Experiment No.5

Name: Aditi Anil Chougale.
Div:CE2

Batch:A

PRN:B25CE1081

Email Address Validator

1.Research:

Email validation is essential because it protects sender reputation by reducing bounces and spam complaints, which in turn improves email deliverability and prevents your domain from being blacklisted. It also enhances data accuracy by catching typos and invalid addresses, saving resources, and ensuring that campaigns reach their intended recipients.

2.Analysis:

Email address validation is important to prevent misuse and fraud.

- **Improves deliverability:** Validating emails ensures that messages are sent to real, active addresses, drastically reducing bounce rates and increasing the likelihood they will reach the recipient's inbox.
- **Protects sender reputation:** High bounce rates can damage your sender reputation, causing internet service providers (ISPs) to flag your emails as spam or send them directly to the spam folder. Sending to spam traps (invalid addresses used by ISPs to catch spammers) also harms your reputation and can lead to your domain being blacklisted.
- **Enhances data quality:** Validation removes outdated, invalid, or inactive addresses from your contact lists, providing a more accurate and reliable database for your marketing efforts.

- **Reduces costs:** Many email services charge based on the number of emails sent. Validating your list eliminates the waste of sending emails to invalid addresses, which saves money.

3.Ideate:

The project idea is a "Command-Line Email Address Validator".

- my_strlen: Used to check the total length of the input and ensure email parts are not empty.
- my_strcmp: Used to check for the "quit" command to exit the program.
- my_strcpy & my_strcat: These are used to manage the input string and to build informative, detailed error messages for the user.

We will build a program to check if the email address is valid or not.

Build:

```
#include <stdio.h>

// Function to calculate string length (my_strlen)
int my_strlen(char *str) {
    int len = 0;
    while (str[len] != '\0') {
        len++;
    }
    return len;
}

// Function to compare two strings (my_strcmp)
int my_strcmp(char *str1, char *str2) {
    int i = 0;
    while (str1[i] != '\0' && str2[i] != '\0' && str1[i] == str2[i]) {
        i++;
    }
    return str1[i] - str2[i];
}
```

```c
// Function to copy a string (my_strcpy)
void my_strcpy(char *dest, char *src) {
    int i = 0;
    while (src[i] != '\0') {
        dest[i] = src[i];
        i++;
    }
    dest[i] = '\0'; // Ensure destination is null-terminated
}

// Function to concatenate two strings (my_strcat)
void my_strcat(char *dest, char *src) {
    int dest_len = my_strlen(dest);
    int i = 0;
    while (src[i] != '\0') {
        dest[dest_len + i] = src[i];
        i++;
    }
    dest[dest_len + i] = '\0'; // Ensure concatenated string is null-terminated
}

// --- Validation Function ---

// Returns 1 if valid, 0 if invalid. Builds an error message in result_msg.
int validateEmail(char *email, char *result_msg) {
    int len = my_strlen(email);
    // Use my_strcpy to start building the message
    my_strcpy(result_msg, "Status: Invalid. Reason: ");

    if (len < 5) {
        my_strcat(result_msg, "Email is too short.");
        return 0;
    }

    int atPos = -1;
    int dotPos = -1;
    for (int i = 0; i < len; i++) {
        if (email[i] == '@') {
            if (atPos != -1) { // Found a second '@'
                my_strcat(result_msg, "Contains more than one '@' symbol.");
                return 0;
            }
           atPos = i;
        } else if (email[i] == '.') {
            dotPos = i;
        }
    }

    if (atPos == -1) {
        my_strcat(result_msg, "Missing '@' symbol.");
        return 0;
    }

    if (atPos == 0) {
        my_strcat(result_msg, "Missing local part (before '@').");
        return 0;
    }

    if (dotPos == -1 || dotPos < atPos) {
         my_strcat(result_msg, "Missing or misplaced '.' symbol after '@'.");
        return 0;
    }

    // If all checks pass
    my_strcpy(result_msg, "Status: Valid.");
    return 1;
}

// --- Main Function (Menu) ---

int main() {
    char email[100];
    char result[150];
```

```
        printf("\nEnter email address to validate (or type 'quit' to exit): ");
        scanf("%s", email);
        printf("Your email address:%s\n",email);
        // Use my_strcmp to check for the exit command
        if (my_strcmp(email, "quit") == 0) {
            printf("Exiting program. Goodbye!\n");

        }

        validateEmail(email, result);
        printf("%s\n", result);


    return 0;
}
```

5.Test:

<u>1)For valid case:</u>

```
Enter email address to validate (or type 'quit' to exit): abcd1234@gmail.com
Your email address:abcd1234@gmail.com
Status: Valid.
```

<u>2)For invalid case:</u>

```
Enter email address to validate (or type 'quit' to exit): abc22@42@gmail.com
Your email
address:abc22@42@gmail.com
Status: Invalid. Reason: Contains more than one '@' symbol.
```

6.Implementation:

Here we have used user defined functions instead of built in functions.Validation of email address is done by using string functions. We have checked whether the entered email address is valid or not. We have done various string operations like copying the string,calculating length of the string,comparing two strings and concatenating the strings.

Reference:

https://kenscio.com/why-email-address-validation-is-essential-for-effective-campaigns/?srsltid=AfmBOoqiwK0Uk7PklUtlNxnDqXSKz9XJz_ehPAhEGMu9ycZaPZxD3RMN

Github link: