Submitted by:

NAME: MD. Sabbeer Chowdhury

REGISTRATION No.: 23201160

SECTION: C-2

COURSE CODE: CSE108

Submitted to:

Zaima Sartaj Taheri

Lecturer

University Of Asia Pacific

1. Given an array of size **N.** The task is to find the maximum and the minimum element of the array using the minimum number of comparisons

ANSWER:

```
#include<stdio.h>

int main()
{



int num [100],n,i;
printf("How many numbers: ");
scanf("%d",&n);

for ( i = 0 ; i<n; i++)
{
   scanf("%d", &num[i]);


}

int max=num[0];

for ( i = 1; i<n; i++)
{
   if(max < num[i]) max = num[i] ;

}

printf("Maximum = %d\n",max);

int min=num[0];

for ( i = 1; i<n; i++)
{
   if(min > num[i]) min = num[i] ;

}

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

2. Given an array **arr[]**, the task is to **reverse** the array. Reversing an array means **rearranging** the elements such that the **first** element becomes the **last**, the **second** element becomes **second last** and so on.

Answer:

```
#include<stdio.h>

int main()

{

int a[5],i;

printf("Enter array Elements: ");
 for (i = 0; i <= 4; i++)

{

scanf("%d",&a[i]);

}

printf("\nReverse array Elements: ");
for( i = 4 ;i >= 0; i--)

{

printf("%d",a[i]);

}

return 0;
}
```

3. Given an array, the task is to cyclically rotate the array clockwise by one time.

ANSWER:

```
#include <stdio.h>

int main() {
   int n, i;


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

   int a[n];


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


   int last = a[n - 1];


   for(i = n - 1; i > 0; i--) {
      a[i] = a[i - 1];
   }


   a[0] = last;


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


}
```

4. Sorting an array means arranging the elements of the array in a certain order. Generally sorting in an array is done to arrange the elements in increasing or decreasing order.
Problem statement: Given an array of integers **arr**, the task is to sort the array in ascending order and return it, **without using any built-in** functions.
ANSWER:

```
#include<stdio.h>

int main() {



int a[5],i,j,temp;

printf("Enter array Elements: "); for( i = 0 ;i<5;i++)

{

scanf("%d",&a[i]);

}

for( i = 0 ;i<5;i++)

{

    for (j = i + 1 ;j<5;j++)

      {

        if(a[i] > a[j])

         {

            temp=a[i];

            a[i]=a[j];

            a[j]=temp;

         }

      }
}

    printf("Array Elements: ");

    for( i = 0 ;i<5;i++)

   {

     printf("%d",a[i]);

   }

return 0;

}
```

5.Given an array of n integers. The task is to print the duplicates in the given array. If there are no duplicates then print -1.

Answer:

```
#include <stdio.h>

int main() {
    int n, i, j;


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

    int a[n];


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


    printf("Duplicate elements in the array are: ");
    for(i = 0; i < n - 1; i++) {
      for(j = i + 1; j < n; j++) {
         if(a[i] = a[j]) {
            printf("%d ", a[i]);
            break;
         }
      }
   }
   return 0;
}
```

6. Given a sorted array **arr[]** of size N and a number **X**, you need to find the number of occurrences of **X** in given array.

Answer:

```
#include <stdio.h>

int main() {

   int a[] = {1, 1, 2, 2, 2, 3, 4, 4, 5};
   int N = sizeof(a) / sizeof(a[0]);
   int X = 2;
   int Y = 4;
   int count = 0;
   for (int i = 0; i < N; i++) {
      if (a[i] == X) {
         count++;
      }

      else if (a[i] > X) {

      }
   }



   for (int i = 0; i < N; i++) {
      if (a[i] == Y) {
         count++;
      }

      else if (a[i] > Y) {
         break;
      }
   }
```

```
    printf("Element %d %d occurs %d %d times\n ",X, Y, count);

    return 0;
}
```

7. sort the array of 0s,1s and 2s.
Answer:

```
#include <stdio.h>
void sortArray(int a[],int n) {
    int low = 0, mid = 0, high = n - 1;
while (mid <= high)
{ if (a[mid] == 0) {
  int temp = a[low];
      a[low] = a[mid];
      a[mid] = temp; low++; mid++;
}
else if (a[mid] == 1) { mid++;
}
 else
    { int temp = a[mid];
          a[mid] = a[high];
          a[high] = temp; high--; }
 }
 }
int main()
{ int a[] = {0, 1, 2, 1, 0, 2, 1, 0};
int n = sizeof(a) / sizeof(a[0]);
sortArray(a, n);
printf("Sorted array: ");
for (int i = 0; i < n; i++)
    { printf("%d ", a[i]); }
    printf("\n");
return 0; }
```

8. An array contains both positive and negative numbers in random order. Rearrange the array elements so that all negative numbers appear before all positive numbers.

Answer:

```
#include <stdio.h>


void rearrangeArray(int a[], int n) {
   int left = 0, right = n - 1;

   while (left <= right) {

      if (a[left] < 0)
         left++;

      else if (a[right] >= 0)
         right--;

      else {
         int temp = a[left];
         a[left] = a[right];
         a[right] = temp;
         left++;
         right--;
      }
   }
}

int main() {
   int a[] = {-12, 11, -13, -5, 6, -7, 5, -3, -6};
   int n = sizeof(a) / sizeof(a[0]);

   rearrangeArray(a, n);


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

   return 0;}
```

9. Given a **binary** 2D array, where each row is **sorted**. Find the row with the maximum number of 1s.
Answer:

```
#include <stdio.h>


int rowWithMax1s(int a[][5], int n, int m) {
   int max_row_index = -1;
   int j = m - 1;
   for (int i = 0; i < n; i++) {
      while (j >= 0 && a[i][j] == 1) {
         j--;
         max_row_index = i;
      }
   }

   return max_row_index;
}

int main() {
   int a[][5] = {
      {0, 0, 0, 1, 1},
      {0, 1, 1, 1, 1},
      {0, 0, 0, 0, 1},
      {0, 0, 1, 1, 1},
   };

   int n = sizeof(a) / sizeof(a[0]);
   int m = sizeof(a[0]) / sizeof(a[0][0]);
   int result = rowWithMax1s(a, n, m);
   if (result != -1)
      printf("The row with the maximum number of 1s is: %d\n", result);
   else
      printf("No 1s found in the array.\n");

   return 0;
}
```

10. Given an array **arr**. Find the majority element in the array. If no majority exists, return -1. A majority element in an array is an element that appears **strictly** more than **arr.size() / 2 times** in the array.
Answer:

```
#include <stdio.h>

int findCandidate(int a[], int n) {
   int candidate = 0, count = 1;
   for (int i = 1; i < n; i++) {
      if (a[i] == a[candidate])
         count++;
      else
         count--;

      if (count == 0) {
         candidate = i;
         count = 1;
      }
   }
   return a[candidate];
}
int isMajority(int a[], int n, int candidate) {
   int count = 0;
   for (int i = 0; i < n; i++) {
      if (a[i] == candidate)
         count++;
   }
   if (count > n / 2)
      return candidate;
   else
      return -1;
}
int majorityElement(int a[], int n) {
   int candidate = findCandidate(a, n);
   return isMajority(a, n, candidate);
}

int main() {
   int a[] = {2, 2, 1, 1, 2, 2, 2};
   int n = sizeof(a) / sizeof(a[0]);
   int result = majorityElement(a, n);
```

```
    if (result != -1)
        printf("The majority element is: %d\n", result);
    else
        printf("No majority element exists.\n");

    return 0;
}
```

11. Given an unsorted array of integers, sort the array into a wave array. An array **arr[0..n-1]** is sorted in wave form if: **arr[0]** >= **arr[1]** <= **arr[2]** >= **arr[3]** <= **arr[4]** >= …..
Answer:

```
#include <stdio.h>

void bubbleSort(int a[], int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j + 1]) {

                int temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
}



void sortInWave(int a[], int n) {
    bubbleSort(a, n);
    for (int i = 0; i < n - 1; i += 2) {
        int temp = a[i];
        a[i] = a[i + 1];
        a[i + 1] = temp;
    }
}

int main() {
     int a[] = {10, 5, 6, 3, 2, 20, 100, 80};
    int n = sizeof(a) / sizeof(a[0]);
    sortInWave(a, n);
    printf("Array in wave form: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");

    return 0;
}
```