PSG COLLEGE OF TECHNOLOGY, COIMBATORE -641004

DEPARTMENT OF COMPUTER APPLICATIONS

I SEMESTER MCA

23MX16 C PROGRAMMING LABORATORY

PROBLEM SHEET 3 – ARRAYS – SIMPLE PROBLEMS

MITHULESH N [25MX338]

1) Write a program to accept the integer values and display the second largest value in an array.

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

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

   if (n < 2) {
      printf("Need at least two elements to find the second largest.\n");
      return 1;
   }

   int largest = INT_MIN;
   int secondLargest = INT_MIN;
   int val;

   printf("Enter %d integers separated by spaces:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &val);

      if (val > largest) {
```

```
            secondLargest = largest;
            largest = val;
        }
        else if (val > secondLargest && val != largest) {
            secondLargest = val;
        }
    }

    if (secondLargest == INT_MIN) {
        printf("No distinct second largest element found.\n");
    }
    else {
        printf("The second largest value is: %d\n", secondLargest);
    }

    return 0;
}
```

2) Write a program to sort the list of numbers in an ascending and descending order.

```
#include <stdio.h>
#define SIZE 100

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

```
    }
}

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

int main() {
    int n, i;
    int arr[SIZE];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    if(n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

    printf("Enter %d numbers:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
    }

    bubbleSortAscending(arr, n);
    printf("Sorted in ascending order:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    bubbleSortDescending(arr, n);
    printf("Sorted in descending order:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

3) Write a program to search for a specified number in an array and display with its position.

```
#include <stdio.h>
#define SIZE 100

int main() {
    int n, i, target;
    int arr[SIZE];
    int found = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);
```

```
    if(n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

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

    printf("Enter the number to search: ");
    scanf("%d", &target);

    for(i = 0; i < n; i++) {
        if(arr[i] == target) {
            printf("Number %d found at position %d\n", target, i + 1);
            found = 1;
        }
    }

    if(!found) {
        printf("Number %d not found in the array.\n", target);
    }

    return 0;
}
```

4) Write a program to find the occurrence of positive, negative, even and odd elements for a given array.

```
#include <stdio.h>
#define SIZE 100

int main() {
   int n, i;
   int arr[SIZE];
   int positiveCount = 0, negativeCount = 0;
   int evenCount = 0, oddCount = 0;

   printf("Enter number of elements: ");
   scanf("%d", &n);

   if(n <= 0 || n > SIZE) {
      printf("Invalid number of elements.\n");
      return 1;
   }

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

   for(i = 0; i < n; i++) {
      if(arr[i] > 0)
         positiveCount++;
      else if(arr[i] < 0)
         negativeCount++;

      if(arr[i] % 2 == 0)
```

```
            evenCount++;
        else
            oddCount++;
    }

    printf("Positive elements: %d\n", positiveCount);
    printf("Negative elements: %d\n", negativeCount);
    printf("Even elements: %d\n", evenCount);
    printf("Odd elements: %d\n", oddCount);

    return 0;
}
```

5) Write a C program to check if array contains a duplicate number.

```
#include <stdio.h>
#define SIZE 100

int main() {
    int n, i, j;
    int arr[SIZE];
    int hasDuplicate = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    if(n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

    printf("Enter %d integers:\n", n);
    for(i = 0; i < n; i++) {
```

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

    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(arr[i] == arr[j]) {
                hasDuplicate = 1;
                break;
            }
        }
        if(hasDuplicate)
            break;
    }

    if(hasDuplicate)
        printf("Array contains duplicate numbers.\n");
    else
        printf("Array does not contain any duplicates.\n");

    return 0;
}
```

6) Write a C program to reverse array in place in C.

```
#include <stdio.h>
#define SIZE 100

int main() {
   int n, i, temp;
   int arr[SIZE];

   printf("Enter number of elements: ");
   scanf("%d", &n);

   if(n <= 0 || n > SIZE) {
      printf("Invalid number of elements.\n");
      return 1;
   }

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

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

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

```
    }
    printf("\n");

    return 0;
}
```

7) Given an array of size n and a number k, find all elements that appear more than n/k times.

```
#include <stdio.h>

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

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

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

    int threshold = n / k;

    printf("Elements appearing more than n/k (%d) times are:\n", threshold);

    for(int i = 0; i < n; i++) {
        if(arr[i] != -1) { // Check to avoid recounting visited elements
            int count = 1;
            for(int j = i + 1; j < n; j++) {
```

```
            if(arr[i] == arr[j]) {
                count++;
                arr[j] = -1; // Mark as visited
            }
        }
        if(count > threshold) {
            printf("%d\n", arr[i]);
        }
    }
  }

  return 0;
}
```

8) Write a program that accepts an array and a key value. Rotate the array element by 'key' times.

Example:

Input: array[]= [1, 2, 3, 4, 5, 6]

key=2

Output : [3, 4, 5, 6, 1, 2]

```
#include <stdio.h>
#define SIZE 100

void rotateLeft(int arr[], int n, int key) {
  key = key % n;
  int temp[key];

  for (int i = 0; i < key; i++) {
     temp[i] = arr[i];
  }

  for (int i = key; i < n; i++) {
```

```
        arr[i - key] = arr[i];
    }

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

int main() {
    int arr[SIZE];
    int n, key;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    if (n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

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

    printf("Enter rotation key: ");
    scanf("%d", &key);

    rotateLeft(arr, n, key);

    printf("Array after %d left rotations:\n", key);
```

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

    return 0;
}
```

9) Write a C program to merge sorted array?

```
#include <stdio.h>
#define SIZE 100

void mergeSortedArrays(int arr1[], int n1, int arr2[], int n2, int merged[]) {
    int i = 0, j = 0, k = 0;

    // Merge elements from both arrays in sorted order
    while (i < n1 && j < n2) {
        if (arr1[i] <= arr2[j]) {
            merged[k++] = arr1[i++];
        } else {
            merged[k++] = arr2[j++];
        }
    }

    // Copy remaining elements of arr1 if any
    while (i < n1) {
        merged[k++] = arr1[i++];
    }

    // Copy remaining elements of arr2 if any
    while (j < n2) {
        merged[k++] = arr2[j++];
```

```
    }
}

int main() {
    int arr1[SIZE], arr2[SIZE], merged[2 * SIZE];
    int n1, n2;

    printf("Enter number of elements in first sorted array: ");
    scanf("%d", &n1);

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

    printf("Enter number of elements in second sorted array: ");
    scanf("%d", &n2);

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

    mergeSortedArrays(arr1, n1, arr2, n2, merged);

    printf("Merged sorted array:\n");
    for (int i = 0; i < n1 + n2; i++) {
        printf("%d ", merged[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

10) Write a C program to check if array contains a duplicate number.

```
#include <stdio.h>
#define SIZE 100

int main() {
    int n, i, j;
    int arr[SIZE];
    int hasDuplicate = 0;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    if (n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

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

    for (i = 0; i < n - 1; i++) {
        for (j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                hasDuplicate = 1;
                break;
            }
        }
```

```
        if (hasDuplicate)
            break;
    }

    if (hasDuplicate)
        printf("Array contains duplicate numbers.\n");
    else
        printf("Array does not contain any duplicates.\n");

    return 0;
}
```

11) Write a C program to reverse array in place in C.

```
#include <stdio.h>
#define SIZE 100

int main() {
    int n, i, temp;
    int arr[SIZE];

    printf("Enter number of elements: ");
    scanf("%d", &n);

    if (n <= 0 || n > SIZE) {
        printf("Invalid number of elements.\n");
        return 1;
    }

    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
```

```
    }

    // Reverse in place by swapping elements
    for (i = 0; i < n / 2; i++) {
        temp = arr[i];
        arr[i] = arr[n - 1 - i];
        arr[n - 1 - i] = temp;
    }

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

    return 0;
}
```

12) Check if a given array contains duplicate elements within k distance from each other

Given an unsorted array that may contain duplicates. Also given a number k which is smaller than size of array. Write a C function that returns true if array contains duplicates within k distance.

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

bool containsDuplicatesWithinK(int arr[], int n, int k) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j <= i + k && j < n; j++) {
            if (arr[i] == arr[j]) {
                return true;
            }
        }
```

```
    }
    return false;
}

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

    if (n <= 0) {
        printf("Invalid array size.\n");
        return 1;
    }

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

    printf("Enter distance k: ");
    scanf("%d", &k);

    if (k <= 0 || k >= n) {
        printf("Invalid k value.\n");
        return 1;
    }

    if (containsDuplicatesWithinK(arr, n, k)) {
        printf("Array contains duplicates within distance %d.\n", k);
    } else {
```

```
        printf("No duplicates within distance %d found.\n", k);
    }

    return 0;
}
```