Array (1)

1. Write a programme to Take as input 5 numbers from the user and store them in an array.

Input :

```
#include <stdio.h>


int main() {
   // Declare an array to store 5 numbers
   int numbers[5];

   // Take input for 5 numbers from the user
   printf("Enter 5 numbers:\n");

   for (int i = 0; i < 5; ++i) {
      printf("Enter number %d: ", i + 1);
      scanf("%d", &numbers[i]);
   }

   // Display the numbers entered by the user
   printf("\nNumbers entered by the user:\n");

   for (int i = 0; i < 5; ++i) {
```

```
        printf("%d ", numbers[i]);
    }

    return 0;
}
```

Output:

```
"C:\Users\ASUS\Documents\aaray p1.exe"
Enter 5 numbers:
Enter number 1: 5
Enter number 2: 8
Enter number 3: 9
Enter number 4: 4
Enter number 5: 8

Numbers entered by the user:
5 8 9 4 8
Process returned 0 (0x0)   execution time : 8.259 s
Press any key to continue.
```

2. Inserting an element into a position of an array. The element and the insertion point are inputs from the user.

Input:

```
#include <stdio.h>

// Function to insert an element into a position of an array
void insertElement(int arr[], int size, int element, int position) {
    // Check if the position is valid
    if (position < 0 || position > size) {
```

```
        printf("Invalid position! Please enter a valid position.\n");
        return;
    }

    // Shift elements to create space for the new element
    for (int i = size - 1; i >= position; i--) {
        arr[i + 1] = arr[i];
    }

    // Insert the element at the specified position
    arr[position] = element;

    // Increment the size of the array
    size++;

    // Display the updated array
    printf("Array after insertion:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
```

```
int main() {
    int size, element, position;

    // Get the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Declare an array of the specified size
    int arr[size];

    // Get the elements of the array from the user
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Get the element and position from the user
    printf("Enter the element to insert: ");
    scanf("%d", &element);

    printf("Enter the position to insert: ");
```

```
    scanf("%d", &position);

    // Call the insertElement function to insert the element at the
specified position
    insertElement(arr, size, element, position);

    return 0;
}
```

Output:

```
"C:\Users\ASUS\Documents\aaray p2.exe"
Enter the size of the array: 10
Enter the elements of the array:
7 8 9 5 7 8 9 7 4 5 6
Enter the element to insert: Enter the position to insert: 10
Array after insertion:
7 8 9 5 7 8 9 7 4 5 6

Process returned 0 (0x0)   execution time : 28.478 s
Press any key to continue.
```

3. Deleting an element from an array.

Input:

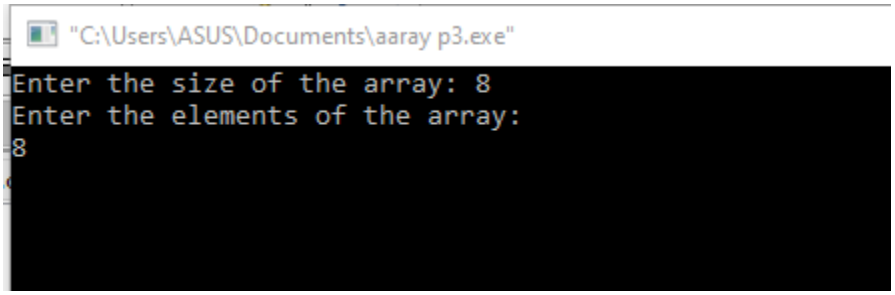
```
#include <stdio.h>

// Function to delete an element from an array
```

```
void deleteElement(int arr[], int *size, int element) {
    int found = 0;

    // Search for the element in the array
    for (int i = 0; i < *size; i++) {
        if (arr[i] == element) {
            found = 1;

            // Shift elements to fill the gap created by deleting the element
            for (int j = i; j < *size - 1; j++) {
                arr[j] = arr[j + 1];
            }

            // Decrement the size of the array
            (*size)--;

            // Break out of the loop since we found and deleted the element
            break;
        }
    }

    if (found) {
```

```
        // Display the updated array
        printf("Element %d deleted from the array.\n", element);
        printf("Array after deletion:\n");
        for (int i = 0; i < *size; i++) {
            printf("%d ", arr[i]);
        }
        printf("\n");
    } else {
        printf("Element %d not found in the array.\n", element);
    }
}

int main() {
    int size, element;

    // Get the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check for a valid array size
    if (size <= 0) {
        printf("Invalid array size. Exiting the program.\n");
```

```
        return 1;
    }

    // Declare an array of the specified size
    int arr[size];

    // Get the elements of the array from the user
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Get the element to delete from the user
    printf("Enter the element to delete: ");
    scanf("%d", &element);

    // Call the deleteElement function to delete the specified element
from the array
    deleteElement(arr, &size, element);

    return 0;
}
```

Output:



4. Write a programme to search for an element from an array input from the user.

Input:

```
#include <stdio.h>


// Function to search for an element in an array
int searchElement(int arr[], int size, int element) {
    // Search for the element in the array
    for (int i = 0; i < size; i++) {
        if (arr[i] == element) {
            // Element found, return its index
            return i;
        }
    }
```

```
    // Element not found, return -1
    return -1;
}

int main() {
    int size, element;

    // Get the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check for a valid array size
    if (size <= 0) {
        printf("Invalid array size. Exiting the program.\n");
        return 1;
    }

    // Declare an array of the specified size
    int arr[size];

    // Get the elements of the array from the user
    printf("Enter the elements of the array:\n");
```

```
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Get the element to search from the user
    printf("Enter the element to search: ");
    scanf("%d", &element);

    // Call the searchElement function to search for the specified
element in the array
    int index = searchElement(arr, size, element);

    // Display the result
    if (index != -1) {
        printf("Element %d found at index %d.\n", element, index);
    } else {
        printf("Element %d not found in the array.\n", element);
    }

    return 0;
}
```

5. Write a programme to find out the maximum , minimum and mode of an array of numbers.

Input:

```
#include <stdio.h>


// Function to find the maximum value in an array
int findMax(int arr[], int size) {
    int max = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}


// Function to find the minimum value in an array
int findMin(int arr[], int size) {
    int min = arr[0];
    for (int i = 1; i < size; i++) {
        if (arr[i] < min) {
            min = arr[i];
```

```
        }
    }
    return min;
}

// Function to find the mode in an array
int findMode(int arr[], int size) {
    int maxCount = 0, mode = -1;

    for (int i = 0; i < size; i++) {
        int count = 1;

        for (int j = i + 1; j < size; j++) {
            if (arr[i] == arr[j]) {
                count++;
            }
        }

        if (count > maxCount) {
            maxCount = count;
            mode = arr[i];
        }
```

```
    }

    return mode;
}

int main() {
    int size;

    // Get the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check for a valid array size
    if (size <= 0) {
        printf("Invalid array size. Exiting the program.\n");
        return 1;
    }

    // Declare an array of the specified size
    int arr[size];

    // Get the elements of the array from the user
```

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

    // Call the findMax, findMin, and findMode functions
    int max = findMax(arr, size);
    int min = findMin(arr, size);
    int mode = findMode(arr, size);

    // Display the results
    printf("Maximum: %d\n", max);
    printf("Minimum: %d\n", min);
    printf("Mode: %d\n", mode);

    return 0;
}
```

6. Write a Programme to Delete Duplicate elements from an array.

Input:

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

```
// Function to remove duplicate elements from an array
int removeDuplicates(int arr[], int size) {
    if (size <= 1) {
        return size; // No duplicates to remove
    }

    int uniqueIndex = 1; // Index to track the position of unique elements

    // Iterate through the array to find and remove duplicates
    for (int i = 1; i < size; i++) {
        int isDuplicate = 0;

        // Check if the current element is a duplicate
        for (int j = 0; j < uniqueIndex; j++) {
            if (arr[i] == arr[j]) {
                isDuplicate = 1;
                break;
            }
        }

        // If the element is not a duplicate, add it to the unique elements
        if (!isDuplicate) {
```

```
            arr[uniqueIndex] = arr[i];
            uniqueIndex++;
        }
    }

    return uniqueIndex; // Return the new size of the array without duplicates
}

int main() {
    int size;

    // Get the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check for a valid array size
    if (size <= 0) {
        printf("Invalid array size. Exiting the program.\n");
        return 1;
    }
```

```
    // Declare an array of the specified size
    int arr[size];

    // Get the elements of the array from the user
    printf("Enter the elements of the array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Call the removeDuplicates function to remove duplicate elements
    int newSize = removeDuplicates(arr, size);

    // Display the array without duplicates
    printf("Array without duplicates:\n");
    for (int i = 0; i < newSize; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

7. Take n number input from the user. Find out their GCD and LCM.

Input:

```
#include<stdio.h>
int main()
{
  int num1,num2,n1,n2,rem,gcd,lcm;
  printf("Enter 2 number : ");
  scanf("%d %d",&num1,&num2);

    n1=num1;
    n2=num2;

  while (n2!=0)
  {

    rem=n1%n2;
    n1=n2;
    n2=rem;

  }

   gcd=n1;
```

```
  lcm=(num1*num2)/gcd;


  printf("GCD = %d\n",gcd);
  printf("LCM = %d\n",lcm);


}
```

Output:

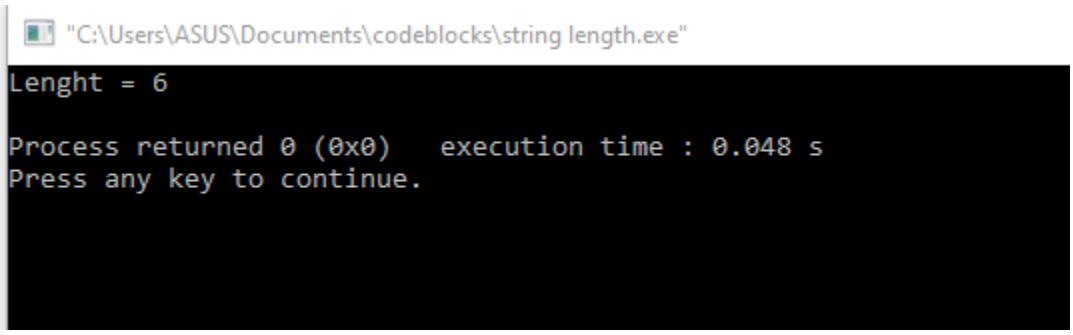
8 Write a programme to find the length of the string.

Input:

```
#include<stdio.h>
int main()
{
  char s1[]= "Sajeed";
```

```
  int len = strlen (s1);

   printf("Lenght = %d\n",len);



}
```

Output:



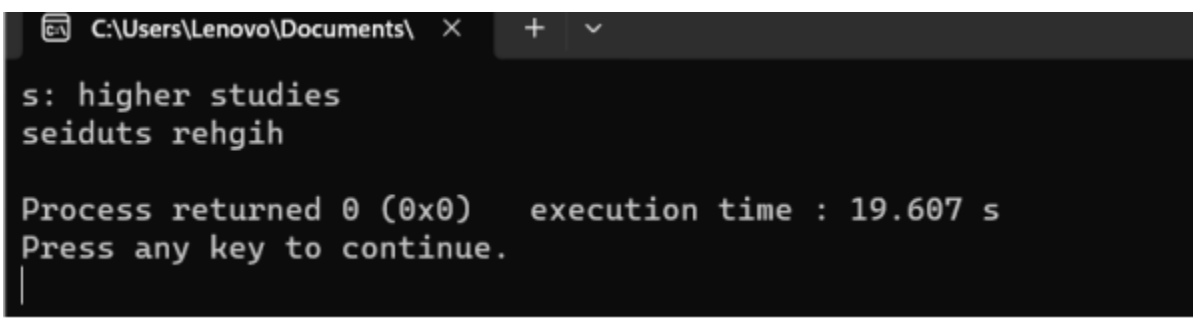
9. Write a Progarmme to Reverse a String.

Input:

```
#include <stdio.h>

int main()

{

char s[100], r[100];


int n,end,i=0;
```

```
printf("s: ");
gets(s);
while (s[i]!='\0')
i++;
end=i-1;
for(n=0;n<i;n++)
{
r[n]=s[end];
end--;
}
r[n]='\0';
printf("%s\n",r);
return 0;
}
```

Output:



10.Write a Programme to count the numbers of word and characters present in the text.

Input:

```
#include <stdio.h>

#include <string.h>


#define MAX_SIZE 1000


// Function to count words and characters in a text

void countWordsAndCharacters(char text[]) {

    int words = 0, characters = 0;


    // Iterate through each character in the text

    for (int i = 0; text[i] != '\0'; i++) {

        // Increment character count for each non-space character

        if (text[i] != ' ' && text[i] != '\t' && text[i] != '\n') {

            characters++;

        }


        // Check for the end of a word (space, tab, or newline)

        if (text[i] == ' ' || text[i] == '\t' || text[i] == '\n') {

            words++;
```

```
        }
    }

    // Increment word count for the last word (if any)
    if (characters > 0) {
        words++;
    }

    // Display the results
    printf("Number of words: %d\n", words);
    printf("Number of characters: %d\n", characters);
}

int main() {
    char text[MAX_SIZE];

    // Get the text from the user
    printf("Enter the text (max %d characters):\n", MAX_SIZE - 1);
    fgets(text, MAX_SIZE, stdin);

    // Call the countWordsAndCharacters function to count words and
characters
```

```
    countWordsAndCharacters(text);


    return 0;
}
```

Output:

```
"C:\Users\ASUS\Documents\array 10.exe"
Enter the text (max 999 characters):
100
Number of words: 2
Number of characters: 3

Process returned 0 (0x0)   execution time : 5.456 s
Press any key to continue.
```

11. Take a String as input as check whether it is a palindrome. If it is not a palindrome ,then add a minimum no of characters after the string to convert into a palindrome.

Input : #include <stdio.h>

```
int main()
{
char s[100], r[100];
int n,end,i=0;
printf("s: ");
gets(s);
while (s[i]!='\0')
```

```
i++;
end=i-1;
for(n=0;n<i;n++)
{
r[n]=s[end];
end--;
}
r[n]='\0';
printf("%s\n",r);

int p=strcmp(s,r);
if (p==0)
{
printf("%s is a palindrome\n", s);
}
else
printf("%s is not a palindrome\n", r);
}
```

Output:

```
C:\Users\Lenovo\Documents\

s: level
level
level is a palindrome

Process returned 0 (0x0)   execution time : 4.982 s
Press any key to continue.
```

12. Write a Programme that will search for a substring within a string.

Input:

```
#include <stdio.h>

#include <string.h>


// Function to search for a substring within a string

int searchString(char mainString[], char subString[]) {

   int mainLen = strlen(mainString);

   int subLen = strlen(subString);


   // Iterate through the main string

   for (int i = 0; i <= mainLen - subLen; i++) {

      int j;
```

```
        // Check for a match starting from the current position in the main
string
        for (j = 0; j < subLen; j++) {
            if (mainString[i + j] != subString[j]) {
                break; // Mismatch, move to the next position in the main
string
            }
        }

        // If the inner loop completed without a break, a match is found
        if (j == subLen) {
            return i; // Return the starting index of the substring in the main
string
        }
    }

    return -1; // Return -1 if the substring is not found
}

int main() {
    char mainString[100], subString[50];

    // Get the main string from the user
```

```
printf("Enter the main string: ");
fgets(mainString, sizeof(mainString), stdin);

// Remove the newline character from the end of the main string
mainString[strcspn(mainString, "\n")] = '\0';

// Get the substring to search from the user
printf("Enter the substring to search: ");
fgets(subString, sizeof(subString), stdin);

// Remove the newline character from the end of the substring
subString[strcspn(subString, "\n")] = '\0';

// Call the searchString function to search for the substring
int index = searchString(mainString, subString);

// Display the result
if (index != -1) {
   printf("Substring found at index %d in the main string.\n", index);
} else {
   printf("Substring not found in the main string.\n");
}
```

```
    return 0;
}
```

Output: