```c
/*
Statistical Analysis Tool
Function Prototype: void computeStats(const double *array, int size,
double *average, double *variance)
Data Types: const double*, int, double*
Concepts: Pointers, arrays, functions, passing constant data, pass by
reference.
Details: Compute the average and variance of an array of experimental
results, ensuring the function uses pointers for
 accessing the data and modifying the results.*/
 #include<stdio.h>
 void computeStats(const double *array, int size, double *average, double
*variance);
 void main()
 {
    double array[5]={10,24,56,78,90};
    double *a=array;
    int size=5;
    double average;
    double *avg=&average;
    double variance[4];
    double *v=variance;
    computeStats(a,size,avg,v);

 }
  void computeStats(const double *array, int size, double *average, double
*variance)
  {
    double sum=0;
    for(int i=0;i<5;i++)
    {
        sum+=*(array+i);
    }
    *average=sum/size;
    printf("The average is %.2f\n",*average);
    printf("The variance is \n");
    for(int i=0;i<size-1;i++)
    {
        *(variance+i)=*(array+i+1)-*(array+i);
        printf("%.2f ",*(variance+i));
```

```
    }
  }
```

```
PS D:\projects\quest\C> cd "d:\projects\
The average is 51.60
The variance is
14.00 32.00 22.00 12.00
PS D:\projects\quest\C>
```

```c
/*Data Normalization
Function Prototype: double* normalizeData(const double *array, int size)
Data Types: const double*, int, double*
Concepts: Arrays, functions returning pointers, loops.
Details: Normalize data points in an array, returning a pointer to the new
normalized array.*/
#include<stdio.h>
#include<stdlib.h>
double* normalizeData(const double *array, int size);
void main()
{
double array[5]={1,6,8,7,3};
double *a=array;
int size=5;
double *n;
n=normalizeData(a,size);
for(int i=0;i<size;i++)
{
    printf("%.2f ",*(n+i));
}
}
double* normalizeData(const double *array, int size)
{
    double * normalized =(double *)malloc(size*sizeof(double));
    for(int i=0;i<size;i++)
    {
        if(*(array+i)<5)
        *(normalized+i)=0;
        else
        *(normalized+i)=1;
```

```
    }
return normalized;
}
```

```
PS D:\projects\quest\C> cd "d:\project
0.00 1.00 1.00 1.00 0.00
PS D:\projects\quest\C>
```

```c
/*Experimental Report Generator
Function Prototype: void generateReport(const double *results, const char
*descriptions[], int size)
Data Types: const double*, const char*[], int
Concepts: Strings, arrays, functions, passing constant data.
Details: Generate a report summarizing experimental results and their
descriptions,
 using constant data to ensure the input is not modified*/
#include <stdio.h>
void generateReport(const double *results, const char *descriptions[], int
size);
void main() {
    const double results[] = {95.5, 89.3, 76.2, 88.4, 91.7};
    const char *descriptions[] = {
        "Test 1: Sample A",
        "Test 2: Sample B",
        "Test 3: Sample C",
        "Test 4: Sample D",
        "Test 5: Sample E"
    };
    int size = 5;
    generateReport(results, descriptions, size);

}
void generateReport(const double *results, const char *descriptions[], int
size) {
    printf("Experimental Report:\n");
    printf("----------------------------\n");
    for (int i = 0; i < size; i++)
```

```
        printf("%s - Result: %.2f%%\n", descriptions[i], results[i]);
    printf("----------------------------\n");
    printf("End of Report\n");
}
```

```
Experimental Report:
----------------------------
Test 1: Sample A - Result: 95.50%
Test 2: Sample B - Result: 89.30%
Test 3: Sample C - Result: 76.20%
Test 4: Sample D - Result: 88.40%
Test 5: Sample E - Result: 91.70%
----------------------------
End of Report
PS D:\projects\quest\C>
```

```
/*Data Anomaly Detector
Function Prototype: void detectAnomalies(const double *data, int size,
double threshold, int *anomalyCount)
Data Types: const double*, int, double, int*
Concepts: Decision-making, arrays, pointers, functions.
Details: Detect anomalies in a dataset based on a threshold, updating the
anomaly count by reference.*/
#include<stdio.h>
void detectAnomalies(const double *data, int size, double threshold, int
*anomalyCount);
void main()
{
int anomalycount=0;
int *ac=&anomalycount;
double threshold =5;
int size =5;
double data[5]={3,4,6,7,1};
double *d=data;

detectAnomalies(d,size,threshold,ac);
}
void detectAnomalies(const double *data, int size, double threshold, int
*anomalyCount)
{
```

```
    for(int i=0;i<size;i++)
    {
        if(data[i]>threshold)
        (*anomalyCount)++;
    }
    printf("Anomaly count is %d",*anomalyCount);
}
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS D:\projects\quest\C> cd "d:\projects\ques
Anomaly count is 2
PS D:\projects\quest\C>
```

```
/*Data Classifier
Function Prototype: void classifyData(const double *data, int size, char
*labels[], double threshold)
Data Types: const double*, int, char*[], double
Concepts: Decision-making, arrays, functions, pointers.
Details: Classify data points into categories based on a threshold,
updating an array of labels.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
void classifyData(const double *data, int size, char *labels[], double
threshold);
void main()
{
    double data[5]={4,3,6,7,10};
    double *d=data;
    int size =5;
    char *labels[size];
    double threshold=5;
     for (int i = 0; i < size; i++)
    labels[i] = (char *)malloc(20 * sizeof(char));
    classifyData(d,size,labels,threshold);

}
void classifyData(const double *data, int size, char *labels[], double
threshold)
```

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

        if((*(data+i))>threshold)
       {
        strcpy(labels[i],"pass");
        printf("%s \t",labels[i]);
        }
        else
        {
            strcpy(labels[i],"fail");
        printf("%s \t",labels[i]);
        }
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
fail    fail    pass    pass    pass
PS D:\projects\quest\C>
```

```
/*Neural Network Weight Adjuster
Function Prototype: void adjustWeights(double *weights, int size, double
learningRate)
Data Types: double*, int, double
Concepts: Pointers, arrays, functions, loops.
Details: Adjust neural network weights using a given learning rate, with
weights passed by reference.*/
#include<stdio.h>
void adjustWeights(double *weights, int size, double learningRate);
void main()
{
    double weight[5]={10,17,27,41,59};
    double *w=weight;
    int size=5;
    double learningrate = 7;
    adjustWeights(w,size,learningrate);
    for(int i=0;i<size;i++)
```

```
    {
        printf(" %.2f ",weight[i]);
    }
}
void adjustWeights(double *weights, int size, double learningRate)
{
    for(int i=0;i<size;i++)
    {
        *(weights+i)-=learningRate;
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { g
 3.00  10.00  20.00  34.00  52.00
PS D:\projects\quest\C>
```

```
/*AI Model Evaluator
Function Prototype: void evaluateModels(const double *accuracies, int
size, double *bestAccuracy)
Data Types: const double*, int, double*
Concepts: Loops, arrays, functions, pointers.
Details: Evaluate multiple AI models, determining the best accuracy and
updating it by reference.*/
#include<stdio.h>
void evaluateModels(const double *accuracies, int size, double
*bestAccuracy);
void main()
{
    double accuracy[5]={79.3,87.4,93.5,98.9,99.4};
    double *a=accuracy;
    int size=5;
    double bestaccuracy;
    double *ba=&bestaccuracy;
    evaluateModels(a,size,ba);
     printf("The best accuracy is %f",bestaccuracy);
}
void evaluateModels(const double *accuracies, int size, double
*bestAccuracy)
```

```
{
    for(int i=0;i<size-1;i++)
    {
        if(*(accuracies+i)>*(accuracies+i+1))
        *bestAccuracy=*(accuracies+i);
        else
        *bestAccuracy =*(accuracies+i+1);
    }

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
The best accuracy is 99.400000
PS D:\projects\quest\C>
```

```
/*Decision Tree Constructor
Function Prototype: void constructDecisionTree(const double *features, int
size, int *treeStructure)
Data Types: const double*, int, int*
Concepts: Decision-making, arrays, functions.
Details: Construct a decision tree based on feature data, updating the
tree structure by reference.*/
#include<stdio.h>
void constructDecisionTree(const double *features, int size, int
*treeStructure);
void main()
{
 double features[5]={1.5,3.7,7.8,5.2,9.8};
 double *f=features;
 int size=5;
 int treestructure[5];
 int *ts=treestructure;
 constructDecisionTree(f,size,ts);
 for(int i=0;i<size;i++)
 {
    printf("Node %d feature %d\n",i,treestructure[i]);
 }
}
```

```c
void constructDecisionTree(const double *features, int size, int
*treeStructure)
{
    int best;
    for(int i=0;i<size;i++)
    {
        best=0;
        for(int j=1;j<size;j++)
        {
            if(features[j]>features[best])
            best=j;
        }
        treeStructure[i]=best;
        ((double *)features)[best]=-1;
    }
}
```

```
PS D:\projects\quest\C> c
Node 0 feature 4
Node 1 feature 2
Node 2 feature 3
Node 3 feature 1
Node 4 feature 0
PS D:\projects\quest\C>
```

```c
/*Sentiment Analysis Processor
Function Prototype: void processSentiments(const char *sentences[], int
size, int *sentimentScores)
Data Types: const char*[], int, int*
Concepts: Strings, arrays, functions, pointers.
Details: Analyze sentiments of sentences, updating sentiment scores by
reference.*/
#include <stdio.h>
#include <string.h>
void processSentiments(const char *sentences[], int size, int
*sentimentScores);
void main() {
    const char *sentences[] = {
        "I love this product, it's amazing!",
        "This is the worst service I've ever experienced.",
```

```
        "The quality is good, but it could be better.",
        "Absolutely fantastic work, I am very impressed.",
        "Not bad, but I expected more."
    };
    int size = 5;
    int sentimentScores[size];
    processSentiments(sentences, size, sentimentScores);
    printf("Sentiment Analysis Results:\n");
    for (int i = 0; i < size; i++) {
        printf("Sentence %d: Score = %d\n", i + 1, sentimentScores[i]);
    }
}
void processSentiments(const char *sentences[], int size, int
*sentimentScores) {
    const char *positive[] = {"love", "amazing", "good", "fantastic",
"impressed"};
    const char *negative[] = {"worst", "bad", "not", "terrible",
"horrible"};
    int positiveCount = sizeof(positive) / sizeof(positive[0]);
    int negativeCount = sizeof(negative) / sizeof(negative[0]);
    for (int i = 0; i < size; i++) {
        int score = 0;
        for (int j = 0; j < positiveCount; j++) {
            if (strstr(sentences[i], positive[j]) != NULL)
                score++;
        }
        for (int j = 0; j < negativeCount; j++) {
            if (strstr(sentences[i], negative[j]) != NULL)
                score--;
        }
        sentimentScores[i] = score;
    }
}
```

```
PS D:\projects\quest\C> cd "d:\p
Sentiment Analysis Results:
Sentence 1: Score = 2
Sentence 2: Score = -1
Sentence 3: Score = 1
Sentence 4: Score = 2
Sentence 5: Score = -1
PS D:\projects\quest\C>
```

```c
/*raining Data Generator
Function Prototype: double* generateTrainingData(const double *baseData,
int size, int multiplier)
Data Types: const double*, int, double*
Concepts: Arrays, functions returning pointers, loops.
Details: Generate training data by applying a multiplier to base data,
returning a pointer to the new data array.*/
#include<stdio.h>
#include<stdlib.h>
double* generateTrainingData(const double *baseData, int size, int
multiplier);
void main()
{
    double basedata[5]={2,3,4,5,6};
    double *bd=basedata;
    int size= sizeof(basedata)/sizeof(double);
    int multiplier =5;
    double *t;
    t=generateTrainingData(bd,size,multiplier);
    printf("Training data\n");
    for(int i=0;i<size;i++)
    {
        printf("%.1f ",t[i]);
    }
}
double* generateTrainingData(const double *baseData, int size, int
multiplier)
{
    double* training=(double *)malloc(size*sizeof(double));
    for(int i=0;i<size;i++)
```

```
    {
        *(training+i)=(*(baseData+i))*multiplier;
    }
    return training;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ;
Training data
10.0 15.0 20.0 25.0 30.0
PS D:\projects\quest\C>
```

```
/*Image Filter Application
Function Prototype: void applyFilter(const unsigned char *image, unsigned
char *filteredImage, int width, int height)
Data Types: const unsigned char*, unsigned char*, int
Concepts: Arrays, pointers, functions.
Details: Apply a filter to an image, modifying the filtered image by
reference*/
#include <stdio.h>
#include <stdlib.h>
void applyFilter(const unsigned char *image, unsigned char *filteredImage,
int width, int height);
void main() {
    int width = 4, height = 4;
    unsigned char image[16] = {
        10, 20, 30, 40,
        50, 60, 70, 80,
        90, 100, 110, 120,
        130, 140, 150, 160
    };
    unsigned char filteredImage[16];
    applyFilter(image, filteredImage, width, height);
    printf("Original Image:\n");
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            printf("%3d ", image[i * width + j]);
        }
        printf("\n");
    }
```

```
    printf("\nFiltered Image:\n");
    for (int i = 0; i < height; i++) {
        for (int j = 0; j < width; j++) {
            printf("%3d ", filteredImage[i * width + j]);
        }
        printf("\n");
    }
}
void applyFilter(const unsigned char *image, unsigned char *filteredImage,
int width, int height) {
    int filter[3][3] = {
        { 1, 1, 1 },
        { 1, 1, 1 },
        { 1, 1, 1 }
    };
    for (int i = 1; i < height - 1; i++) {
        for (int j = 1; j < width - 1; j++) {
            int sum = 0;
            for (int k = -1; k <= 1; k++) {
                for (int l = -1; l <= 1; l++)
                sum += image[(i + k) * width + (j + l)] * filter[k + 1][l
+ 1];
            }
            printf("Sum for pixel (%d, %d): %d\n", i, j, sum);
            if (sum < 0)
            sum = 0;
            if (sum > 255)
            sum = 255;
            filteredImage[i * width + j] = (unsigned char)sum;
        }
    }
    for (int i = 0; i < width; i++) {
        filteredImage[i] = 0;
        filteredImage[(height - 1) * width + i] = 0;
    }
    for (int i = 0; i < height; i++) {
        filteredImage[i * width] = 0;
        filteredImage[i * width + (width - 1)] = 0;
    }
}
```

```
 50  60  70  80
 90 100 110 120
130 140 150 160

Filtered Image:
  0   0   0   0
  0 255 255   0
  0 255 255   0
  0   0   0   0
PS D:\projects\quest\C>
```

```c
#include <stdio.h>
#include<stdbool.h>
 bool func(char a[],char b[])
 {
     int count=0;

     while(a[count]!='\0'&&b[count]!='\0')
     {
         if(a[count]!=b[count])
         return false;
         count++;
     }
     return true;
 }
int main(){
char a[]="hello";
char b[]="hello";
bool flag;
flag=func(a,b);

if(flag==true)
printf("The strings are same\n");
else
printf("The strings are not the same");
return 0;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\qu
The strings are same
PS D:\projects\quest\C>
```

```c
#include <stdio.h>
 void func(char a[],char b[],char c[])
 {
     int count1=0,count2=0;
     while(a[count1]!='\0')
     {
         c[count1]=a[count1];
         count1++;
     }
     while(b[count2]!='\0')
     {
         c[count1]=b[count2];
         count1++;
         count2++;
     }
     printf("%s",c);
 }
int main(){
char a[]="hello";
char b[]="world";
char result[12];
func(a,b,result);
return 0;
}
```

```
PS D:\projects\quest\C> cd "d:\projects
helloworld@
PS D:\projects\quest\C>
```

```c
/*String Length Calculation
Requirement: Write a program that takes a string input and calculates its
length using strlen(). The program should handle empty strings and output
appropriate messages.
```

```c
Input: A string from the user.
Output: Length of the string.*/
#include<stdio.h>
#include<string.h>
void main()
{
char a[20];
scanf("%s",a);
printf("Length is %d",strlen(a));
}
```

```
PS D:\projects\quest\C> cd "d:\pr
helloworld
Length is 10
PS D:\projects\quest\C>
```

```c
/*String Copy
Requirement: Implement a program that copies one string to another using
strcpy(). The program should validate if the source string fits into the
destination buffer.
Input: Two strings from the user (source and destination).
Output: The copied string.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[6]="Hello\0";
    char b[5];
    strcpy(b,a);
    printf("%s",b);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\
Hello
PS D:\projects\quest\C>
```

```c
/*Requirement: Create a program that concatenates two strings using
strcat().
 Ensure the destination string has enough space to hold the result.
Input: Two strings from the user.
Output: The concatenated string.*/
#include<stdio.h>
#include<string.h>
int main()
{
    char str1[20]="hello", str2[6]="world";
    strcat(str1, str2);
    printf("\nString after concatenation is:\n%s", str1);
    return 0;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\

String after concatenation is:
helloworld
PS D:\projects\quest\C>
```

```c
/*Requirement: Develop a program that compares two strings using strcmp().
 It should indicate if they are equal or which one is greater.
Input: Two strings from the user.
Output: Comparison result*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[6]="hello";
```

```c
    char b[6]="hello";
    if(strcmp(a,b)==0)
    printf("Both strings are equal");
    else if(strcmp(a,b)>0)
    printf("string a is greater");
    else
    printf("string b is greater");

}
```

```
PS D:\projects\quest\C> cd "d:\proje
Both strings are equal
PS D:\projects\quest\C>
```

```c
/*Requirement: Write a program that converts all characters in a string to
uppercase using strupr().
Input: A string from the user.
Output: The uppercase version of the string.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[20]="Hello world";
    printf("%s\n",a);
    strupr(a);
    printf("%s",a);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Hello world
HELLO WORLD
PS D:\projects\quest\C>
```

```c
/*Requirement: Create a program that searches for a substring within a
given string using strstr() and
returns its starting index or an appropriate message if not found.
```

```c
Input: A main string and a substring from the user.
Output: Starting index or not found message.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[10]="world";
    char b[30]="hello world";
    char *c=strstr(b,a);
    if(c)
    {
        for(int i=0;i<strlen(b);i++)
        {
            if(*c==b[i])
            printf("Starting index is %d",i);
        }
    }
    else
    printf("The string is not in the first string\n");
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Starting index is 6
PS D:\projects\quest\C>
```

```c
/*Requirement: Write a program that finds the first occurrence of a
character in a string using strchr()
 and returns its index or indicates if not found.
Input: A string and a character from the user.
Output: Index of first occurrence or not found message.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[15]="Hello world";
    char b='l';
    char *c=strchr(a,b);
    if(c)
```

```
    {
        for(int i=0;i<strlen(a);i++)
        {
            if(*c==a[i])
            printf("The character is at index %d\n",i);
        }
    }
    else
    printf("The character %c does not exit in the string",c);
}
```

```
PS D:\projects\quest\C> cd "d:\proje
The character is at index 2
The character is at index 3
The character is at index 9
PS D:\projects\quest\C>
```

```
/*Requirement: Implement a function that reverses a given string in place
without using additional memory,
leveraging strlen() for length determination.
Input: A string from the user.
Output: The reversed string.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[20]="Hello world";
    int start=0;
    int end=strlen(a)-1;
    char temp;
    while(start<end)
    {
        temp=a[start];
        a[start]=a[end];
        a[end]=temp;
        start++;
        end--;
    }
    printf("%s",a);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\qu
dlrow olleH
PS D:\projects\quest\C>
```

```c
/*Requirement: Create a program that tokenizes an input string into words
using strtok() and counts how many tokens were found.
Input: A sentence from the user.
Output: Number of words (tokens).*/
#include<stdio.h>
#include<string.h>
void main()
{
    char string[100]="Hello world,how are you?";
    char *delim=" ,?";
    char *token=strtok(string,delim);
    while(token!=NULL)
    {
        printf("%s \n",token);
        token=strtok(NULL,delim);
    }

}
```

```
PS D:\projects\quest\C> cd "d:\pr
Hello
world
how
are
you
PS D:\projects\quest\C>
```

```c
/*Requirement: Write a function that duplicates an input string
(allocating new memory) using strdup() and
```

```c
displays both original and duplicated strings.
Input: A string from the user.
Output: Original and duplicated strings.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char string[20];
    scanf("%s",string);
     char *dup=strdup(string);
    printf("%s",dup);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
string
string
PS D:\projects\quest\C>
```

```c
/*Requirement: Develop a program to compare two strings without case
sensitivity using strcasecmp() and report equality or differences.
Input: Two strings from the user.
Output: Comparison result.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char str[10]="hi";
    char str2[10]="H";
    int n;
    n=strcasecmp(str,str2);
 if(n==0)
 printf("Both strings are equal\n");
 else
 printf("Booth strings are not equal,differnce %d\n",n);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\
Booth strings are not equal,differnce 1
PS D:\projects\quest\C>
```

```c
/*Requirement:
Implement functionality to trim leading and trailing whitespace from a given string,
utilizing pointer arithmetic with strlen().
Input: A string with extra spaces from the user.
Output: Trimmed version of the string.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char string[20]="  Hello World  ";
    char trim[20];
    int start =0,end=0,count=0;
    for(int i=0;i<strlen(string);i++)
    {
        if(string[i]!=' ')
        {start=i;
        break;
        }
    }
    for(int i=strlen(string)-1;i>0;i--)
    {
        if(string[i]!=' ')
        {
            end=i;
            break;
        }
    }
    for(int i=start;i<=end;i++)
    {
        trim[count]=string[i];
        count++;
    }
    printf("trimmed  string is :%s",trim);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ;
trimmed  string is :Hello World
PS D:\projects\quest\C>
```

```c
/*Requirement: Write a program that finds the last occurrence of a
character in a string using manual
iteration instead of library functions, returning its index.
Input: A string and a character from the user.
Output: Index of last occurrence or not found message.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[20]="hello world";
    char b='l';
    int pos=0;
    for(int i=0;i<strlen(a);i++)
    {
        if(a[i]==b)
        pos=i;
    }
    if(pos)
    printf("The last occurence is %d",pos);
    else
    printf("Character not found");
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
The last occurence is 9
PS D:\projects\quest\C>
```

```c
/*Requirement: Create a program that counts how many vowels are present in
an input string by iterating through each character.
Input: A string from the user.
Output: Count of vowels.*/
#include<stdio.h>
#include<string.h>
```

```c
void main()
{
    char string[]="hello world and how are you";
    char *str=string;
    char *vowals="aeiouAEIOU";
    int count=0;
    while(*str)
    {
        if(strchr(vowals,*str))
        count++;
        str++;
    }
    printf("Number of vowels is %d",count);

}
```

```
PS D:\projects\quest\C> cd "d:\
Number of vowels is 9
PS D:\projects\quest\C>
```

```c
/*Requirement: Implement functionality to count how many times a specific
character appears in an input string,
 allowing for case sensitivity options.
Input: A string and a character from the user.
Output: Count of occurrences.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char b='l';
    int count =0;
     char a[50]="hello world like the list";
    for(int i=0;i<strlen(a);i++)
    {
        if(a[i]==b)
        count++;
    }
```

```
    printf("The number of times the speciific character appears is
%d",count);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc ten
The number of times the speciific character appears is 5
PS D:\projects\quest\C>
```

```
/*Requirement: Write a function that removes all occurrences of a
specified character from an input string, modifying it in place.
Input: A string and a character to remove from it.
Output: Modified string without specified characters.*/
#include <stdio.h>
#include <string.h>
void removeChar(char *str, char ch) {
    int i = 0, j = 0;
    while (str[i] != '\0') {
        if (str[i] != ch) {
            str[j++] = str[i];
        }
        i++;
    }
    str[j] = '\0';
}
void main() {
    char str[100];
    char ch;

    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    printf("Enter the character to remove: ");
    scanf("%c", &ch);
    removeChar(str, ch);
    printf("Modified string: %s\n", str);
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Enter a string: helloworld
Enter the character to remove: l
Modified string: heoword
PS D:\projects\quest\C>
```

```c
/*Requirement: Develop an algorithm to check if an input string is a
palindrome by comparing characters
from both ends towards the center, ignoring case and spaces.
Input: A potential palindrome from the user.<
Output: Whether it is or isn't a palindrome.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char a[20];
    int start=0,end=0,flag=1;
    printf("Enter the word\n");
    scanf("%s",a);
    end=strlen(a)-1;
    for(int i=0,j=end;i<strlen(a)/2,j>strlen(a)/2;i++,j--)
    {
        if(a[i]!=a[j])
        flag=0;
    }
    if(flag==0)
    printf("It is not a pallindrome\n");
    else
    printf("It is a pallindrome\n");

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter the word
malayalam
It is a pallindrome
PS D:\projects\quest\C>
```

```c
/*Requirement: Create functionality to extract a substring based on
specified start index and length parameters,
ensuring valid indices are provided by users.
Input: A main string, start index, and length from the user.
Output: Extracted substring or error message for invalid indices*/
#include <stdio.h>
#include <string.h>

void extractSubstring(char *str, int start, int length) {
    int strLength = strlen(str);
    char substring[length + 1];
    if (start < 0 || start >= strLength || length < 0) {
        printf("Error: Invalid start index or length.\n");
        return;
    }
    if (start + length > strLength) {
        length = strLength - start;
    }
    strncpy(substring, &str[start], length);
    substring[length] = '\0';
    printf("Extracted substring: %s\n", substring);
}

void main() {
    char str[100];
    int start, length;
    printf("Enter a string: ");
    scanf("%99[^\n]", str);
    printf("Enter start index: ");
    scanf("%d", &start);
    printf("Enter length of the substring: ");
    scanf("%d", &length);
```

```
    extractSubstring(str, start, length);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\
Enter a string: helloworld
Enter start index: 2
Enter length of the substring: 5
Extracted substring: llowo
PS D:\projects\quest\C>
```

```
/*Requirement: Implement functionality to sort characters in an input
string alphabetically, demonstrating usage of
nested loops for comparison without library sorting functions.
Input: A string from the user.
Output: Sorted version of the characters in the string.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char str[20];
    int len;
    char temp;
    printf("Enter the string\n");
    scanf("%s",str);
    len=strlen(str);
    for(int i=0;i<len-1;i++)
    {
        for(int j=i+1;j<len;j++)
        {
            if(str[i]>str[j])
            {
                temp=str[i];
                str[i]=str[j];
                str[j]=temp;
            }
        }
```

```
    }
    printf("%s",str);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Enter the string
helloworld
dehlllloorw
PS D:\projects\quest\C>
```

```c
/*Requirement: Write code to count how many words are present in an input
sentence by identifying
 spaces as delimiters, utilizing strtok().
Input: A sentence from the user.
-  Output: Number of words counted.*/
#include<stdio.h>
#include<string.h>
int countWords(const char *sentence) {
    char tempSentence[strlen(sentence) + 1];
    strcpy(tempSentence, sentence);
    char *token = strtok(tempSentence, " ");
    int wordCount = 0;
    while (token != NULL) {
        wordCount++;
        token = strtok(NULL, " ");
    }
    return wordCount;
}
void main()
{
    char sentence[100];
    printf("Enter a sentence: ");
    fgets(sentence, sizeof(sentence), stdin);
    sentence[strcspn(sentence, "\n")] = '\0';
    int wordCount = countWords(sentence);
    printf("Number of words in the sentence: %d\n", wordCount);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tem
Enter a sentence: hello i am rahul
Number of words in the sentence: 4
PS D:\projects\quest\C>
```

```
/* Requirement: Develop an algorithm to remove duplicate characters while
maintaining their first occurrence order in an input string.
-  Input: A string with potential duplicate characters.
-  Output: Modified version of the original without duplicates.*/
#include<stdio.h>
#include<string.h>
void main()
{
    char str[20]="hello world";
    int len=strlen(str);
     for(int i = 0; i < len; i++) {
        for(int j = i + 1; j < len; j++) {
            if(str[i] == str[j]) {

                for(int k = j; k < len; k++) {
                    str[k] = str[k + 1];
                }
                len--;
                j--;
            }
        }
    }
    printf("%s",str);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\
helo wrd
PS D:\projects\quest\C>
```

```
/*-  Requirement: Create functionality to find the first non-repeating
character in an input string, demonstrating
```

```c
effective use of arrays for counting occurrences.
-  Input: A sample input from the user.
-  Output: The first non-repeating character or indication if all are
repeating.*/
#include<stdio.h>
#include<string.h>
#define MAX_CHAR 256
char findFirstNonRepeating(char *str) {
    int count[MAX_CHAR] = {0};
    for(int i = 0; str[i] != '\0'; i++) {
        count[str[i]]++;
    }
    for(int i = 0; str[i] != '\0'; i++) {
        if(count[str[i]] == 1) {
            return str[i];
        }
    }
    return '\0';
}
void main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    char result = findFirstNonRepeating(str);
    if(result != '\0') {
        printf("The first non-repeating character is: %c\n", result);
    } else {
        printf("All characters are repeating.\n");
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter a string: hellloworld
The first non-repeating character is: h
PS D:\projects\quest\C>
```

```c
/*- Requirement:
Implement functionality to convert numeric strings into integer values
without using
 standard conversion functions like atoi(), handling invalid inputs
gracefully.
```

```c
- Input: A numeric string.
- Output: Converted integer value or error message.*/
#include <stdio.h>
#include <ctype.h>
int func(const char *str) {
    int result = 0;
    int i = 0;
    int sign = 1;
    if (str == NULL || *str == '\0') {
        printf("Error: Invalid input\n");
        return -1;
    }
    if (str[i] == '-') {
        sign = -1;
        i++;
    }
    for (; str[i] != '\0'; i++)
     {
        if (!isdigit(str[i]))
         {
            printf("Error: Invalid character in input\n");
            return -1;
        }
        result = result * 10 + (str[i] - '0');
    }

    return result * sign;
}

void main() {
    char str[100];
    printf("Enter a numeric string: ");
    scanf("%s", str);
    int result = func(str);
    if (result != -1) {
        printf("Converted integer value: %d\n", result);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\project
Enter a numeric string: 786
Converted integer value: 786
PS D:\projects\quest\C>
```

```c
/*- Requirement: Write code to check if two strings are anagrams by
sorting their characters and comparing them.
- Input: Two strings.
- Output: Whether they are anagrams.*/
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int func(char *str1, char *str2) {
    if (strlen(str1) != strlen(str2)) {
        return 0;
    }
    for (int i = 0; i < strlen(str1) - 1; i++)
    {
        for (int j = i + 1; j < strlen(str1); j++)
         {
            if (str1[i] > str1[j])
            {
                char temp = str1[i];
                str1[i] = str1[j];
                str1[j] = temp;
            }
        }
    }
    for (int i = 0; i < strlen(str2) - 1; i++)
     {
        for (int j = i + 1; j < strlen(str2); j++)
        {
            if (str2[i] > str2[j])
             {
                char temp = str2[i];
                str2[i] = str2[j];
```

```
                str2[j] = temp;
            }
        }
    }
    return strcmp(str1, str2) == 0;
}

void main() {
    char str1[100], str2[100];
    printf("Enter the first string: ");
    scanf("%s", str1);
    printf("Enter the second string: ");
    scanf("%s", str2);

    for (int i = 0; str1[i]; i++) {
        str1[i] = tolower(str1[i]);
    }
    for (int i = 0; str2[i]; i++) {
        str2[i] = tolower(str2[i]);
    }
    if (func(str1, str2)) {
        printf("The strings are anagrams.\n");
    } else {
        printf("The strings are not anagrams.\n");
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Enter the first string: listen
Enter the second string: silent
The strings are anagrams.
PS D:\projects\quest\C>
```

```
/*- Requirement: Create functionality to merge two strings alternately into one while handling cases where
strings may be of different lengths.
```

```c
- Input: Two strings.
- Output: Merged alternating characters.*/
#include <stdio.h>
#include <string.h>
void mergeStringsAlternately(char *str1, char *str2, char *result) {
    int i = 0, j = 0, k = 0;
    while (str1[i] != '\0' && str2[j] != '\0')
    {
        result[k++] = str1[i++];
        result[k++] = str2[j++];
    }
    while (str1[i] != '\0')
    result[k++] = str1[i++];
    while (str2[j] != '\0')
    result[k++] = str2[j++];

    result[k] = '\0';
}

void main() {
    char str1[100], str2[100], result[200];
    printf("Enter the first string: ");
    scanf("%s", str1);
    printf("Enter the second string: ");
    scanf("%s", str2);
    mergeStringsAlternately(str1, str2, result);
    printf("Merged alternating string: %s\n", result);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter the first string: helloworld
Enter the second string: howareyou
Merged alternating string: hheolwlaorweoyrolud
PS D:\projects\quest\C>
```

```c
/*- Requirement: Develop code to count consonants while ignoring vowels
and whitespace characters.
```

```c
- Input: Any input text.
- Output: Count of consonants.*/
#include <stdio.h>
#include <ctype.h>
int countConsonants(const char *str) {
    int consonantCount = 0;
    char ch;
    while (*str) {
        ch = tolower(*str);
        if (isalpha(ch) && ch != 'a' && ch != 'e' && ch != 'i' && ch != 
'o' && ch != 'u') {
            consonantCount++;
        }
        str++;
    }
    return consonantCount;
}

void main() {
    char text[100];
    printf("Enter a text: ");
    fgets(text, sizeof(text), stdin);
    int consonantCount = countConsonants(text);
    printf("The number of consonants is: %d\n", consonantCount);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\c
Enter a text: hello world
The number of consonants is: 7
PS D:\projects\quest\C>
```

```c
/*- Requirement: Write functionality to replace all occurrences of one 
substring with another within a given main string.
- Input: Main text, target substring, replacement substring.
- Output: Modified main text after replacements.*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

```c
void replaceSubstring(char *str, const char *old_substr, const char
*new_substr) {
    char *pos;
    int old_len = strlen(old_substr);
    int new_len = strlen(new_substr);
    int index = 0;
    while ((pos = strstr(str + index, old_substr)) != NULL)
     {
        char *temp = (char *)malloc(strlen(str) + (new_len - old_len) +
1);
        strncpy(temp, str, pos - str);
        temp[pos - str] = '\0';
        strcat(temp, new_substr);
        strcat(temp, pos + old_len);
        strcpy(str, temp);
        free(temp);
        index = pos - str + new_len;
    }
}

void main() {
    char str[200];
    char old_substr[50], new_substr[50];
    printf("Enter the main text: ");
    fgets(str, sizeof(str), stdin);
    str[strcspn(str, "\n")] = '\0';
    printf("Enter the target substring to replace: ");
    fgets(old_substr, sizeof(old_substr), stdin);
    old_substr[strcspn(old_substr, "\n")] = '\0';

    printf("Enter the replacement substring: ");
    fgets(new_substr, sizeof(new_substr), stdin);
    new_substr[strcspn(new_substr, "\n")] = '\0';
    replaceSubstring(str, old_substr, new_substr);
    printf("\nModified text: %s\n", str);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { g
Enter the main text: The sea has sea ships and sea men
Enter the target substring to replace: sea
Enter the replacement substring: sky

Modified text: The sky has sky ships and sky men
PS D:\projects\quest\C>
```

```c
/* Requirement: Create code that counts how many times one substring
appears within another larger main
 text without overlapping occurrences.
- Input: Main text and target substring.
- Output: Count of occurrences.*/
#include <stdio.h>
#include <string.h>
int countOccurrences(const char *mainText, const char *subText) {
    int count = 0;
    const char *pos = mainText;
    while ((pos = strstr(pos, subText)) != NULL) {
        count++;
        pos += strlen(subText);
    }
    return count;
}
void main() {
    char mainText[200];
    char subText[50];
    printf("Enter the main text: ");
    fgets(mainText, sizeof(mainText), stdin);
    mainText[strcspn(mainText, "\n")] = '\0';
    printf("Enter the target substring: ");
    fgets(subText, sizeof(subText), stdin);
    subText[strcspn(subText, "\n")] = '\0';
    int count = countOccurrences(mainText, subText);
    printf("\nThe substring \"%s\" appears %d times in the main text.\n",
subText, count);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) {
Enter the main text: hello world ,world is at peace
Enter the target substring: world

The substring "world" appears 2 times in the main text.
PS D:\projects\quest\C>
```

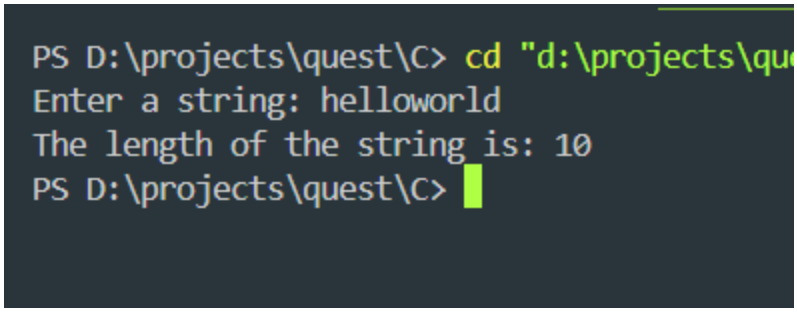
```c
/*- Requirement: Finally, write your own implementation of strlen()
function from scratch,
demonstrating pointer manipulation techniques.
- Input: Any input text.
- Output: Length calculated by custom function.*/
#include <stdio.h>
int my_strlen(const char *str) {
    const char *ptr = str;
    int length = 0;
    while (*ptr != '\0') {
        length++;
        ptr++;
    }

    return length;
}

void main() {
    char string[200];
    printf("Enter a string: ");
    scanf("%s",string);

    int length = my_strlen(string);

    printf("The length of the string is: %d\n", length);
}
```


PS D:\projects\quest\C> cd "d:\projects\qu
Enter a string: helloworld
The length of the string is: 10
PS D:\projects\quest\C>