Advanced C Lab assignment 2 Ex3

Name: Sankalp Mukim

Registration Number: 20BDS0128

Course Code: CSE2010

Slot: L27+L28

Table of Contents

[Part 1 4](#_Toc94820992)

[WAP to sort n alphabets (both upper and lower) in an array in ascending order. 4](#_Toc94820993)

[Code 4](#_Toc94820994)

[Output 5](#_Toc94820995)

[WAP to extract odd, even, and prime numbers in an array to 3 different arrays and display this array with proper formatting 5](#_Toc94820996)

[Code 5](#_Toc94820997)

[Output 6](#_Toc94820998)

[WAP to rearrange positive and negative integers in a 1-D array 6](#_Toc94820999)

[Code 6](#_Toc94821000)

[Output 8](#_Toc94821001)

[WAP to insert an element into the array, considering all the 3 cases i.e., Beginning of array, Middle of Array, End of array. Finally display the resultant array. 8](#_Toc94821002)

[Code 8](#_Toc94821003)

[Output 9](#_Toc94821004)

[WAP a program to swap adjacent elements in a one-dimensional array 9](#_Toc94821005)

[Code 9](#_Toc94821006)

[Output 10](#_Toc94821007)

[WAP to search an element in an array and replace with an element \* 10](#_Toc94821008)

[Code 10](#_Toc94821009)

[Output 11](#_Toc94821010)

[WAP to split the given 1-D array into repeating and non-repeating elements of the array 11](#_Toc94821011)

[Code 11](#_Toc94821012)

[Output 12](#_Toc94821013)

[WAP to find the sum of two matrix 13](#_Toc94821014)

[Code 13](#_Toc94821015)

[Output 14](#_Toc94821016)

[WAP to transpose the given matrix 14](#_Toc94821017)

[Code 14](#_Toc94821018)

[Output 15](#_Toc94821019)

[WAP to the find the product of two matrix 15](#_Toc94821020)

[Code 15](#_Toc94821021)

[Output 16](#_Toc94821022)

[WAP to find the product of boundary elements of a given matrix. 16](#_Toc94821023)

[Algorithm 16](#_Toc94821024)

[Code 17](#_Toc94821025)

[Output 18](#_Toc94821026)

[WAP to find the maximum element in the given row and minimum element in the given column for a given matrix 18](#_Toc94821027)

[Algorithm 18](#_Toc94821028)

[Code 18](#_Toc94821029)

[Output 19](#_Toc94821030)

[WAP to carry out Left to Right rotations of a 1-D array n times 19](#_Toc94821031)

[Algorithm 19](#_Toc94821032)

[Code 20](#_Toc94821033)

[Output 20](#_Toc94821034)

[WAP to find all the patterns of 0(2+)0 in the given string. Given a string containing 0’s and 2’s. Find the total number of 0(2+)0 patterns in the string and output it. 21](#_Toc94821035)

[Algorithm 21](#_Toc94821036)

[Code 21](#_Toc94821037)

[Output 22](#_Toc94821038)

[Part 2 22](#_Toc94821039)

[Write a C program to count maximum and minimum occurrence for a given character ch in the given string s1. 22](#_Toc94821040)

[Code 22](#_Toc94821041)

[Output 23](#_Toc94821042)

[WAP a program to reverse the given string and display the resultant string 23](#_Toc94821043)

[Code 23](#_Toc94821044)

[Output 23](#_Toc94821045)

[Write a C program to concatenate the following quotes of Dr. A. P. J “If you fail, never give up because FAIL means “First Attempt in Learning.” “All of us do not have equal talent. But, all of us have an equal opportunity to develop our talents.” “Excellence is a continuous process & not an accident” Finally display the output in the following format Dr. A. P. J quotes follows......................(with the resultant string). 24](#_Toc94821046)

[Code 24](#_Toc94821047)

[Output 24](#_Toc94821048)

[Considering the given quotes of Dr. A. P. J stated above, find the frequency of occurrence of all the special characters in the given quotes. 24](#_Toc94821049)

[Code 24](#_Toc94821050)

[Output 25](#_Toc94821051)

[Write a C program to count the number of words, number of sentences and finally the length of the string without using any built- in functions of strings. 25](#_Toc94821052)

[Code 25](#_Toc94821053)

[Output 26](#_Toc94821054)

[WAP to remove all characters from the given string except the alphabet character and display the final string 26](#_Toc94821055)

[Code 26](#_Toc94821056)

[Output 27](#_Toc94821057)

[Write a program in C to replace the spaces of a given string “ARISE, AWAKE, and STOP NOT UNTIL THE GOAL is ACHIEVED” with a special character of your choice. 27](#_Toc94821058)

[Code 27](#_Toc94821059)

[Output 27](#_Toc94821060)

[WAP to split the given sentence into words and display the words one after the other. 27](#_Toc94821061)

[Code 27](#_Toc94821062)

[Output 28](#_Toc94821063)

[Write a program to find the number of times a given word 'the” appears in the given string “Best brain of the nation may be found in the last benches of the classroom”. 28](#_Toc94821064)

[Algorithms 28](#_Toc94821065)

[Code 29](#_Toc94821066)

[Output 29](#_Toc94821067)

[WAP to remove the duplicate characters from the given string, S and print the resultant string, RS (Hint: S <-AabCaBcdEGzYz , RS <-AbCdEGzY) 29](#_Toc94821068)

[Code 29](#_Toc94821069)

[Output 30](#_Toc94821070)

[WAP to remove all characters from the given string except the alphabet character and display the final string (without using the second string) 30](#_Toc94821071)

[Code 30](#_Toc94821072)

[Output 31](#_Toc94821073)

[WAP to insert a substring anywhere in between the given string except in the first and last position of the string. Finally display the resultant string on inserting the sub-string. 31](#_Toc94821074)

[Code 31](#_Toc94821075)

[Output 32](#_Toc94821076)

[WAP to reverse words in the given strings and display the reversed string 32](#_Toc94821077)

[Code 32](#_Toc94821078)

[Output 33](#_Toc94821079)

# Part 1

## WAP to sort n alphabets (both upper and lower) in an array in ascending order.

### Code

#include **<**stdio.h**>**

***int*** main()

{

    printf(**"**Enter the number of characters:**"**);

***int*** n;

    scanf(**"**%d**"**, &n);

    printf(**"**Enter %d characters:\n**"**, n);

***char*** arr[50];

    scanf(**"**%c**"**, &arr[0]);

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%c**"**, &(arr[i]));

    }

    printf(**"**Before sorting:\n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%c, **"**, arr[i]);

    }

    printf(**"**\n]**"**);

***int*** min\_idx;

**//** selection sort

    for (***int*** i = 0; i < n - 1; i++)

    {

        min\_idx = i;

        for (***int*** j = i + 1; j < n; j++)

        {

            if (arr[j] < arr[min\_idx])

            {

                min\_idx = j;

            }

        }

**//** swap

***char*** temp = arr[min\_idx];

        arr[min\_idx] = arr[i];

        arr[i] = temp;

    }

    printf(**"**\nAfter sorting\n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%c, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    return 0;

}

### Output

Text

Description automatically generated

## WAP to extract odd, even, and prime numbers in an array to 3 different arrays and display this array with proper formatting

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** arr[100], odds[100], evens[100], primes[100];

***int*** n;

    printf(**"**Enter n: **"**);

    scanf(**"**%d**"**, &n);

    printf(**"**Enter arr: **"**);

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%d**"**, &arr[i]);

    }

    printf(**"**\n**"**);

***int*** odd\_count = 0, even\_count = 0, prime\_count = 0;

    for (***int*** i = 0; i < n; i++)

    {

        if (arr[i] % 2 == 0)

        {

            evens[even\_count++] = arr[i];

        }

        else

        {

            odds[odd\_count++] = arr[i];

        }

***int*** flag = 0;

        for (***int*** j = 2; j <= arr[i] / 2; j++)

        {

            if (arr[i] % j == 0)

            {

                flag = 1;

                break;

            }

        }

        if (flag == 0)

        {

            primes[prime\_count++] = arr[i];

        }

    }

    printf(**"**Odds: **"**);

    for (***int*** i = 0; i < odd\_count; i++)

    {

        printf(**"**%d, **"**, odds[i]);

    }

    printf(**"**\nEvens: **"**);

    for (***int*** i = 0; i < even\_count; i++)

    {

        printf(**"**%d, **"**, evens[i]);

    }

    printf(**"**\nPrimes: **"**);

    for (***int*** i = 0; i < prime\_count; i++)

    {

        printf(**"**%d, **"**, primes[i]);

    }

    printf(**"**\n**"**);

    return 0;

}

### Output

Text

Description automatically generated

## WAP to rearrange positive and negative integers in a 1-D array

### Code

#include **<**stdio.h**>**

**//** Arranging negative then positive values

***int*** main()

{

***int*** n, arr[50];

    printf(**"**Enter n:**"**);

    scanf(**"**%d**"**, &n);

    printf(**"**Enter %d numbers:\n**"**, n);

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%d**"**, &arr[i]);

    }

    printf(**"**\n**"**);

**//** traverse the array

    for (***int*** i = 0; i < n; i++)

    {

**//** if the element is negative

        if (arr[i] > 0)

        {

**//** traverse the array again

            for (***int*** j = i + 1; j < n; j++)

            {

**//** if the element is positive

                if (arr[j] < 0)

                {

**//** swap

***int*** temp = arr[i];

                    arr[i] = arr[j];

                    arr[j] = temp;

                    break;

                }

            }

        }

    }

    printf(**"**\nAfter sorting:\n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    return 0;

}

### Output

A screenshot of a computer

Description automatically generated with medium confidence

## WAP to insert an element into the array, considering all the 3 cases i.e., Beginning of array, Middle of Array, End of array. Finally display the resultant array.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** arr[100] = {10, 20, 30};

***int*** SIZE = 3;

***int*** i, j;

    printf(**"**Array to start with:\n[\n\t**"**);

    for (i = 0; i < SIZE; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    printf(**"**Enter number to add to beginning of array: **"**);

***int*** num;

    scanf(**"**%d**"**, &num);

    for (i = SIZE; i > 0; i--)

    {

        arr[i] = arr[i - 1];

    }

    arr[0] = num;

    SIZE++;

    printf(**"**After adding %d to beginning of array:\n[\n\t**"**, num);

    for (i = 0; i < SIZE; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    printf(**"**Enter element index to add to array: **"**);

***int*** index, element;

    scanf(**"**%d**"**, &index);

    printf(**"**Enter element to add to array: **"**);

    scanf(**"**%d**"**, &element);

    for (***int*** i = SIZE; i > index; i--)

    {

        arr[i] = arr[i - 1];

    }

    arr[index] = element;

    SIZE++;

    printf(**"**After adding %d to middle of array:\n[\n\t**"**, element);

    for (i = 0; i < SIZE; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    printf(**"**Enter element to add to end of array: **"**);

    scanf(**"**%d**"**, &arr[SIZE++]);

    printf(**"**After adding element to end of array:\n[\n\t**"**);

    for (i = 0; i < SIZE; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

}

### Output

Text

Description automatically generated

## WAP a program to swap adjacent elements in a one-dimensional array

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** n, arr[100];

    printf(**"**Enter n:  **"**);

    scanf(**"**%d**"**, &n);

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%d**"**, &arr[i]);

    }

    for (***int*** i = 1; i < n; i += 2)

    {

***int*** temp = arr[i];

        arr[i] = arr[i - 1];

        arr[i - 1] = temp;

    }

    printf(**"**Array after swapping:\n**"**);

    printf(**"**[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

}

### Output

Graphical user interface, text

Description automatically generated

## WAP to search an element in an array and replace with an element \*

### Code

#include **<**stdio.h**>**

***int*** main()

{

    printf(**"**Enter the number of characters:**"**);

***int*** n;

    scanf(**"**%d**"**, &n);

    printf(**"**Enter %d characters:\n**"**, n);

***char*** arr[50];

    scanf(**"**%c**"**, &arr[0]);

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%c**"**, &(arr[i]));

    }

    printf(**"**Inputted array:\n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%c, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    printf(**"**Enter the element to search and replace: **"**);

***char*** ch;

    scanf(**"**%c**"**, &ch);

    scanf(**"**%c**"**, &ch);

    for (***int*** i = 0; i < n; i++)

    {

        if (arr[i] == ch)

        {

            arr[i] = **'**\***'**;

        }

    }

    printf(**"**Output array:\n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%c, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

    return 0;

}

### Output

Text

Description automatically generated

## WAP to split the given 1-D array into repeating and non-repeating elements of the array

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** n;

    printf(**"**Enter n: \n**"**);

    scanf(**"**%d**"**, &n);

    printf(**"**Enter array: \n**"**);

***int*** arr[50];

    for (***int*** i = 0; i < n; i++)

    {

        scanf(**"**%d**"**, &arr[i]);

    }

    printf(**"**\n**"**);

***int*** last\_non\_repeating\_element = -1;

    for (***int*** i = 0; i < n; i++)

    {

***int*** repeating = 0;

        for (***int*** j = 0; j < n; j++)

        {

            if (arr[i] == arr[j] && i != j)

            {

                repeating++;

                break;

            }

        }

        if (!repeating)

        {

***int*** non\_repeating = arr[i];

            for (***int*** k = i; k > last\_non\_repeating\_element; k--)

            {

                arr[k] = arr[k - 1];

            }

            arr[++last\_non\_repeating\_element] = non\_repeating;

        }

    }

    printf(**"**Non repeating to the left and repeating to the right\n**"**);

    printf(**"**Final array is: \n[\n\t**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

}

### Output

Text

Description automatically generated

## WAP to find the sum of two matrix

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** m, n;

    printf(**"**Enter no. of rows(m) and no. of columns(n).\n**"**);

    scanf(**"**%d**"**, &m);

    scanf(**"**%d**"**, &n);

***int*** mat1[20][20], mat2[20][20], mat3[20][20];

    printf(**"**Enter matrix 1 rowwise\n**"**);

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            scanf(**"**%d**"**, &mat1[i][j]);

        }

    }

    printf(**"**Enter matrix 2 rowwise\n**"**);

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            scanf(**"**%d**"**, &mat2[i][j]);

            mat3[i][j] = mat1[i][j] + mat2[i][j];

        }

    }

    printf(**"**Sum of matrix 1 & 2 is:\n**"**);

    printf(**"**[**"**);

    for (***int*** i = 0; i < m; i++)

    {

        printf(**"**\n\t**"**);

        for (***int*** j = 0; j < n; j++)

        {

            printf(**"**%d  **"**, mat3[i][j]);

        }

    }

    printf(**"**\n]\n**"**);

}

### Output

Text

Description automatically generated

## WAP to transpose the given matrix

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** m, n;

    printf(**"**Enter no. of rows(m) and no. of columns(n).\n**"**);

    scanf(**"**%d**"**, &m);

    scanf(**"**%d**"**, &n);

***int*** mat1[20][20], mat2[20][20];

    printf(**"**Enter matrix 1 rowwise\n**"**);

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            scanf(**"**%d**"**, &mat1[i][j]);

        }

    }

    for (***int*** i = 0; i < n; i++)

    {

        for (***int*** j = 0; j < m; j++)

        {

            mat2[i][j] = mat1[j][i];

        }

    }

    printf(**"**Transpose of given matrix:\n**"**);

    printf(**"**[**"**);

    for (***int*** i = 0; i < n; i++)

    {

        printf(**"**\n\t**"**);

        for (***int*** j = 0; j < m; j++)

        {

            printf(**"**%d  **"**, mat2[i][j]);

        }

    }

    printf(**"**\n]\n**"**);

}

### Output

Text

Description automatically generated

## WAP to the find the product of two matrix

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** m1, n1, m2, n2;

***int*** mat1[20][20], mat2[20][20], mat3[20][20];

    printf(**"**Enter no. of rows(m) and no. of columns(n) for 1st matrix.\n**"**);

    scanf(**"**%d**"**, &m1);

    scanf(**"**%d**"**, &n1);

    printf(**"**Enter no. of rows(m) and no. of columns(n) for 2nd matrix.\n**"**);

    scanf(**"**%d**"**, &m2);

    scanf(**"**%d**"**, &n2);

    if (n1 != m2)

    {

        printf(**"**Matrix multiplication not possible.\n**"**);

        return 0;

    }

    printf(**"**Enter matrix 1 rowwise\n**"**);

    for (***int*** i = 0; i < m1; i++)

    {

        for (***int*** j = 0; j < n1; j++)

        {

            scanf(**"**%d**"**, &mat1[i][j]);

        }

    }

    printf(**"**Enter matrix 2 rowwise\n**"**);

    for (***int*** i = 0; i < m2; i++)

    {

        for (***int*** j = 0; j < n2; j++)

        {

            scanf(**"**%d**"**, &mat2[i][j]);

        }

    }

    for (***int*** i = 0; i < m1; i++)

    {

        for (***int*** j = 0; j < n2; j++)

        {

            mat3[i][j] = 0;

            for (***int*** k = 0; k < n1; k++)

            {

                mat3[i][j] += mat1[i][k] \* mat2[k][j];

            }

        }

    }

    printf(**"**Product of matrix 1 & 2 is:\n**"**);

    printf(**"**[**"**);

    for (***int*** i = 0; i < m1; i++)

    {

        printf(**"**\n\t**"**);

        for (***int*** j = 0; j < n2; j++)

        {

            printf(**"**%d  **"**, mat3[i][j]);

        }

    }

    printf(**"**\n]\n**"**);

}

### Output

Text

Description automatically generated

## WAP to find the product of boundary elements of a given matrix.

### Algorithm

1. Take the matrix from the user.

2. Declare a variable to store the product.

3. Traverse the matrix.

4. If the element is the first element of the row, then multiply it with the product.

5. Else if the element is the first element of the column, then multiply it with the product.

6. Else if the element is the last element of the row, then multiply it with the product.

7. Else if the element is the last element of the column, then multiply it with the product.

8. Display the product.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** m, n, product = 1;

    printf(**"**Enter no. of rows(m) and no. of columns(n).\n**"**);

    scanf(**"**%d**"**, &m);

    scanf(**"**%d**"**, &n);

***int*** mat[20][20];

    printf(**"**Enter matrix 1 rowwise\n**"**);

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            scanf(**"**%d**"**, &mat[i][j]);

        }

    }

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            if (i == 0 || j == 0 || i == m - 1 || j == n - 1)

            {

                product \*= mat[i][j];

            }

        }

    }

    printf(**"**Product of boundary elements: %d\n**"**, product);

}

### Output

Text

Description automatically generated with low confidence

## WAP to find the maximum element in the given row and minimum element in the given column for a given matrix

### Algorithm

1. Take the matrix from the user.

2. Take the row number from the user.

3. Take the column number from the user.

4. Declare a variable to store the maximum element.

5. Declare a variable to store the minimum element.

6. Traverse the row.

7. If the element is greater than the maximum element, then set the maximum element to the element.

8. Else if the element is less than the minimum element, then set the minimum element to the element.

9. Display the maximum element and the minimum element.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** m, n;

    printf(**"**Enter no. of rows(m) and no. of columns(n).\n**"**);

    scanf(**"**%d**"**, &m);

    scanf(**"**%d**"**, &n);

***int*** mat[20][20];

    printf(**"**Enter matrix 1 rowwise\n**"**);

    for (***int*** i = 0; i < m; i++)

    {

        for (***int*** j = 0; j < n; j++)

        {

            scanf(**"**%d**"**, &mat[i][j]);

        }

    }

***int*** row, col, max;

    printf(**"**Enter the row index to find element in: **"**);

    scanf(**"**%d**"**, &row);

    printf(**"**Enter the column index to find element in: **"**);

    scanf(**"**%d**"**, &col);

**//** row

    max = mat[row][0];

    for (***int*** i = 0; i < n; i++)

    {

        if (mat[row][i] > max)

        {

            max = mat[row][i];

        }

    }

    printf(**"**The maximum element in row index %d is %d\n**"**, row, max);

**//** column

    max = mat[0][col];

    for (***int*** i = 0; i < m; i++)

    {

        if (mat[i][col] > max)

        {

            max = mat[i][col];

        }

    }

    printf(**"**The maximum element in column index %d is %d\n**"**, col, max);

}

### Output

Text

Description automatically generated

## WAP to carry out Left to Right rotations of a 1-D array n times

### Algorithm

1. Take the array from the user.

2. Take the number of rotations from the user.

3. For n rotations,

4. Traverse the array and shift the elements to the right.

5. Set the last element to 0.

6. Display the array.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** n, SIZE;

    printf(**"**Enter size of array: **"**);

    scanf(**"**%d**"**, &SIZE);

    printf(**"**Enter %d elements: **"**, SIZE);

***int*** arr[50];

    for (***int*** i = 0; i < SIZE; i++)

    {

        scanf(**"**%d**"**, &arr[i]);

    }

    printf(**"**\n**"**);

    printf(**"**Enter the number of rotations: **"**);

    scanf(**"**%d**"**, &n);

    for (***int*** i = 0; i < n; i++)

    {

***int*** last = arr[SIZE - 1];

        for (***int*** j = SIZE - 1; j > 0; j--)

        {

            arr[j] = arr[j - 1];

        }

        arr[0] = last;

    }

    printf(**"**\n**"**);

    printf(**"**Array after %d left to right rotations:\n[\n\t**"**, n);

    for (***int*** i = 0; i < SIZE; i++)

    {

        printf(**"**%d, **"**, arr[i]);

    }

    printf(**"**\n]\n**"**);

}

### Output

Graphical user interface, text, application, Excel

Description automatically generated

## WAP to find all the patterns of 0(2+)0 in the given string. Given a string containing 0’s and 2’s. Find the total number of 0(2+)0 patterns in the string and output it.

### Algorithm

1. Take string from the user.

2. Declare a variable to store the count of 0(2+)0 patterns.

3. Traverse the string.

4. If the character is 0, then continue.

5. Else if the character is 2, and if the last character is a 0, then

6. Keep traversing the string and check if the next character is 0 or 2.

7. If the next character is 0, then increment the count and set last\_zero to 1.

8. Else if the next character is 2, then continue.

9. Display the count.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[1024];

    printf(**"**Enter a string of 0's and 2's: **"**);

    scanf(**"**%s**"**, string);

***int*** i = 0, last\_zero = 0, count = 0;

    while (string[i])

    {

        if (string[i] == **'**0**'**)

        {

            last\_zero = 1;

            i++;

            continue;

        }

        else if (string[i] == **'**2**'** && last\_zero)

        {

            last\_zero = 0;

            while (string[i] && string[i] == **'**2**'**)

            {

                i++;

            }

**//** making sure loop exited on 0 instead of null

            if (string[i] == **'**0**'**)

            {

                count++;

                last\_zero = 1;

            }

        }

        i++;

    }

    printf(**"**Number of patterns: %d\n**"**, count);

}

### Output

A screenshot of a computer

Description automatically generated with medium confidence

# Part 2

## Write a C program to count maximum and minimum occurrence for a given character ch in the given string s1.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[200], max\_char = **'**\0**'**, min\_char = **'**\0**'**;

***int*** max\_count = 0, min\_count = 201;

    printf(**"**Enter string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

    for (***int*** i = 0; string[i]; i++)

    {

        if (max\_char != string[i] || min\_count != string[i])

        {

***int*** count = 0;

            for (***int*** j = 0; string[j]; j++)

            {

                if (string[j] == string[i])

                {

                    count++;

                }

            }

            if (count > max\_count)

            {

                max\_count = count;

                max\_char = string[i];

            }

            if (count < min\_count)

            {

                min\_count = count;

                min\_char = string[i];

            }

        }

    }

    printf(**"**The character that appeared the maximum times is: %c\n**"**, max\_char);

    printf(**"**The character that appeared the minimum times is: %c\n**"**, min\_char);

}

### Output

Text

Description automatically generated

## WAP a program to reverse the given string and display the resultant string

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[200], reveresed[200];

    printf(**"**Enter string to reverse: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***int*** length;

    for (length = 0; string[length]; length++)

        ;

    for (***int*** i = length - 1, j = 0; i >= 0; i--, j++)

    {

        reveresed[j] = string[i];

    }

    reveresed[length] = **'**\0**'**;

    printf(**"**Reversed string: %s\n**"**, reveresed);

}

### Output

Text

Description automatically generated

## Write a C program to concatenate the following quotes of Dr. A. P. J “If you fail, never give up because FAIL means “First Attempt in Learning.” “All of us do not have equal talent. But, all of us have an equal opportunity to develop our talents.” “Excellence is a continuous process & not an accident” Finally display the output in the following format Dr. A. P. J quotes follows......................(with the resultant string).

### Code

#include **<**stdio.h**>**

#include **<**string.h**>**

***int*** main()

{

***char*** main\_quote[] = **"**Dr. A. P. J. quotes follows....\n**"**;

***char*** quote1[] = **"**If you fail, never give up because FAIL means \"First Atetmpt in Learning.\"**"**;

***char*** quote2[] = **"**All of us do not have equal talent. But, all of us have an equal opportunity to develop our talents.**"**;

***char*** quote3[] = **"**Excellence is a continuous process & not an accident**"**;

    strcat(main\_quote, quote1);

    strcat(main\_quote, quote2);

    strcat(main\_quote, quote3);

    printf(**"**%s\n**"**, main\_quote);

}

### Output

Text

Description automatically generated

## Considering the given quotes of Dr. A. P. J stated above, find the frequency of occurrence of all the special characters in the given quotes.

### Code

#include **<**stdio.h**>**

#include **<**string.h**>**

#include **<**ctype.h**>**

***int*** main()

{

***char*** main\_quote[] = **"**Dr. A. P. J. quotes follows....\n**"**;

***char*** quote1[] = **"**If you fail, never give up because FAIL means \"First Atetmpt in Learning.\"**"**;

***char*** quote2[] = **"**All of us do not have equal talent. But, all of us have an equal opportunity to develop our talents.**"**;

***char*** quote3[] = **"**Excellence is a continuous process & not an accident**"**;

    strcat(main\_quote, quote1);

    strcat(main\_quote, quote2);

    strcat(main\_quote, quote3);

***int*** count = 0;

    for (***int*** i = 0; main\_quote[i]; i++)

        if (ispunct(main\_quote[i]))

            count++;

    printf(**"**Total number of special characters in APJ's quotes: %d\n**"**, count);

}

### Output

A screenshot of a computer

Description automatically generated with medium confidence

## Write a C program to count the number of words, number of sentences and finally the length of the string without using any built- in functions of strings.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***int*** words = 0, sentences = 0, length;

***char*** string[200];

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

**//** length

    for (length = 0; string[length]; length++)

        ;

**//** words and sentences

***int*** i = 0, sentence\_start = 0, word\_start = 0;

    while (string[i])

    {

        if (string[i] == **'** **'**)

        {

            word\_start = i + 1;

            words++;

        }

        if (string[i] == **'**.**'** || string[i] == **'**!**'** || string[i] == **'**?**'**)

        {

            sentences++;

            sentence\_start = i + 1;

        }

        i++;

    }

    printf(**"**Words: %d\nSentences: %d\nLength: %d\n**"**, words, sentences, length);

}

### Output

Graphical user interface, text

Description automatically generated

## WAP to remove all characters from the given string except the alphabet character and display the final string

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[200];

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***int*** i = 0, j = 0;

    while (string[i])

    {

        if (string[i] >= **'**a**'** && string[i] <= **'**z**'** || string[i] >= **'**A**'** && string[i] <= **'**Z**'**)

        {

            string[j] = string[i];

            j++;

        }

        i++;

    }

    string[j] = **'**\0**'**;

    printf(**"**%s\n**"**, string);

}

### Output

Graphical user interface, text, Excel

Description automatically generated

## Write a program in C to replace the spaces of a given string “ARISE, AWAKE, and STOP NOT UNTIL THE GOAL is ACHIEVED” with a special character of your choice.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[] = **"**ARISE, AWAKE, and STOP NOT UNTIL THE GOAL is ACHIEVED**"**;

***int*** i = 0, j = 0;

    while (string[i])

    {

        if (string[i] == **'** **'**)

        {

            string[j] = **'**$**'**;

            j++;

        }

        else

        {

            string[j] = string[i];

            j++;

        }

        i++;

    }

    string[j] = **'**\0**'**;

    printf(**"**%s\n**"**, string);

}

### Output

Graphical user interface, text

Description automatically generated with medium confidence

## WAP to split the given sentence into words and display the words one after the other.

### Code

#include **<**stdio.h**>**

#include **<**stdlib.h**>**

***int*** main()

{

***char*** \*string = (***char*** \*)malloc(200);

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***int*** i = 0, j = 0;

    while (string[i])

    {

        if (string[i] == **'** **'**)

        {

            string[i] = **'**\0**'**;

            printf(**"**%s\n**"**, string);

            string = string + i + 1;

            i = 0;

        }

        i++;

    }

    string[i] = **'**\0**'**;

    printf(**"**%s\n**"**, string);

}

### Output

Text

Description automatically generated

## Write a program to find the number of times a given word 'the” appears in the given string “Best brain of the nation may be found in the last benches of the classroom”.

### Algorithms

1. Define the string.

2. Traverse the string.

3. If a character is a space, then check if the word "the" is present or not.

4. If the word "the" is present, then increment the count.

5. Else skip the character.

6. Display the count.

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[] = **"**Best brain  of the nation may be found in the last benches of the classroom**"**;

***int*** i = 0, j = 0, count = 0;

    while (string[i])

    {

        if (string[i] == **'** **'**)

        {

            j = i + 1;

            while (string[j] == **'** **'**)

                j++;

            if (string[j] == **'**t**'** && string[j + 1] == **'**h**'** && string[j + 2] == **'**e**'**)

                count++;

        }

        i++;

    }

    printf(**"**%d\n**"**, count);

}

### Output

Table

Description automatically generated with low confidence

## WAP to remove the duplicate characters from the given string, S and print the resultant string, RS (Hint: S <-AabCaBcdEGzYz , RS <-AbCdEGzY)

### Code

#include **<**stdio.h**>**

***char*** toLower(***char*** c)

{

    if (c >= **'**A**'** && c <= **'**Z**'**)

        return c + 32;

    else

        return c;

}

***int*** charInArray(***char*** c, ***char*** \*array, ***int*** size)

{

***int*** i;

    for (i = 0; i < size; i++)

    {

        if (toLower(c) == toLower(array[i]))

            return 1;

    }

    return 0;

}

***int*** main()

{

***char*** string[200];

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***char*** result\_string[200];

***int*** length = 0;

***int*** i = 0;

    while (string[i])

    {

        if (charInArray(string[i], result\_string, length))

        {

            i++;

            continue;

        }

        result\_string[length] = string[i];

        length++;

        i++;

    }

    result\_string[length] = **'**\0**'**;

    printf(**"**%s\n**"**, result\_string);

}

### Output

Text

Description automatically generated with medium confidence

## WAP to remove all characters from the given string except the alphabet character and display the final string (without using the second string)

### Code

#include **<**stdio.h**>**

***int*** main()

{

***char*** string[200];

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***int*** i = 0, j = 0;

    while (string[i])

    {

        if (string[i] >= **'**a**'** && string[i] <= **'**z**'** || string[i] >= **'**A**'** && string[i] <= **'**Z**'**)

        {

            string[j] = string[i];

            j++;

        }

        i++;

    }

    string[j] = **'**\0**'**;

    printf(**"**%s\n**"**, string);

}

### Output

Graphical user interface, text

Description automatically generated

## WAP to insert a substring anywhere in between the given string except in the first and last position of the string. Finally display the resultant string on inserting the sub-string.

### Code

#include **<**stdio.h**>**

#include **<**stdlib.h**>**

**//** function to determine length of string

***int*** strlen(***char*** \*string)

{

***int*** i = 0;

    while (string[i])

        i++;

    return i;

}

***int*** main()

{

***char*** \*string = (***char*** \*)malloc(200);

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, string);

    getchar();

***char*** sub\_string[200];

    printf(**"**Enter a sub-string: **"**);

    scanf(**"**%[^\n]s**"**, sub\_string);

    getchar();

    printf(**"**Enter the position: **"**);

***int*** position, k = 0;

    scanf(**"**%d**"**, &position);

    while (string[position] != **'**\0**'** && sub\_string[k] != **'**\0**'**)

    {

***int*** i;

        for (i = strlen(string); i >= position; i--)

        {

            string[i + 1] = string[i];

        }

        string[position] = sub\_string[k];

        position++;

        k++;

    }

    printf(**"**Output: %s\n**"**, string);

}

### Output

Graphical user interface, text, Excel

Description automatically generated with medium confidence

## WAP to reverse words in the given strings and display the reversed string

### Code

#include **<**stdio.h**>**

***void*** reverse(***char*** \*begin, ***char*** \*end)

{

***char*** temp;

    while (begin < end)

    {

        temp = \*begin;

        \*begin++ = \*end;

        \*end-- = temp;

    }

}

**//** Function to reverse words\*/

***void*** reverseWords(***char*** \*s)

{

***char*** \*word\_begin = s;

**//** Word boundary

***char*** \*temp = s;

    while (\*temp)

    {

        temp++;

        if (\*temp == **'**\0**'**)

        {

            reverse(word\_begin, temp - 1);

        }

        else if (\*temp == **'** **'**)

        {

            reverse(word\_begin, temp - 1);

            word\_begin = temp + 1;

        }

    }

**//** Reverse the entire string

    reverse(s, temp - 1);

}

**//** Driver Code

***int*** main()

{

***char*** s[200];

    printf(**"**Enter a string: **"**);

    scanf(**"**%[^\n]s**"**, s);

    getchar();

***char*** \*temp = s;

    reverseWords(s);

    printf(**"**%s**"**, s);

    return 0;

}

### Output

Graphical user interface

Description automatically generated