(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    while (low < high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

public static int versionB(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid;
        } else {
            high = mid;
        }
    }
    return -1;
}

public static int versionC(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
```

```
    }
    return -1;
}

public static int versionD(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    int mid = (low + high) / 2;
    while (low <= high) {
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

public static int versionE(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    while (low <= high) {
        int mid = low;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}

public static int versionF(int[] arr, int target) {
    int low = 0;
    int high = arr.length - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}
```