Context:

[**1)Anagram 1**](#_ohg1flj5qxrh)

[**2)Password checker 2**](#_bp1m1r4bxkbm)

[**3.Reverse an string: 4**](#_1ycb6uur04r3)

[**4.Parenthesis Question: 4**](#_9jn7v7wpx0wg)

[**5.AAABBBCCC PALINDROME check 6**](#_gmptz5fo2s50)

[**6.Case manipulation: 8**](#_yzap6k3dyz9a)

[**7.FLAMES 9**](#_ygpu0q7uqnp0)

[**8.rotating string: 12**](#_flsmp3bmemli)

[**9.Sniping and replacing 13**](#_rqah04rzkyg7)

[**10.Isomorphic strings 13**](#_ygax5sub5gkc)

[**11.Panagram 15**](#_72hazvfsvnzd)

[**12. ROMAN TO INTEGER 16**](#_2mjqw2kwxr37)

[**13.Longest palindromic sequence 17**](#_ye32aowzadfh)

[**14.String encoding: 19**](#_xh0t5kqtx8eb)

[**15.Minimum occurence 20**](#_d8f0r6loouju)

[**16. Reverse the word at even pos: 21**](#_lkyxbxdv3t3k)

[**17.Adding digits in an string: 21**](#_6pfzt1azlsgd)

[**18.Capitalizing the first letter of each word: 22**](#_rvuq2ox1dumd)

[**19.concatenate the odd pos words: 23**](#_j25nkntzzq9x)

[**20.Remove vowels: 24**](#_2fno8dyrmyb0)

[**21.Fill the gap with %10: 24**](#_rh0e1jmez5ex)

[**22.Extracing a no of digits 25**](#_t2m47vm8c79l)

[**23.Changing the char for our requirements: 26**](#_8jd10u6jyhiq)

[**#include <stdio.h> 26**](#_56hvp5w8klwv)

[**24.Character matrix rotation by 90degree. 27**](#_xexreu5ctfef)

[**25.Finding max and min occurrence of alphabets: 28**](#_mh9iuhgyibq0)

[**26.Sum up all the digit in an string and rev it and print 29**](#_xelnm0wz8yq8)

[**27.Validate the time. 30**](#_cxqmegc00pm2)

[**28.Removing the target word: 31**](#_1mzbzkjwk7kk)

[**29.Getting input of words and printing the size of the largest word 31**](#_lgpwko3qguh4)

[**30.finding dupli min dist: 32**](#_7kwf38dna51d)

[**31.Extracting data from string. 33**](#_4ehqghimlb30)

[**32.Finding longest and smallest strings 34**](#_ue9kwsbl6hi3)

# 1)Anagram

i)

#include <stdio.h>

#include<string.h>

#include <ctype.h>

int main() {

char s[100];

fgets(s,sizeof(s),stdin);

s[strcspn(s,"\n")]='\0';

char a[100];

fgets(a,sizeof(a),stdin);

a[strcspn(a,"\n")]='\0';

int l1=strlen(s);

int l2=strlen(a);

for(int i=0;i<l1;i++){

for(int j=0;j<l1-i-1;j++){

if(s[j]>s[j+1]){

char temp=s[j];

s[j]=s[j+1];

s[j+1]=temp;

}

}

}

for(int i=0;i<l2;i++){

for(int j=0;j<l2-i-1;j++){

if(a[j]>a[j+1]){

char temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

if(strcmp(s,a)==0){

printf("Anagram");

}else{

printf("Non-Anagram");

}

}

# 2)Password checker

i

#include<stdio.h>

#include<string.h>

#include<ctype.h>

int isPalindrome(char str[], int n) {

int i = 0, j = n - 1;

while (i < j) {

if (!isalnum(str[i])) {

i++;

} else if (!isalnum(str[j])) {

j--;

} else if (tolower(str[i]) != tolower(str[j])) {

return 0;

} else {

i++;

j--;

}

}

return 1;

}

int main() {

char P[10000007];

fgets(P, sizeof(P), stdin);

int n = strlen(P);

P[strcspn(P, "\n")] = '\0';

if (isPalindrome(P, n)) {

printf("Password is Vulnerable\n");

} else {

printf("Password is Non-Vulnerable\n");

}

return 0;

}

ii

#include <stdio.h>

#include <string.h>

int main() {

char pass[100];

char rev[100];

fgets(pass, sizeof(pass), stdin);

pass[strcspn(pass, "\n")] = '\0';

strcpy(rev, pass);

int length = strlen(pass);

for (int i = 0; i < length / 2; i++) {

char temp = pass[i];

pass[i] = pass[length - i - 1];

pass[length - i - 1] = temp;

}

if (strcmp(rev, pass) == 0) {

printf("Password is Vulnerable\n");

} else {

printf("Password is Non-Vulnerable\n");

}

return 0;

}

# 3.Reverse an string:

#include<stdio.h>

#include<string.h>

int main(){

char s[100];

char re[100];

fgets(s,sizeof(s),stdin);

s[strcspn(s,"\n")]='\0';

strcpy(re,s);

int len=strlen(s);

for(int i=0;i<len/2;i++){

int temp=re[i];

re[i]=re[len-i-1];

re[len-i-1]=temp;

}

fputs(re,stdout);

}

# 4.Parenthesis Question:

#include <stdio.h>

#include <string.h>

int main() {

char str[100];

printf("ENTER THE STRING:");

fgets(str, sizeof(str), stdin);

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

int l = strlen(str);

int c = 0;

for (int i = 0; i < l - 1; i++) {

if (str[i] == '(' && str[i + 1] == ')') {

c++;

}

}

if (c > 0) {

printf("TRUE\n");

} else {

printf("FALSE\n");

}

return 0;

}

ii)

#include<stdio.h>

#include<string.h>

int isvalid(char \*str){

int curly=0,square=0,round=0;

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

int current = str[i];

if(current = '('){

round++;

}

else if(current = '{'){

curly++;

}

else if(current = '['){

square++;

}

else if(current = ')'){

round--;

if(round < 0) return 0;

}

else if(current = '}'){

curly--;

if(curly < 0) return 0;

}

else if(current = ']'){

square--;

if(square < 0) return 0;

}

}

return (round == 0 && curly == 0 && square == 0);

}

int main(){

char str[100];

printf("Enter string of paranthesises : ");

fgets(str, sizeof(str), stdin);

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

if(isvalid(str)){

// printf("Valid string of parentheses\n");

printf("True");

}

else{

// printf("Invalid string of parentheses\n");

printf("False");

}

}

# **5.AAABBBCCC PALINDROME check**

#include <stdio.h>

#include <string.h>

int main() {

char s[100];

printf("ENTER THE STRING: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

// Check for invalid characters

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

if (s[i] != 'a' && s[i] != 'b' && s[i] != 'c') {

printf("INVALID INPUT\n");

return 0;

}

}

// Handle empty string

if (strlen(s) == 0) {

printf("0\n");

return 0;

}

int left = 0, right = strlen(s) - 1;

// Remove prefixes and suffixes

while (left < right) {

// Find matching characters

while (left < right && s[left] == s[left + 1]) left++;

while (left < right && s[right] == s[right - 1]) right--;

if (s[left] == s[right]) {

left++;

right--;

} else {

break; // Exit if they don't match

}

}

// Print the length of the remaining string

printf("%d\n", right - left + 1);

return 0;

}

# 6.Case manipulation:

#include <stdio.h>

#include <ctype.h>

#include <string.h>

int main() {

char s[100];

int choice;

printf("Enter a string: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

printf("Choose an option: \n");

printf("1. Convert to CamelCase\n");

printf("2. Convert to lowercase\n");

printf("3. Convert to UPPERCASE\n");

scanf("%d", &choice);

switch (choice) {

case 1: {

int capNext = 1; // Capitalize the first character

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

if (s[i] == ' ') {

capNext = 1;

} else if (capNext) {

s[i] = toupper(s[i]);

capNext = 0;

} else {

s[i] = tolower(s[i]);

}

}

printf("Converted to CamelCase: %s\n", s);

break;

}

case 2:

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

s[i] = tolower(s[i]);

}

printf("Converted to lowercase: %s\n", s);

break;

case 3:

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

s[i] = toupper(s[i]);

}

printf("Converted to UPPERCASE: %s\n", s);

break;

default:

printf("Invalid option! Please choose between 1 and 3.\n");

}

return 0;

}

# 7.FLAMES

#include <stdio.h>

#include <string.h>

#include <ctype.h>

void removeCommonCharacters(char \*s1, char \*s2) {

int count[26] = {0};

for (int i = 0; s1[i] != '\0'; i++) {

if (isalpha(s1[i])) count[tolower(s1[i]) - 97]++;

}

for (int i = 0; s2[i] != '\0'; i++) {

if (isalpha(s2[i])) count[tolower(s2[i]) - 97]--;

}

int j = 0;

for (int i = 0; s1[i] != '\0'; i++) {

if (isalpha(s1[i]) && count[tolower(s1[i]) - 97] != 0) {

s1[j++] = s1[i];

}

}

s1[j] = '\0';

k = 0;

for (int i = 0; s2[i] != '\0'; i++) {

if (isalpha(s2[i]) && count[tolower(s2[i]) - 97] != 0) {

s2[k++] = s2[i];

}

}

s2[k] = '\0';

}

char flamesGame(int count) {

char str[7] = "FLAMES";

int l = strlen(str);

int pos = 0;

while (l > 1) {

pos = (pos + count - 1) % l;

for (int i = pos; i < l - 1; i++) {

str[i] = str[i + 1];

}

l--;

}

return str[0];

}

int main() {

char s1[101], s2[101];

printf("Enter the first name: ");

fgets(s1, sizeof(s1), stdin);

s1[strcspn(s1, "\n")] = '\0';

printf("Enter the second name: ");

fgets(s2, sizeof(s2), stdin);

s2[strcspn(s2, "\n")] = '\0';

if (strlen(s1) < 2 || strlen(s2) < 2 || strcmp(s1, s2) == 0) {

printf("Please enter two different names of 2 to 100 characters long\n");

return 0;

}

removeCommonCharacters(s1, s2);

int count = strlen(s1) + strlen(s2);

if (count == 0) {

printf("Result = 0 (no remaining characters)\n");

} else {

char result = flamesGame(count);

printf("Result = '%c'\n", result);

}

return 0;

}

ii)

#include<stdio.h>

#include<string.h>

#include<ctype.h>

char flamesResult(int count) {

char str[]="FLAMES";

int pos=0;

int len=strlen(str);

while(len>1){

pos=(pos+count-1)%len;

for(int j=pos;j<len-1;j++){

str[j]=str[j+1];

}

str[len-1]='\0';

len--;

}

return str[0];

}

int Count(char \*str1, char \*str2){

char s1[101], s2[101];

int count=0;

strcpy(s1,str1);

strcpy(s2,str2);

// Convert everything to lowercase and remove spaces

int index1 = 0, index2 = 0;

for(int i=0;s1[i]!='\0';i++){

if(s1[i]!=' '){

s1[index1++]=tolower(s1[i]);

}

}

s1[index1]='\0';

for(int i=0;s2[i]!='\0';i++){

if(s2[i]!=' '){

s2[index2++]=tolower(s2[i]);

}

}

s2[index2]='\0';

for(int i=0;s1[i]!='\0';i++){

for(int j=0;s2[j]!='\0';j++){

if(s1[i]==s2[j]){

s1[i]=s2[j]='\*';

}

}

}

for(int i=0;s1[i]!=0;i++){

if(s1[i]!='\*'){

count++;

}

}

for(int i=0;s2[i]!=0;i++){

if(s2[i]!='\*'){

count++;

}

}

return count;

}

int main(){

char str1[101], str2[101];

printf("Enter name 1 : ");

fgets(str1, sizeof(str1), stdin);

str1[strcspn(str1, "\n")] = '\0';

printf("Enter name 2 : ");

fgets(str2, sizeof(str2), stdin);

str2[strcspn(str2, "\n")] = '\0';

// Check for invalid input

if(strlen(str1) < 2 || strlen(str2) < 2 || strlen(str1) > 100 || strlen(str2) > 100){

printf("Please enter two different names of 2 to 100 characters long");

return 0;

}

// Calculate the count of non-common characters

int count = Count(str1, str2);

// Get the FLAMES result based on the count

char result = flamesResult(count);

printf("Result: %c\n", result);

return 0;

}

# 8.rotating string:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char s[100];

printf("Enter a string: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

int rot;

printf("Enter number of positions to rotate: ");

scanf("%d", &rot);

int len = strlen(s);

rot = rot % len;

char rotated[100];

for (int i = 0; i < len; i++) {

rotated[i] = s[(i + len - rot) % len];

}

rotated[len] = '\0';

printf("Rotated string: %s\n", rotated);

return 0;

}

# 9.Sniping and replacing

#include <stdio.h>

#include <string.h>

int main() {

char str[100];

char target, replacement;

printf("Enter a string: ");

fgets(str, sizeof(str), stdin);

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

printf("Enter the character to replace: ");

scanf(" %c", &target);

printf("Enter the replacement character: ");

scanf(" %c", &replacement);

int len = strlen(str);

for (int i = len - 1; i >= 0; i--) {

if (str[i] == target) {

str[i] = replacement;

break;

}

}

printf("Modified string: %s\n", str);

return 0;

}

# 10.Isomorphic strings

#include <stdio.h>

#include <ctype.h>

#include<string.h>

#include<stdbool.h>

#define max 256

bool isomorphic(char \*s,char \*a){

int map1[max]={0};

int map2[max]={0};

if(strlen(s)!=strlen(a)){

return false;

}

for(int i=0;i<strlen(s);i++){

int char1=(int)s[i];

int char2=(int)a[i];

if(map1[char1]==0 && map2[char2]==0){

map1[char1]=char2;

map2[char2]=char1;

}else if(map1[char1]!=char2|| map2[char2]!=char1){

return false;

}

}

return true;

}

int main() {

char s[100];

char a[100];

fgets(s,sizeof(s),stdin);

fgets(a,sizeof(a),stdin);

s[strcspn(s,"\n")]='\0';

a[strcspn(a,"\n")]='\0';

if(isomorphic(s,a)){

printf("Isomorphic");

}else{

printf("Non-Isomorphic");

}

}

# 11.Panagram

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char s[200];

int alphabet[26] = {0};

int uniqueCount = 0;

printf("Enter a sentence: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

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

int ch=tolower(s[i]);

if(ch>='a'&& ch<='z'){

int index=ch-'a';

if(alphabet[index]==0){

alphabet[index]=1;

uniqueCount++;

}

}

}

if (uniqueCount == 26) {

printf("Pangram\n");

} else {

printf("Not a Pangram\n");

}

return 0;

}

# 12. ROMAN TO INTEGER

#include<stdio.h>

#include<string.h>

#include<stdbool.h>

bool validRoman(char \*str){

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

if(str[i]!='I' && str[i]!='V' && str[i]!='X' && str[i]!='L' && str[i]!='C' && str[i]!='D' && str[i]!='M'){

return false;

}

}

return true;

}

int romanToInt(char \*str){

int result=0;

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

switch(str[i]){

case 'I': result+=1; break;

case 'V': result+=5; break;

case 'X': result+=10; break;

case 'L': result+=50; break;

case 'C': result+=100; break;

case 'D': result+=500; break;

case 'M': result+=1000; break;

default: result+=0;

}

if((str[i] == 'V' || str[i] == 'X') && str[i-1] == 'I'){

result-=2;

}

else if((str[i] == 'L' || str[i] == 'C') && str[i-1] == 'X'){

result-=20;

}

else if((str[i] == 'D' || str[i] == 'M') && str[i-1] == 'C'){

result-=200;

}

}

return result;

}

int main(){

char str[100];

printf("Enter ROMAN value : ");

fgets(str,sizeof(str),stdin);

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

if(!validRoman(str)){

printf("Invalid input");

return 0;

}

int result = romanToInt(str);

printf("The integer value for %s is %d ",str,result);

return 0;

}

# 13.Longest palindromic sequence

#include <stdio.h>

#include <string.h>

#define MAX 1000

void longestPalindromicSubstring(char \*s) {

int len = strlen(s);

if (len == 0) {

printf("Longest palindromic substring: \n");

return;

}

int dp[MAX][MAX] = {0};

int start = 0;

int maxLength = 1;

for (int i = 0; i < len; i++) {

dp[i][i] = 1;

}

for (int i = 0; i < len - 1; i++) {

if (s[i] == s[i + 1]) {

dp[i][i + 1] = 1;

start = i;

maxLength = 2;

}

}

for (int length = 3; length <= len; length++) {

for (int i = 0; i < len - length + 1; i++) {

int j = i + length - 1;

// Check if the characters at the ends match and if the inner substring is a palindrome

if (s[i] == s[j] && dp[i + 1][j - 1]) {

dp[i][j] = 1;

start = i;

maxLength = length;

}

}

}

// Output the longest palindromic substring

printf("Longest palindromic substring: ");

for (int i = start; i < start + maxLength; i++) {

putchar(s[i]);

}

printf("\n");

}

int main() {

char s[MAX];

printf("Enter a string: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0'; // Remove the trailing newline character

longestPalindromicSubstring(s);

return 0;

}

# 14.String encoding:

#include <stdio.h>

#include<string.h>

#include <ctype.h>

int main() {

char s[100];

fgets(s,sizeof(s),stdin);

char freq[256]={0};

s[strcspn(s,"\n")]='\0';

int sumosq=0;

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

freq[(int)s[i]]++;

}

for(int i=0;i<256;i++){

if(freq[i]>0){

sumosq+=freq[i]\*freq[i];

}

}

for(int i=0;s[i]!=0;i++){

if(freq[(int)s[i]]>0){

printf("%d",freq[(int)s[i]]);

freq[(int)s[i]]=0;

}

}

printf("\n%d",sumosq);

}

# 15.Minimum occurence

#include <stdio.h>

#include <string.h>

int main() {

char s[100];

int freq[256] = {0};

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

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

freq[(int)s[i]]++;

}

printf("Output: ");

int count = 0;

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

if (freq[(int)s[i]] == 1) {

printf("%c, ", s[i]);

count++;

}

}

if (count > 0) {

printf("\b\b ");

}

printf("\nCount: %d\n", count);

return 0;

}

# 

# 16. Reverse the word at even pos:

#include <stdio.h>

#include <string.h>

void reverseWord(char\* word) {

int len=strlen(word);

for(int i=0;i<len/2;i++){

char temp=word[i];

word[i]=word[len-i-1];

word[len-i-1]=temp;

}

}

int main() {

int n;

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 += 2) {

reverseWord(words[i]);

}

printf("\nReversed words at even positions:\n");

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

printf("%s\n", words[i]);

}

return 0;

}

# 17.Adding digits in an string:

#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;s[i]!='\0';i++){

if(isdigit(s[i])){

int num=s[i]-'0';

sum+=num;

}

}

printf("%d",sum);

}

# 18.Capitalizing the first letter of each word:

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

char s[1000];

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0'; // Remove newline character

if (strlen(s) == 0) {

printf("Empty input!\n");

return 0;

}

// Capitalize the first character if it's a letter

if (isalpha(s[0])) {

s[0] = toupper(s[0]);

}

// Traverse the string and capitalize the first letter of each word

for (int i = 1; s[i] != '\0'; i++) {

if (s[i] == ' ' && isalpha(s[i+1])) {

// Capitalize the letter after a space

s[i+1] = toupper(s[i+1]);

}

}

fputs(s, stdout); // Output the modified string

return 0;

}

# 19.concatenate the odd pos words:

#include <stdio.h>

#include <string.h>

int main() {

int n;

printf("Enter the number of words: ");

scanf("%d", &n);

char words[n][100];

printf("Enter %d words (one per line):\n", n);

for(int i = 0; i < n; i++) {

fgets(words[i], sizeof(words[i]), stdin);

// Remove the newline character if present

words[i][strcspn(words[i], "\n")] = '\0';

}

char result[1000] = ""; // To store the concatenated result

// Concatenate words at odd positions (1, 3, 5, ...)

for(int i = 0; i < n; i++) {

if (i % 2 == 0) { // 0-based index: odd position means even index

strcat(result, words[i]);

}

}

// Print the concatenated result

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

return 0;

}

# 20.Remove vowels:

#include <stdio.h>

#include <string.h>

int main() {

char s[100];

// Input string

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0'; // Remove newline character from the input

int j = 0;

// Iterate through the string

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

// Check if the current character is not a vowel

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') {

s[j++] = s[i]; // Copy non-vowel characters

}

}

s[j] = '\0'; // Terminate the modified string

// Output the result

fputs(s, stdout);

return 0;

}

# 21.Fill the gap with %10:

#include <stdio.h>

#include <string.h>

int main() {

char s[100],res[1000];

int freq[256] = {0};

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

int j=0;

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

if(s[i]==' '){

res[j++]='%';

res[j++]='1';

res[j++]='0';

}else{

res[j++]=s[i];

}

}

res[j]='\0';

fputs(res,stdout);

return 0;

}

# 22.Extracing a no of digits

#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 num=0;

int found=0;

for(int i=0;s[i]!=0;i++){

if(isdigit(s[i])){

num=num\*10+(s[i]-'0');

found=1;

}else{

if(found){

printf("%d\n",num);

found=0;

num=0;

}

}

}

if(found){

printf("%d\n",num);

found=0;

}

return 0;

}

# 23.Changing the char for our requirements:

# #include <stdio.h>

#include<string.h>

#include <ctype.h>

int main() {

char s[100];

fgets(s,sizeof(s),stdin);

s[strcspn(s,"\n")]='\0';

char res[250];

int j=0;

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

if(s[i]=='a'){

res[j++]='b';

}else if(s[i]=='e'){

res[j++]='f';

}else if(isdigit(s[i])){

res[j++]='d';

}else{

res[j++]=s[i];

}

}

res[j]='\0';

fputs(res,stdout);

}

# 24.Character matrix rotation by 90degree.

# 

#include <stdio.h>

#include <string.h>

int main() {

int r, c;

// Input for number of rows and columns

printf("Row: ");

scanf("%d", &r);

printf("Column: ");

scanf("%d", &c);

getchar(); // To consume the newline left after entering rows and columns

char mat[r][c];

printf("Enter the elements row by row:\n");

// Input matrix elements (character by character)

for (int i = 0; i < r; i++) {

for (int j = 0; j < c; j++) {

scanf(" %c", &mat[i][j]); // Notice the space before %c to consume any newline/space

}

}

// Print the 90-degree rotated matrix (clockwise)

printf("90 Degree Rotated Matrix:\n");

for (int j = 0; j < c; j++) {

for (int i = r - 1; i >= 0; i--) {

printf("%c ", mat[i][j]);

}

printf("\n");

}

return 0;

}

# 25.Finding max and min occurrence of alphabets:

#include <stdio.h>

#include <string.h>

#include<ctype.h>

int main() {

char s[100];

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

char freq[256] = {0};

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

if (s[i] != ' ' && isalpha(s[i])) {

freq[(int)s[i]]++;

}

}

int max = 0;

char maxchar = '\0';

int min = 256;

char minchar = '\0';

for (int i = 0; i < 256; i++) {

if (freq[i] > max) {

max = freq[i];

maxchar = i;

}

if (freq[i] > 0 && freq[i] < min) {

min = freq[i];

minchar = i;

}

}

if (max > 0) {

printf("The character '%c' is repeated %d times\n", maxchar, max);

} else {

printf("No characters found\n");

}

if (min < 256) {

printf("The character '%c' is repeated %d times\n", minchar, min);

} else {

printf("No characters found\n");

}

return 0;

}

# 26.Sum up all the digit in an string and rev it and print

#include <stdio.h>

#include <string.h>

#include <ctype.h>

// Function to reverse a string

void strrev(char\* str) {

int n = strlen(str);

for (int i = 0; i < n / 2; i++) {

char temp = str[i];

str[i] = str[n - i - 1];

str[n - i - 1] = temp;

}

}

int main() {

char s[100];

printf("Enter a string: ");

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0';

int sum = 0;

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

if (isdigit(s[i])) {

sum += s[i] - '0';

}

}

char numstr[20];

sprintf(numstr, "%d", sum);

// Reverse the number string

strrev(numstr);

// Print the reversed number string

printf("Sum of digits: %d\n", sum);

printf("Reversed sum as a string: %s\n", numstr);

return 0;

}

# 27.Validate the time.

#include <stdio.h>

#include <string.h>

int main() {

char timetf[9];

int hr, min, sec;

int valid\_format;

fgets(timetf, sizeof(timetf), stdin);

timetf[strcspn(timetf, "\n")] = '\0';

valid\_format = sscanf(timetf, "%d:%d:%d", &hr, &min, &sec) == 3;

if (valid\_format && (hr >= 0 && hr <= 23) && (min >= 0 && min <= 59) && (sec >= 0 && sec <= 59)) {

printf("Hour:%02d Minutes:%02d Seconds:%02d\n", hr, min, sec);

} else {

printf("Invalid time format or values\n");

}

return 0;

}

# 28.Removing the target word:

#include<stdio.h>

#include<string.h>

void deleteTarget(char \*str,const char \*target){

int targetlen=strlen(target);

char \*pos;

while((pos = strstr(str,target)) != NULL){

memmove(pos, pos + targetlen, strlen(pos + targetlen)+1);

}

}

int main(){

char str[1000],target[1000];

printf("Enter a string : ");

fgets(str,sizeof(str),stdin);

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

printf("Enter the target string : ");

fgets(target,sizeof(target),stdin);

target[strcspn(target,"\n")]='\0';

deleteTarget(str,target);

printf("The string after target deletion : %s",str);

return 0;

}

# 29.Getting input of words and printing the size of the largest word

#include <stdio.h>

#include <string.h>

#include <ctype.h>

int main() {

int n;

scanf("%d", &n);

getchar();

char word[n][100];

for (int i = 0; i < n; i++) {

fgets(word[i], sizeof(word[i]), stdin);

word[i][strcspn(word[i], "\n")] = '\0';

}

int max = 0;

int size = 0;

int allSameLength = 1;

size = strlen(word[0]);

max = size;

for (int i = 1; i < n; i++) {

size = strlen(word[i]);

if (size > max) {

max = size;

}

if (size != strlen(word[0])) {

allSameLength = 0;

}

}

if (allSameLength) {

printf("0");

} else {

printf("%d",max);

}

return 0;

}

# 30.finding dupli min dist:

#include <stdio.h>

#include <string.h>

int main() {

char s[1000];

fgets(s, sizeof(s), stdin);

s[strcspn(s, "\n")] = '\0'; // Remove newline character

int min\_dist = 1000; // Initialize with a large value

char min\_char = '\0'; // To store the character with the minimum distance

// Iterate over each character in the string

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

for (int j = i + 1; s[j] != '\0'; j++) {

if (s[i] == s[j]) { // Check for duplicates

int dist = j - i;

if (dist < min\_dist) {

min\_dist = dist;

min\_char = s[i];

}

break; // Move to the next character after finding the duplicate

}

}

}

if (min\_char != '\0') {

printf("Character with minimum distance: '%c' with distance %d\n", min\_char, min\_dist);

} else {

printf("No duplicates found\n");

}

return 0;

}

# 31.Extracting data from string.

#include <stdio.h>

#include <string.h>

#include<stdlib.h>

int main() {

int n, count = 0;

// Input the number of passengers

printf("Enter the number of passengers: ");

scanf("%d", &n);

char details[n][16]; // Array to store details of each passenger (15 characters + 1 for null terminator)

// Input the passenger details

for (int i = 0; i < n; i++) {

printf("Enter details of passenger %d: ", i + 1);

scanf("%s", details[i]);

}

// Process each passenger's details

for (int i = 0; i < n; i++) {

// Extract the age (characters 12 and 13)

char age\_str[3];

age\_str[0] = details[i][11];

age\_str[1] = details[i][12];

age\_str[2] = '\0'; // Null-terminate the age string

int age = atoi(age\_str); // Convert the age string to an integer

// Check if the age is greater than 60

if (age > 60) {

count++;

}

}

// Output the number of passengers with age greater than 60

printf("Number of passengers with age greater than 60: %d\n", count);

return 0;

}

# 

# 32.Finding longest and smallest strings

#include <stdio.h>

#include <string.h>

void main() {

int n;

printf("Enter the number of strings: ");

scanf("%d", &n);

char strings[n][100], longest[100], smallest[100];

getchar();

printf("Enter the strings:\n");

for (int i = 0; i < n; i++) {

fgets(strings[i], sizeof(strings[i]), stdin);

strings[i][strcspn(strings[i], "\n")] = '\0';

}

strcpy(longest, strings[0]);

strcpy(smallest, strings[0]);

for (int i = 1; i < n; i++) {

if (strlen(strings[i]) > strlen(longest)) {

strcpy(longest, strings[i]);

}

if (strlen(strings[i]) < strlen(smallest)) {

strcpy(smallest, strings[i]);

}

}

printf("Longest string: %s\n", longest);

printf("Smallest string: %s\n", smallest);

}