Set Questions

Set 1

1.Anagram

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

int main() {
    int b = 0, c = 0, k = 0;
    char s[100], g[100];
    fgets(s, 100, stdin);
    fgets(g, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    g[strcspn(g, "\n")] = 0;
    int len_s = strlen(s);
    int len_g = strlen(g);
    if (len_s != len_g) {
        printf("Strings must be of the same length\n");
        return 1;
    }
    int visited[100] = {0};
    for (int i = 0; i < len_s; i++) {
            b=0;
            for (int j = 0; j < len_g; j++){
                if(!visited[j]&&s[i] == g[j]){
                        visited[j] = 1;
                        b=1;
                        break;
                    }
               }
                if(!b){
                    printf("False");
                    return 0;
                 }
    }
    printf("True");
    return 0;
}
```

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

int main() {
    char s[100], g[100];
    int count_s[256] = {0};
    int count_g[256] = {0};
    fgets(s, 100, stdin);
    fgets(g, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    g[strcspn(g, "\n")] = 0;
    int len_s = strlen(s);
    int len_g = strlen(g);
    if (len_s != len_g) {
        printf("Strings must be of the same length\n");
        return 1;
    }
    for (int i = 0; i < len_s; i++) {
        count_s[s[i]]++;
        count_g[g[i]]++;
    }
    for (int i = 0; i < 256; i++) {
        if (count_s[i] != count_g[i]) {
            printf("False\n");
            return 0;
        }
    }
    printf("True\n");
    return 0;
}
```

2. 5. Longest Palindromic Substring

```c
#include <stdio.h>
#include <string.h>
#include <string.h>
#include <stdlib.h>
// function to find it is palindrome or not
int isPalindrome(char str[],int start,int end){
     while(start<=end){
         if(str[start++]!=str[end--]){
             return 0;
         }
     }
    return 1;
}
int main() {
  char str[100];
    fgets(str, 100, stdin);
    str[strcspn(str, "\n")] = 0;
//   maxlen to store the max len of string and use is to check
with upcomming palindrome
   int maxLen=0,start;
   for(int i=0;i<strlen(str);i++){
       for(int j=i;j<strlen(str);j++){
        //   only if it is a palindrome and its length is greater
than previous palindrome length
           if(isPalindrome(str,i,j)&&maxLen<(j-i+1)){
               //   j-i+1 is to find the length of the word
               maxLen=j-i+1;
               start=i;
           }
       }
   }
   char* longest = (char*)malloc((maxLen + 1) * sizeof(char));
    strncpy(longest, str + start, maxLen);
    longest[maxLen] = '\0';
    printf("%s",longest);
    return 0;
}
```

3.Greatest English Letter in Upper and Lower Case

```
#include <stdio.h>
#include <string.h>
#include<ctype.h>
int main() {
    int m=0,n=0;
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    int len = strlen(s);
    for(int i=0;i<len;i++){
        for(int j=0;j<len;j++){
           if(tolower(s[i])==s[j]&&i!=j){
            m=(int)(s[i])>m?(int)(s[i]):m;
           }
           if(toupper(s[i])==s[j]&&i!=j){
            n=(int)(s[i])>n?(int)(s[i]):n;
           }
        }
    }
        if(!(m||n)){
            printf("");
            return 0;
        }
    printf("%c",m);
    return 0;
}
```

4.Valid Parentheses

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

int main() {
    int k = 0;
    char s[100], ch[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    int len = strlen(s);

    for (int i = 0; i < len; i++) {
        char c = s[i];
        if (c == '(' || c == '{' || c == '[') {
            ch[k++] = c;
        } else {
            if (k == 0) {
                printf("false\n");
                return 0;
            }
            if (c == ')' && ch[k - 1] != '(') {
                printf("false\n");
                return 0;
            }
            if (c == '}' && ch[k - 1] != '{') {
                printf("false\n");
                return 0;
            }
            if (c == ']' && ch[k - 1] != '[') {
                printf("false\n");
                return 0;
            }
            k--;
        }
    }

    if (k == 0) {
        printf("true\n");
    } else {
        printf("false\n");
    }

    return 0;
}
```

5.Minimum Length of String After Deleting Similar Ends.

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

int main() {
    char s[100], ch[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    int len = strlen(s);
    int l=0,r=len-1,k=0;
   while(l<r) {
        if(s[l]==s[r]){
            char c=s[l];
            while(l<=r&&c==s[l]){
                l++;
                k+=1;
            }
            while(r>=l&&c==s[r]){
                r--;
                k+=1;
            }
        }
        else
            break;
    }
    printf("%d",len-k);
    return 0;
}
```

6.Simple FLAMES Game

```
#include <stdio.h>
#include <string.h>
#include<ctype.h>
int main() {
    char s1[100];
    fgets(s1, 100, stdin);
    s1[strcspn(s1, "\n")] = 0;
     for (int i = 0; s1[i]; i++) {
        s1[i] = tolower(s1[i]);
    }
    char s2[100];
    fgets(s2, 100, stdin);
    s2[strcspn(s2, "\n")] = 0;
     for (int i = 0; s2[i]; i++) {
        s2[i] = tolower(s2[i]);
    }
    char Flames[6]={'F','L','A','M','E','S'};
    int visited[100] = {0};
    int count=0,n=6,c=0;
    int v[100]={0};
    for(int i=0;i<strlen(s1);i++){
        for(int j=0;j<strlen(s2);j++){
           if (!v[j] && s1[i] == s2[j]){
                    c+=2;
                    v[j]=1;
                    break;
                }
            }
        }
    count=strlen(s1)+strlen(s2)-c;
    int pos=0;
  while (n > 1) {
        pos = (pos + count - 1) % n;
        for (int i = pos; i < n - 1; i++) {
            Flames[i] = Flames[i + 1];
        }
        n--;
    }

    printf("%c",Flames[0]);
    return 0;
}
```

7.Kth Distinct String in an Array

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

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    char words[100][100];
    int visited[100] = {0};
    int k = 0, m = 0;
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    for (int i = 0; i < k; i++) {
        if (!visited[i]) {
            for (int j = i + 1; j < k; j++) {
                if (strcmp(words[i], words[j]) == 0) {
                    visited[j] = 1;
                    visited[i] = 1;
                }
            }
        }
    }
    char unique[100][100];
    for (int i = 0; i < k; i++) {
        if (!visited[i]) {
            strcpy(unique[m], words[i]);
            m++;
        }
    }
    int kth;
    scanf("%d", &kth);
    if (kth <= m) {
        printf("%s", unique[kth - 1]);
    } else {
        printf("!");
    }

    return 0;
}
```

8. 131. Palindrome Partitioning

in java

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class PalindromePartitioning {

    private boolean isPalindrome(String s, int start, int end) {
        while (start < end) {
            if (s.charAt(start) != s.charAt(end)) {
                return false;
            }
            start++;
            end--;
        }
        return true;
    }

    private void backtrack(String s, int start, List<String>
currentList, List<List<String>> result) {
        if (start == s.length()) {
            result.add(new ArrayList<>(currentList));
            return;
        }

        for (int end = start; end < s.length(); end++) {
            if (isPalindrome(s, start, end)) {
                currentList.add(s.substring(start, end + 1));
                backtrack(s, end + 1, currentList, result);
                currentList.remove(currentList.size() - 1);
            }
        }
    }

    public List<List<String>> partition(String s) {
        List<List<String>> result = new ArrayList<>();
        if (s.length() == 1) {
            return result;
        }
        backtrack(s, 0, new ArrayList<>(), result);
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        PalindromePartitioning pp = new PalindromePartitioning();

        System.out.print("Enter a string: ");
        String input = scanner.nextLine();

        List<List<String>> result = pp.partition(input);
        if (input.length() == 1) {
            System.out.println("-2");
        } else {
                System.out.println(result);

        }

        scanner.close();
    }
}
```

in c

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

#define MAX_LEN 100

// Function to check if a substring is a palindrome
bool isPalindrome(char *s, int start, int end) {
    while (start < end) {
        if (s[start] != s[end]) {
            return false;
        }
        start++;
        end--;
    }
    return true;
}

// Helper function for backtracking
void backtrack(char *s, int start, char **currentList, int
currentListSize, char ***result, int *returnSize, int
**returnColumnSizes) {
    int len = strlen(s);
    if (start == len) {
        result[*returnSize] = (char **)malloc(currentListSize *
sizeof(char *));
```

```
        (*returnColumnSizes)[*returnSize] = currentListSize;
        for (int i = 0; i < currentListSize; i++) {
            result[*returnSize][i] = strdup(currentList[i]);
        }
        (*returnSize)++;
        return;
    }

    for (int end = start; end < len; end++) {
        if (isPalindrome(s, start, end)) {
            char *substring = (char *)malloc((end - start + 2) *
sizeof(char));
            strncpy(substring, s + start, end - start + 1);
            substring[end - start + 1] = '\0';
            currentList[currentListSize] = substring;
            backtrack(s, end + 1, currentList, currentListSize +
1, result, returnSize, returnColumnSizes);
            free(substring);
        }
    }
}

// Function to find all palindrome partitions
char ***partition(char *s, int *returnSize, int
**returnColumnSizes) {
    *returnSize = 0;
    int len = strlen(s);
    char ***result = (char ***)malloc(MAX_LEN * sizeof(char **));
    *returnColumnSizes = (int *)malloc(MAX_LEN * sizeof(int));
    char **currentList = (char **)malloc(len * sizeof(char *));
    backtrack(s, 0, currentList, 0, result, returnSize,
returnColumnSizes);
    free(currentList);
    return result;
}

// Main function to read input and call the partition function
int main() {
    char input[MAX_LEN];
    printf("Enter a string: ");
    scanf("%s", input);

    int returnSize;
    int *returnColumnSizes;
    char ***result = partition(input, &returnSize,
&returnColumnSizes);

    if (returnSize == 0) {
        printf("[[]]\n");
    } else {
        printf("[");
        for (int i = 0; i < returnSize; i++) {
            printf("[");
            for (int j = 0; j < returnColumnSizes[i]; j++) {
                printf("\"%s\"", result[i][j]);
                if (j < returnColumnSizes[i] - 1) {
                    printf(",");
                }
                free(result[i][j]);
            }
            printf("]");
            if (i < returnSize - 1) {
                printf(",");
            }
        }
        printf("]");
    }

    free(returnColumnSizes);
    free(result);

    return 0;
}
```

Set 2

318. Maximum Product of Word Lengths

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

int calculateBitmask(char *word) {
    int bitmask = 0;
    for (int i = 0; word[i] != '\0'; i++) {
        bitmask |= (1 << (word[i] - 'a'));
    }
    return bitmask;
}
int maxProduct(char** words, int wordsSize) {
    int bitmasks[wordsSize];
    for (int i = 0; i < wordsSize; i++) {
        bitmasks[i] = calculateBitmask(words[i]);
    }
    int maxProduct = 0;
    for (int i = 0; i < wordsSize; i++) {
        for (int j = i + 1; j < wordsSize; j++) {
            if ((bitmasks[i] & bitmasks[j]) == 0) {
                int product = strlen(words[i]) * strlen(words[j]);
                if (product > maxProduct) {
                    maxProduct = product;
                }
            }
        }
    }
    return maxProduct;
}
```

Own

```c
// Online C compiler to run C program online
#include <stdio.h>
#include<string.h>
#include<ctype.h>

```

```c
int main() {
   char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    char words[100][100];
    int k = 0,max=0,a=0,product=0;
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    for(int i=0;i<k;i++){
        for(int j=i+1;j<k;j++){
            if(strlen(words[i])==strlen(words[j])){
            char s1[100],s2[100];
            strcpy(s1,words[i]),strcpy(s2,words[j]);
             a=0;
              for(int m=0;m<strlen(words[i]);m++){
                    for(int n=0;m<strlen(words[j]);n++){
                       if(s1[m]==s2[n]){
                           a=1;
                           break;
                       }
                    }
                if(a)
                    break;
              }
              if(!a){
                 product=strlen(words[i])*strlen(words[j]);
                max=max>product?max:product;
              }
          }
        }
      }
    printf("%d",max);
    return 0;
}
```

383. Ransom Note

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

int main() {
    char ransomNote[100];
    fgets(ransomNote, 100, stdin);
    ransomNote[strcspn(ransomNote, "\n")] = 0;
    char magazine[100];
    int a;
    fgets(magazine, 100, stdin);
    magazine[strcspn(magazine, "\n")] = 0;
    int visited[100] = {0};
    for(int i=0;i<strlen(ransomNote);i++){
        a=0;
        for(int j=0;j<strlen(magazine);j++){
             if(!visited[j]&&ransomNote[i]==magazine[j]){
                a=1;
                visited[j]=1;
                break;
            }
        }
        if(!a){
            printf("False");
          return 0;
         }
    }
    printf("TRUE");
    return 0;
}
```

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

```
bool canConstruct(char* ransomNote, char* magazine) {
    int magazineCount[26] = {0}; // Array to count occurrences of
each character ('a' to 'z')
    // Count occurrences of each character in magazine
    for (int i = 0; i < strlen(magazine); i++) {
        magazineCount[magazine[i] - 'a']++;
    }
    // Check each character in ransomNote against magazineCount
    for (int i = 0; i < strlen(ransomNote); i++) {
        int index = ransomNote[i] - 'a';
        if (magazineCount[index] > 0) {
            magazineCount[index]--;
        } else {
            return false; // If character is not available in
magazine or count is exhausted
        }
    }
    return true; // If all characters in ransomNote can be
constructed
}
```

0 ms code ```

```
bool canConstruct(char* ransomNote, char* magazine) {

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

        char *ptr = strchr(magazine, ransomNote[i]);

        if (ptr == NULL) {

            return false;

        }

        *ptr = '*';
```

```
    }

    return true;

}
```

451. Sort Characters By Frequency

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define MAX_CHAR 256
// Structure to store characters and their frequencies
typedef struct {
    char character;
    int frequency;
} CharFreq;
// Comparison function to sort CharFreq by frequency in descending
order
int compare(const void* a, const void* b) {
    CharFreq* cf1 = (CharFreq*)a;
    CharFreq* cf2 = (CharFreq*)b;
    return cf2->frequency - cf1->frequency;
}
char* frequencySort(char* s) {
    int n = strlen(s);
    int freq[MAX_CHAR] = {0};
    // Count frequency of each character
    for (int i = 0; i < n; i++) {
        freq[(unsigned char)s[i]]++;
    }
    // Create an array of CharFreq to store characters and their
frequencies
    CharFreq charFreq[MAX_CHAR];
    int count = 0;
    for (int i = 0; i < MAX_CHAR; i++) {
        if (freq[i] > 0) {
```

```c
              charFreq[count].character = i;
             charFreq[count].frequency = freq[i];
              count++;
          }
      }
      qsort(charFreq, count, sizeof(CharFreq), compare);
      // Reconstruct the string based on the sorted characters
      int index = 0;
      for (int i = 0; i < count; i++) {
          for (int j = 0; j < charFreq[i].frequency; j++) {
              s[index++] = charFreq[i].character;
          }
      }
      s[index] = '\0'; // Null terminate the sorted string
           return s;
  }
```

OWN

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

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0; // Remove the newline character

    int visited[100] = {0};
    int count, max = 0;
    char s1;

    // Determine the character with the maximum frequency
    for (int i = 0; i < strlen(s); i++) {
        if (!visited[i]) {
            count = 1;
            for (int j = i + 1; j < strlen(s); j++) {
                if (s[i] == s[j]) {
                    count++;
                    visited[j] = 1;
                }
            }
            if (max < count) {
                max = count;
                s1 = s[i];
            }
        }
    }

    // Rearrange the string based on the frequency of the
character
    int k = 0;
    for (int i = 0; i < strlen(s); i++) {
        if (s[i] == s1) {
            for (int j = i; j > k; j--) {
                s[j] = s[j - 1];
            }
            s[k] = s1;
            k++;
        }
    }

    printf("%s\n", s);
    return 0;
}
```

387. First Unique Character in a String

```
int firstUniqChar(char* s) {
    int visited[256] = {0};  // Adjusted to 256 to cover all ASCII
characters
    int len = strlen(s);
    int count;
    // Count the frequency of each character
    for (int i = 0; i < len; i++) {
        visited[(unsigned char)s[i]]++;
    }
    // Find the first unique character
    for (int i = 0; i < len; i++) {
        if (visited[(unsigned char)s[i]] == 1) {
            return i;
        }
    }
    return -1;  // Return -1 if no unique character is found
}
```

OWN

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

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0; // Remove the newline character
    int len = strlen(s);
    for (int i = 0; i < len; i++) {
            int j;
            for ( j = 0; j < len; j++) {
                if (i != j && s[i] == s[j]) {
                    break;
                }
            }
            if (s[i]!=s[j]) {
                printf("%c",s[i]);
                return 0;
            }
    }
     printf("None");
    return 0;
}
```

1796. Second Largest Digit in a String

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0; // Remove the newline character
    int len = strlen(s);
    int m1=-1,m2=-1;
    for (int i = 0; i < len; i++) {
               if(isdigit(s[i])){
                   int digit = s[i]- '0';
                   if(m1<digit){
                        m2=m1;
                       m1=digit;
                   }
                   else if(m1>digit&&m2<digit){
                       m2=digit;
                   }
               }
    }
     printf("%d",m2);
    return 0;
}
```

3006. Find Beautiful Indices in the Given Array I

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

// Comparator function for qsort
int compare(const void *a, const void *b) {
    return (*(int *)a - *(int *)b);
}

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    char a[100];
    fgets(a, 100, stdin);
    a[strcspn(a, "\n")] = 0;
    char b[100];
    fgets(b, 100, stdin);
    b[strcspn(b, "\n")] = 0;
    int k;
    scanf("%d",&k);
    int len_s = strlen(s);
    int len_a = strlen(a);
    int len_b = strlen(b);

  // Dynamically allocate memory for indices arrays, assuming a
worst-case size
    int *indices_a = (int *)malloc(len_s * sizeof(int));
    int *indices_b = (int *)malloc(len_s * sizeof(int));
    if (indices_a == NULL || indices_b == NULL) {
        return 0;
    }
    int count_a = 0, count_b = 0;

    // Find all starting indices where substring a is found
    for (int i = 0; i <= len_s - len_a; i++) {
        if (strncmp(&s[i], a, len_a) == 0) {
            indices_a[count_a++] = i;
        }
    }

    // Find all starting indices where substring b is found
    for (int i = 0; i <= len_s - len_b; i++) {
        if (strncmp(&s[i], b, len_b) == 0) {
            indices_b[count_b++] = i;
        }
    }

    // Dynamically allocate memory for beautiful indices
    int *beautiful_indices = (int *)malloc(len_s * sizeof(int));
    if (beautiful_indices == NULL) {
        free(indices_a);
        free(indices_b);
        return 0;
    }
    int beautiful_count = 0;

    // Check each index in indices_a against all indices in
indices_b
    for (int i = 0; i < count_a; i++) {
        for (int j = 0; j < count_b; j++) {
            if (abs(indices_a[i] - indices_b[j]) <= k) {
                beautiful_indices[beautiful_count++] =
indices_a[i];
                break; // Once we find a valid b index for a, we
can stop checking further b indices for this a
            }
        }
    }

    // Sort the beautiful indices
    qsort(beautiful_indices, beautiful_count, sizeof(int),
compare);

    // Free the memory allocated for indices arrays
    free(indices_a);
    free(indices_b);


    // If no beautiful indices found, free the allocated memory
and return NULL
    if (beautiful_count == 0) {
        free(beautiful_indices);
        return 0;
    }
    // Print the beautiful indices
    for (int i = 0; i < beautiful_count; i++) {
        printf("%d ", beautiful_indices[i]);
    }
    printf("\n");

    return 0;
}
```

Set 4

1.Make Three Strings Equal by deleting rightmost element

```
#include <stdio.h>
#include <string.h>
int main()
{
    char str1[100], str2[100], str3[100];
    int count=0;
    fgets(str1, sizeof(str1), stdin);
    str1[strcspn(str1, "\n")] = '\0';
    fgets(str2, sizeof(str2), stdin);
    str2[strcspn(str2, "\n")] = '\0';
    fgets(str3, sizeof(str3), stdin);
    str3[strcspn(str3, "\n")] = '\0';
     int ts=strlen(str1)+strlen(str2)+strlen(str3);
        if (strcmp(str1, str2) == 0 && strcmp(str1, str3) == 0) {
        printf("0");
         }
        else {
        printf("Strings are not identical.");
         }
        for(int i=0;i<100;i++){
            if(str1[i]!=str2[i]||str1[i]!=str3[i]||str3[i]!=str2[i])
{
                if(i==0){
                    printf("-1");
                return -1;
                }
             }
        }
        for(int i=0;i<ts;i++){

if(str1[i]==str2[i]&&str1[i]==str3[i]&&str3[i]==str2[i]){
                count+=1;
            }
        }
        int n;
        n=ts-count;
        printf("%d",n);
        return 0;
    }
```

2.First non repeated character and last repeated character in a word.

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

int main() {
    char str[100];
    int f1 = 0, f2 = 0;

    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';

    for (int i = 0; i < strlen(str); i++) {
        f1 = 1;
        for (int j = i + 1; j < strlen(str); j++) {
            if (str[i] == str[j]) {
                f1 = 0;
                break;
            }
        }
        if (f1) {
            printf("First non-repeated character: %c\n", str[i]);
            break;
        }
    }

    for (int i = strlen(str) - 1; i >= 0; i--) {
        for (int j = i - 1; j >= 0; j--) {
            if (str[i] == str[j]) {
                printf("Last repeated character: %c\n", str[j]);
                f2 = 1;
                break;
            }
        }
        if (f2) {
            break;
        }
    }

    return 0;
}
```

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

int main() {
    char ch[100];
    char ch1[100]={0};

    fgets(ch,sizeof(ch),stdin);
    ch[(strlen(ch)-1)]='\0';

    for(int i=0;i<strlen(ch);i++){
        for(int j=i+1;j<strlen(ch);j++){
            if(ch[i]==ch[j]){
                ch1[i]=1;
            }
        }
    }
    for(int i=0;i<strlen(ch);i++){
        if(ch1[i]!=1){
            printf("%c\n",ch[i]);
            break;
        }
    }
     for(int i=strlen(ch)-1;i>=0;i--){
        if(ch1[i]==1){
            printf("%c\n",ch[i]);
            break;
        }
    }
    return 0;
}
```

3.Bulls and Cows game

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

int main() {
    int b = 0, c = 0, k = 0;
    char s[100], g[100];

    // Read input strings
    fgets(s, 100, stdin);
    fgets(g, 100, stdin);

    // Remove newline character if present
    s[strcspn(s, "\n")] = 0;
    g[strcspn(g, "\n")] = 0;
```

```c
    int len_s = strlen(s);
    int len_g = strlen(g);

    // Check if lengths are the same
    if (len_s != len_g) {
        printf("Strings must be of the same length\n");
        return 1;
    }

    int visited_s[100] = {0};
    int visited_g[100] = {0};

    // Count A
    for (int i = 0; i < len_s; i++) {
        if (s[i] == g[i]) {
            visited_s[i] = 1;
            visited_g[i] = 1;
            b++;
        }
    }

    // Count B
    for (int i = 0; i < len_s; i++) {
        if (!visited_s[i]) {
            for (int j = 0; j < len_g; j++) {
                if (!visited_g[j] && s[i] == g[j]) {
                    visited_g[j] = 1;
                    c++;
                    break;
                }
            }
        }
    }

    printf("%dA%dB\n", b, c);
    return 0;
}
```

4.Email Id Validation.

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

int main() {
    char mail[100];
    fgets(mail, 100, stdin);
     mail[strcspn(mail, "\n")] = 0;
    int k=1,b1=0,b2=0;

        for(int i=0;i<strlen(mail);i++){
            if(mail[0]=='@'&& mail[strlen(mail)-1]=='@'){
            if(mail[i]=='@'&&k!=0){
                b1=1;
                k--;
            }
            if((mail[i]=='.')){
                b2=1;
            }
        }
        }
    if(b1==1&&b2==1){
        printf("True");
        return 0;
    }
     printf("False");
    return 0;
}
```

6.Count the occurrence of special characters in the given string.

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

```c
#include <ctype.h>

int main() {
    char s[100];
    fgets(s, 100, stdin);
    s[strcspn(s, "\n")] = 0;
    int count;
    char visited[100] = {0};
    for (int i = 0; i < strlen(s); i++) {
        if (!visited[i] && !((isalpha(s[i]))&&!(s[i]==' ')) {
            char c = s[i];
            count = 1;
            visited[i] = 1;
            for (int j = i + 1; j < strlen(s); j++) {
                if (s[j] == c && !visited[j]) {
                    count++;
                    visited[j] = 1;
                }
            }
            printf("%c:%d\n", c, count);
        }
    }
    return 0;
}
```

7.Word Frequency Counter

```c
// Online C compiler to run C program online
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
   char str[100];
   int k=0,count=0;
  fgets(str,100,stdin);
  str[strcspn(str,"\n")] = 0;
```

```
    int i, j = 0;
    for (i = 0; str[i] != '\0'; i++) {
        if (isalnum(str[i]) || isspace(str[i])) {
            str[j++] = tolower(str[i]);  // Convert to lowercase
for case insensitivity
        }
    }
    str[j] = '\0';
  char words[100][100];
  char *word = strtok(str, " ");
  while(word!=NULL){
     strcpy(words[k], word);
      k+=1;
      word = strtok(NULL, " ");
  }
  int visited[100]={0};
  for (int i = 0; i < k; i++) {
      if(visited[i]==0){
          count=1;
      for (int j = i+1; j < k; j++) {
          if(!(strcmp(words[i],words[j]))){
              count+=1;
              visited[j]=1;
           }
         }
         printf("count of \"%s\" : %d \n", words[i],count);
       }
    }
    return 0;
}
```

Extra

2d matrix 90 deg rotation(Transpose of String characters)

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

int main() {
    int m;
    scanf("%d",&m);
    getchar();
    char s[m][m];
    char temp[10]; // temporary buffer to read each row
    // Reading the input
    for (int i = 0; i < m; i++) {
        fgets(temp, sizeof(temp), stdin);
        temp[strcspn(temp, "\n")] = '\0';
        for (int j = 0, k = 0; j < m; j++, k += 2) {
            s[i][j] = temp[k];
        }
    }

    // Printing the array in reverse order
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
           printf("%c ",s[j][i]);
        }
        printf("\n");
    }

    return 0;
}
```

Remove vowels(program to print the given string without any vowel character example: ChatGPT 3.0 @REn_Gan --> ChtGPT 3.0 @ Rn_Gn)

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for (int i = 0; i < strlen(s); i++){

if(s[i]!='a'&&s[i]!='e'&&s[i]!='i'&&s[i]!='o'&&s[i]!='u'&&s[i]!='A
'&&s[i]!='E'&&s[i]!='I'&&s[i]!='O'&&s[i]!='U'){
            printf("%c",s[i]);
        }
    }
    return 0;
}
```

Replacing Words(program to print the given string by replacing specific letter in a string and replace it with provided characters Example : e-->f h-->j numbers --> n hello there 123! --> jfllo tjfrf nnn!)

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for (int i = 0; i < strlen(s); i++){
        if(s[i]=='e'){
            printf("f");
        }
        else if(s[i]=='h'){
           printf("j");
        }
        else if(isdigit(s[i])){
            printf("n");
        }
        else
            printf("%c",s[i]);
    }
    return 0;
}
```

program to print maximum character that is repeated in the string i.e Hello World - l (l 3 repetition)

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int max=0,k;
    for (int i = 0; i < strlen(s); i++){
        int count=0;
        for (int j = 0; j < strlen(s); j++){
            if(s[i]==s[j]&&i!=j){
                count++;
            }
        }
        if(max<count)
            k=i;
    }
    printf("%c",s[k]);
    return 0;
    }
```

program to print the given string without any alphabet character

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for (int i = 0; i < strlen(s); i++){
            if(!isalpha(s[i])){
                printf("%c",s[i]);
        }
    }
    return 0;
    }
```

Hello123sfj4 --> 123+4=127,fjs100fsokf100 -->200

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int  n=0,m=0;
    for (int i = 0; i < strlen(s); i++){
           if(isdigit(s[i])){
               n=(n*10)+(s[i]-'0');
           }
           else {
               m+=n;
               n=0;
           }
    }
    m+=n;
    printf("%d",m);
    return 0;
    }
```

print the string replacing the space with given character -- input : string - "Hello World" replace :- "$''output :''Hello$World"

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for (int i = 0; i < strlen(s); i++){
            if(isspace(s[i])){
                printf("$");
                }
                else{
                    printf("%c",s[i]);
                }
    }
    return 0;
    }
```

count no of parenthesis , semicolons , question marks (that is special characters in a string) - Input = "Hello World ? }" Output = 2

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int count=0;
    for (int i = 0; i < strlen(s); i++){
            if(!isalnum(s[i])&&!isspace(s[i])){
                count++;
                }
    }
    printf("%d",count);
    return 0;
    }
```

Print first non-repeating character of a string

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int count=0;
    for (int i = 0; i < strlen(s); i++){
        count=0;
        for (int j = 0; j < strlen(s); j++){
            if(s[i]==s[j]&&i!=j){
                count=1;
            }
        }

            if(!count){
                printf("%c",s[i]);
                return 0;
            }
    }
    printf("%d",count);
    return 0;
    }
```

String encoding and print the square count(input: star , output: star, 1111, 16)(input: apple, output: apple, 1211,25)

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int count=0;
    printf("%s,",s);
    int visited[100]={0};
    for (int i = 0; i < strlen(s); i++){
        count=0;
        if(!visited[i]){
                for (int j = 0; j < strlen(s); j++){
                    if(s[i]==s[j]){
                        count++;
```

```
                            visited[j]=1;
                     }
                }
            }
     if(count>0)
     printf("%d",count);
}
printf(",%d",strlen(s)*strlen(s));
return 0;
}
```

Extract numbers from string Input: 20 apples and 5 oranges Output: 20 5 (print numbers in newline)

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int  n=0,m=0;
    for (int i = 0; i < strlen(s); i++){
           if(isdigit(s[i])){
               n=(n*10)+(s[i]-'0');
           }
           else if(n>0){
               printf("%d ",n);
               n=0;
           }
    }
    if(n>0)
    printf("%d",n);
    return 0;
    }
```

print the odd position words and concatenate them. Sample Test case: Input : 3 apple orange mango (words will be in separate lines one below other 👇🏼). Output :applemango

```
#include <stdio.h>
#include <string.h>
int main() {
    int numWords;
    printf("Enter the number of words: ");
    scanf("%d", &numWords);
    char words[numWords][100];
    printf("Enter the words:\n");
    for (int i = 0; i < numWords; i++) {
        scanf("%s", words[i]);
    }
    printf("Words at even positions:\n");
    for (int i = 0; i < numWords; i ++) {
        if((i+1)%2!=0)
        printf("%s", words[i]);
    }
    return 0;
}
```

remove duplicate characters in a string EG: INPUT: Hello world OUTPUT: Helo wrd

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

```c
#include<math.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int visited[100]={0};
    for (int i = 0; i < strlen(s); i++){
        if(!visited[i]){
                for (int j = i+1; j < strlen(s); j++){
                    if(s[i]==s[j]){
                        visited[j]=1;
                 }
            }
        }
    }
    for (int i = 0; i < strlen(s); i++){
        if(!visited[i])
            printf("%c",s[i]);
    }
    return 0;
}
```

remove duplicate words in a string(not a case sensitive) INPUT : Sucess sucess is a problem OUTPUT: Sucess is a problem

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
int main() {
    char s1[100];
    fgets(s1, sizeof(s1), stdin);
    s1[strcspn(s1, "\n")] = '\0';
    char words[100][100];
    int visited[100] = {0};
    int k = 0, m = 0;
    char s[200];
    for (int i = 0; i<strlen(s1); i++) {
        s[i] = tolower((unsigned char)s1[i]);
    }
    s[strlen(s1)] = '\0';
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    for (int i = 0; i < k; i++){
        if(!visited[i]){
                for (int j = i+1; j < k; j++){
                    if(!strcmp(words[i],words[j])){
                        visited[j]=1;
                 }
            }
        }
    }
   int firstWord = 1;
    for (int i = 0; i < k; i++) {
        if (!visited[i]) {
            if (!firstWord) {
                printf(" ");
            } else {
                // Capitalize the first letter of the first word
                words[i][0] = toupper(words[i][0]);
            }
            printf("%s", words[i]);
            firstWord = 0;
        }
    }
    return 0;
}
```

Delete target word in a string

```c
#include <stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    char words[100][100];
    int visited[100] = {0};
    int k=0;
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    for (int i = 0; i < k; i++){
                if(i+1==1){
                    visited[i]=1;
        }
    }
    int m=0;
    for (int i = 0; i < k; i++) {
        if (!visited[i]) {
            if(m>0){
                printf(" ");
            }
            printf("%s", words[i]);
            m++;
        }
    }
    return 0;
}
```

REVERSE THE WORD WHICH ARE PRESENT IN THE EVEN POSITION INPUT: 5 APPLE MANGO

ORANGE BANANA GRAPES OUTPUT: OGNAM ANANAB

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    char words[100][100];
    int visited[100] = {0};
    int k=0;
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    for (int i = 0; i < k; i++){
                if(i%2==0){
                    visited[i]=1;
             }
        }
    for (int i = 0; i < k; i++) {
        if (!visited[i]) {
           for (int j = strlen(words[i])-1; j>=0; j--){
            printf("%c", words[i][j]);
           }
           printf("\n");
        }
    }
    return 0;
}
```

Convert unformat string into Title case.

```
#include <stdio.h>
#include <ctype.h>
int main() {
    char str[100];
    fgets(str, 100, stdin);
    int capitalize = 1;
    for (int i = 0; str[i] != '\0'; i++) {
        if (isspace(str[i])) {
            capitalize = 1;
        } else if (capitalize && isalpha(str[i])) {
            str[i] = toupper(str[i]);
            capitalize = 0;
        } else {
            str[i] = tolower(str[i]);
        }
    }
    printf("%s", str);
    return 0;
}
```

Sum of digits in a string and reverse the sum .if i/p->"djs1245d" , o/p->21.if i/p->"djs55d" , o/p->01

```
#include <stdio.h>
#include<string.h>
#include<ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int sum=0;
    for(int i=0;i<strlen(s);i++){
        if(isdigit(s[i])){
            sum+=(s[i]-'0');
        }
    }
    char sumStr[20];
    sprintf(sumStr, "%d", sum);
    int len = strlen(sumStr);
    for (int i = len - 1; i >= 0; i--) {
        printf("%c",sumStr[i]);
    }

    return 0;
}
```

Railway time if valid print it. if incorrect format print "Invalid"

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char s[9];
    int hours, minutes, seconds;
    scanf("%8s", s);
    if (strlen(s) != 8 || s[2] != ':' || s[5] != ':' ||
        !isdigit(s[0]) || !isdigit(s[1]) || !isdigit(s[3]) ||
!isdigit(s[4]) || !isdigit(s[6]) || !isdigit(s[7])) {
        printf("Invalid\n");
        return 0;
    }
    sscanf(s, "%d:%d:%d", &hours, &minutes, &seconds);
    if (hours < 0 || hours > 23 || minutes < 0 || minutes > 59 ||
seconds < 0 || seconds > 59) {
            printf("Invalid\n");
        } else {
            printf("Hour:%d Minutes:%d
seconds:%d",hours,minutes,seconds);
        }

    return 0;
}
```

Found the all the Index of a target word in a i/p string

Enter the first string: School is school

Enter the word to search for: school

Found at index: 0

Found at index: 10

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

int main() {
    char arr[100], word[100];
    fgets(arr,100,stdin);
    fgets(word,100,stdin);
    arr[strlen(arr)-1]='\0';
    word[strlen(word)-1]='\0';
    int c=0;
    for(int i=0; i<sizeof(arr); i++){
        if(arr[i]==' '){
            c++;
         }
     }
    char *t = strtok(arr," ");
    int i=0;
    while(t!=NULL ){
         if(strcmp(t,word)==0){
            printf("word found at index: %d\n",i);
        }
        i+=strlen(t)+1;
        t = strtok(NULL, " ");
    }
    return 0;
}
```

Swap the Adjacent char of words .i/p->Open Ai , o/p->pOne iA

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

int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for (int i = 0; i < strlen(s) - 1; i++) {
        if (!isspace(s[i]) && !isspace(s[i + 1])) {
            char temp = s[i];
            s[i] = s[i + 1];
            s[i + 1] = temp;
            i++;
        }
    }
    printf("%s", s);
    return 0;
}
```

Get a integer as input. Get that much strings Each of 15 Characters.

**(i) 1st 10 characters represent phone number
(ii) Next Character represent Gender
(iii) Next 2 Chai represent age

(iv) Remaining represent seat no.

Find the count whose age is greats than 60.

Sample i/p:

3

9034567890M62H5

9034563767M4245

9034563767M6245

Output: 2

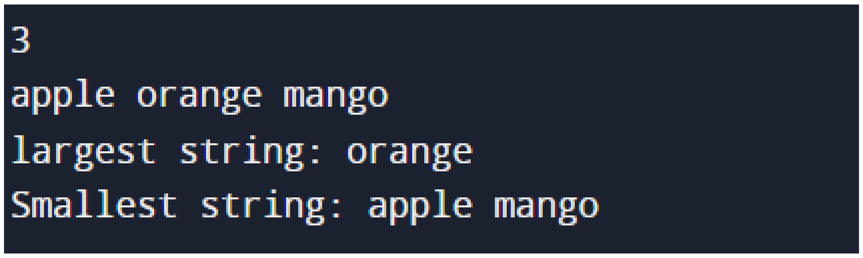
```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    int n, count = 0;
    scanf("%d", &n);
    getchar();
    char strings[n][100];
    for (int i = 0; i < n; i++) {
        fgets(strings[i], sizeof(strings[i]), stdin);
        strings[i][strcspn(strings[i], "\n")] = '\0';
    }
    for (int i = 0; i < n; i++) {
            int m = (strings[i][11] - '0') * 10 + (strings[i][12]
- '0');
            if (m > 60) {
                count++;
            }
    }
    printf("%d\n", count);
    return 0;
}
```

2 largest string size in a 2d array

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

```c
#include <ctype.h>
int main() {
    int n,m1=0,m2=0;
    printf("Enter the number of words: ");
    scanf("%d", &n);
    char words[n][100];
    printf("Enter the words:\n");
    for (int i = 0; i < n; i++) {
        scanf("%s", words[i]);
    }
   for(int i=0;i<n;i++){
                if(m1<strlen(words[i])){
                        m2=m1;
                       m1=strlen(words[i]);
                   }
                   else
if(m1>strlen(words[i])&&m2<strlen(words[i])){
                       m2=strlen(words[i]);
                   }
    }
    printf("%d",m2);
    return 0;
}
```

Largest and smallest in an 2d array



```
3
apple orange mango
largest string: orange
Smallest string: apple mango
```

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    int n, m2=1000000,m1=0;
    scanf("%d", &n);
    getchar();
    char s[100],s1[100][100],s2[100][100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    char words[100][100];
    int k=0;
    char *word = strtok(s, " ");
    while (word != NULL) {
        strcpy(words[k++], word);
        word = strtok(NULL, " ");
    }
    int a,b;
   for(int i=0;i<n;i++){
                if(m1<strlen(words[i])){
                      m1=strlen(words[i]);
                      a=0;
                      strcpy(s1[a++],words[i]);
                  }
                  else if(strlen(words[i])==m1){
                      strcpy(s1[a++],words[i]);
                  }
                if(m2>strlen(words[i])){
                      m2=strlen(words[i]);
                      b=0;
                      strcpy(s2[b++],words[i]);
                  }
                  else if(strlen(words[i])==m2){
                      strcpy(s2[b++],words[i]);
                  }

    }
    printf("largest string: ");
     for(int i=0;i<a;i++){
         if(i>0){
             printf(" ");
         }
         printf("%s",s1[i]);
     }
     printf("\n");
    printf("Smallest string: ");
   for(int i=0;i<b;i++){
        if(i>0){
             printf(" ");
         }
         printf("%s",s2[i]);
     }
    return 0;
}
```

Pangram

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char s[100];
    char a[256]={0};
    int count=0;
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    for(int i=0;i<strlen(s);i++){
        s[i]=tolower(s[i]);
    }
    for(int i=0;i<strlen(s);i++){
       if(isalpha(s[i])){
            a[(int)s[i]]=1;
       }
    }
    for(int i=97;i<123;i++){
       if(!a[i]){
            printf("Not Pangram");
            return 0;
       }
    }
    printf("Pangram");
```

```c
    return 0;
}
```

Left rotation

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int count;
    scanf("%d",&count);
    for(int i=0;i<count;i++){
        char temp=s[0];
        for(int j=0;j<strlen(s)-1;j++){
            s[j]=s[j+1];
        }
        s[strlen(s)-1]=temp;
    }
        printf("%s",s);
    return 0;
}
```

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

int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0'; // Remove the newline character

    int count;
    scanf("%d", &count);
    int len = strlen(s);

    // Ensure count is within the bounds of the string length
    count = count % len;

    // Temporary buffer to hold the rotated part
    char temp[count];

    // Copy the first 'count' characters to the temp array
    strncpy(temp, s, count);

    // Shift the remaining part of the string to the beginning
    memmove(s, s + count, len - count);

    // Append the temp array to the end
    strncpy(s + len - count, temp, count);

    printf("%s", s);
    return 0;
}
```

Right Rotation

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0';
    int count;
    scanf("%d",&count);
    for(int i=0;i<count;i++){
        char temp=s[strlen(s)-1];
        for(int j=strlen(s)-1;j>0;j--){
            s[j]=s[j-1];
```

```c
        }
        s[0]=temp;
    }
        printf("%s",s);
    return 0;
}
```

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

int main() {
    char s[100];
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0'; // Remove the newline character

    int count;
    scanf("%d", &count);

    int len = strlen(s);

    // Ensure count is within the bounds of the string length
    count = count % len;

    // Temporary buffer to hold the rotated part
    char temp[count];

    strncpy(temp, s + len - count, count);

    // Shift the remaining part of the string to the right
    memmove(s + count, s, len - count);

    // Prepend the temp array to the beginning
    strncpy(s, temp, count);

    printf("%s", s);
    return 0;
}
```