

1. WAP to search an element 'm' in an array of size 'n' using linear search.
 2. WAP to search an element 'm' in an array of size 'n' using binary search. Assume that the array is sorted.
 3. WAP to find a peak element which is not smaller than its neighbours. Eg: Input: array[] = {5, 10, 20, 15} Output: 20 Explanation: The element 20 has neighbors 10 and 15, both of them are less than 20.
 4. WAP to find the minimum element of an array.
 5. WAP to reverse an array. 6. WAP to find the K'th smallest element in unsorted array. K=user input.
 7. WAP to find the Union and Intersection of two sorted arrays. Both the arrays are entered by the user.
 8. Print a given matrix in spiral form. Input: matrix = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16 }} Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
 9. Given an array of n elements that contains elements from 0 to n-1, with any of these numbers appearing any number of times. WAP to find these repeating numbers.
 10. Given an array of integers arr[], The task is to find the index of first repeating element in it i.e. the element that occurs more than once and whose index of the first occurrence is the smallest. WAP for the same.
-

1. Linear Search

Pseudocode:

1. Take input array size **n** and elements.
2. Take input element **m** to search.
3. Loop through the array and check if any element equals **m**.
4. If found, print the index; otherwise, print not found.

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int n, m;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
```

```
        scanf("%d", &arr[i]);
    }

    printf("Enter the element to search: ");
    scanf("%d", &m);

    int found = -1;
    for (int i = 0; i < n; i++) {
        if (arr[i] == m) {
            found = i;
            break;
        }
    }

    if (found != -1) {
        printf("Element found at index: %d\n", found);
    } else {
        printf("Element not found\n");
    }

    return 0;
}
```

2. Binary Search (Array Must Be Sorted)

Pseudocode:

1. Take sorted array and input `m` to search.
2. Set `low` to 0 and `high` to `n-1`.
3. Use binary search: calculate `mid`, compare `arr[mid]` with `m`.
4. Adjust `low` or `high` based on comparison until `m` is found or range is exhausted.

Code:

c
Copy code
`#include <stdio.h>`

```

int main() {
    int n, m;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d sorted elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the element to search: ");
    scanf("%d", &m);

    int low = 0, high = n - 1, found = -1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == m) {
            found = mid;
            break;
        } else if (arr[mid] < m) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    if (found != -1) {
        printf("Element found at index: %d\n", found);
    } else {
        printf("Element not found\n");
    }

    return 0;
}

```

3. Find a Peak Element

Pseudocode:

1. Take array input.
2. Loop through each element, checking if it's greater than its neighbors.
3. If a peak is found, print it and exit.

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    for (int i = 0; i < n; i++) {
        if ((i == 0 || arr[i] >= arr[i - 1]) && (i == n - 1 || arr[i]
>= arr[i + 1])) {
            printf("Peak element is: %d\n", arr[i]);
            break;
        }
    }

    return 0;
}
```

4. Find the Minimum Element in an Array**Pseudocode:**

1. Take array input.

2. Initialize `min` to the first element.
3. Loop through the array, updating `min` whenever a smaller element is found.
4. Print the minimum element.

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int min = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] < min) {
            min = arr[i];
        }
    }

    printf("Minimum element is: %d\n", min);
    return 0;
}
```

5. Reverse an Array

Pseudocode:

1. Take array input.
2. Swap elements from start and end towards the center.
3. Print the reversed array.

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Reverse the array
    for (int i = 0; i < n / 2; i++) {
        int temp = arr[i];
        arr[i] = arr[n - i - 1];
        arr[n - i - 1] = temp;
    }

    printf("Reversed array: ");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

6. Find the K'th Smallest Element in Unsorted Array**Pseudocode:**

1. Take array and **k** input.

2. Sort the array.
3. Print the element at index $k-1$.

Code:

c

Copy code

```
#include <stdio.h>

void sort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] > arr[j]) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

int main() {
    int n, k;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter the value of k: ");
    scanf("%d", &k);

    sort(arr, n);

    if (k > 0 && k <= n) {
        printf("The %d'th smallest element is: %d\n", k, arr[k - 1]);
    } else {
```

```
        printf("Invalid value of k\n");
    }

    return 0;
}
```

7. Union and Intersection of Two Sorted Arrays

Pseudocode:

1. Take input of two sorted arrays.
2. For Union: Merge arrays and eliminate duplicates.
3. For Intersection: Print common elements.

Code:

c

Copy code

```
#include <stdio.h>

void printUnion(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    printf("Union: ");
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) {
            printf("%d ", arr1[i++]);
        } else if (arr2[j] < arr1[i]) {
            printf("%d ", arr2[j++]);
        } else {
            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
    while (i < n1) printf("%d ", arr1[i++]);
    while (j < n2) printf("%d ", arr2[j++]);
    printf("\n");
}
```



```

void printIntersection(int arr1[], int n1, int arr2[], int n2) {
    int i = 0, j = 0;
    printf("Intersection: ");
    while (i < n1 && j < n2) {
        if (arr1[i] < arr2[j]) i++;
        else if (arr2[j] < arr1[i]) j++;
        else {
            printf("%d ", arr1[i]);
            i++;
            j++;
        }
    }
    printf("\n");
}

int main() {
    int n1, n2;
    printf("Enter the size of the first array: ");
    scanf("%d", &n1);
    int arr1[n1];
    printf("Enter %d sorted elements for the first array: ", n1);
    for (int i = 0; i < n1; i++) {
        scanf("%d", &arr1[i]);
    }

    printf("Enter the size of the second array: ");
    scanf("%d", &n2);
    int arr2[n2];
    printf("Enter %d sorted elements for the second array: ", n2);
    for (int i = 0; i < n2; i++) {
        scanf("%d", &arr2[i]);
    }

    printUnion(arr1, n1, arr2, n2);
    printIntersection(arr1, n1, arr2, n2);

    return 0;
}

```

I can continue with the rest if you'd like!

40

You said:

continue

ChatGPT said:

ChatGPT

Here are the solutions and pseudocode for the remaining questions:

8. Print a Matrix in Spiral Form

Pseudocode:

1. Take the matrix input.
2. Initialize four boundaries: **top**, **bottom**, **left**, **right**.
3. Use a loop to print elements in the order of top row, right column, bottom row, and left column, adjusting boundaries accordingly.
4. Repeat until the entire matrix is printed in spiral order.

Code:

c

Copy code

```
#include <stdio.h>
```

```
int main() {
    int m, n;
    printf("Enter the number of rows and columns: ");
    scanf("%d %d", &m, &n);

    int matrix[m][n];
    printf("Enter the matrix elements:\n");
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}
```

```

int top = 0, bottom = m - 1, left = 0, right = n - 1;
printf("Spiral order: ");
while (top <= bottom && left <= right) {
    for (int i = left; i <= right; i++) {
        printf("%d ", matrix[top][i]);
    }
    top++;

    for (int i = top; i <= bottom; i++) {
        printf("%d ", matrix[i][right]);
    }
    right--;

    if (top <= bottom) {
        for (int i = right; i >= left; i--) {
            printf("%d ", matrix[bottom][i]);
        }
        bottom--;
    }

    if (left <= right) {
        for (int i = bottom; i >= top; i--) {
            printf("%d ", matrix[i][left]);
        }
        left++;
    }
}
printf("\n");

return 0;
}

```

9. Find Repeating Numbers in an Array

Pseudocode:

1. Take input array of size n .

2. Initialize an array `count` to track occurrences.
3. Loop through the input array, incrementing `count[arr[i]]`.
4. Print elements with count greater than 1 as repeating numbers.

Code:

c

Copy code

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements (0 to %d): ", n, n - 1);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int count[n];
    for (int i = 0; i < n; i++) {
        count[i] = 0;
    }

    for (int i = 0; i < n; i++) {
        count[arr[i]]++;
    }

    printf("Repeating numbers: ");
    for (int i = 0; i < n; i++) {
        if (count[i] > 1) {
            printf("%d ", i);
        }
    }
    printf("\n");

    return 0;
}
```

10. Find Index of the First Repeating Element

Pseudocode:

1. Take input array of size `n`.
2. Use an array `count` to track occurrences and another `index` array to store the first occurrence index.
3. Traverse the array; if an element appears again, update minimum index.
4. Print the smallest index of the first repeating element.

Code:

C

Copy code

```
#include <stdio.h>

int main() {
    int n;
    printf("Enter the size of the array: ");
    scanf("%d", &n);

    int arr[n];
    printf("Enter %d elements: ", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int index[1000];
    for (int i = 0; i < 1000; i++) {
        index[i] = -1;
    }

    int min_index = n;
    for (int i = 0; i < n; i++) {
        if (index[arr[i]] != -1) {
            if (index[arr[i]] < min_index) {
                min_index = index[arr[i]];
            }
        }
    }
}
```

```
        } else {
            index[arr[i]] = i;
        }
    }

    if (min_index == n) {
        printf("No repeating element found\n");
    } else {
        printf("Index of first repeating element: %d\n", min_index);
    }

    return 0;
}
```