```c
/*Write a function void reverseString(char *str) that takes a pointer to a
string and reverses the string in place.
*/
#include<stdio.h>
#include<string.h>
void reverseString(char *str);
void main()
{
    char string[]="Hello world";
    char *str=string;
     printf("%s\n",string);
    reverseString(str);
    printf("%s",string);
}
void reverseString(char *str)
{
     char *start = str;
    char *end = str + strlen(str) - 1;
    char temp;
    while (start < end) {
        temp = *start;
        *start = *end;
        *end = temp;
        start++;
        end--;
    }
}
```

```
PS D:\projects\quest\C> cd "d:\project
Hello world
dlrow olleH
PS D:\projects\quest\C>
```

```c
/*Implement a function void concatenateStrings(char *dest, const char
*src)
 that appends the source string to the destination string using
pointers.*/
 #include<stdio.h>
 #include<string.h>
```

```c
void concatenateStrings(char *dest, const char *src);
void main()
{
   char str1[20]="hello";
   char str2[]="world";
   concatenateStrings(str1,str2);
   printf("%s",str1);
}
void concatenateStrings(char *dest, const char *src)
{
   char *start=dest+strlen(dest);
   while(*src!='\0')
   {
       *start=*src;
       start++;
       src++;
   }


}
```

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

```c
/*Create a function int stringLength(const char *str) that calculates and
returns the length of a string using pointers*/
#include<stdio.h>
int stringLength(const char *str);
void main()
{
char string[]="Hello world";
char *str=string;
int c;
c=stringLength(str);
printf("length of string is %d",c);
}
int stringLength(const char *str)
{
    int count =0;
```

```
    while(*str!='\0')
    {
        count++;
        str++;
    }
    return count;
}
```

```
PS D:\projects\quest\C> cd "d:\proj
length of string is 11
PS D:\projects\quest\C>
```

```
/*Write a function int compareStrings(const char *str1, const char *str2)
that compares two strings
lexicographically and returns 0 if they are equal, a positive number if
str1 is greater, or a negative number if str2 is greater.*/
#include<stdio.h>
int compareStrings(const char *str1, const char *str2);
void main()
{
    char str1[]="Hello world";
    char str2[]="hello world";
    char *s1,*s2;
    int num;
    s1=str1;
    s2=str2;
    num=compareStrings(s1,s2);
    if(num>0)
    printf("String 1 is greater");
    else if(num<0)
    printf("String 2 is greater");
    else
    printf("Both strings are equal");
}
int compareStrings(const char *str1, const char *str2)
{
    int count=0;
```

```
    while(*str1!='\0' && *str2!='\0')
    {
        if(*str1>*str2)
        count++;
        else if(*str1<*str2)
        count--;
        else ;
        str1++;
        str2++;
    }
    return count;
}
```

```
PS D:\projects\quest\C> cd "d:\
String 2 is greater
PS D:\projects\quest\C>
```

```
/*Implement char* findSubstring(const char *str, const char *sub) that
returns a pointer to the
 first occurrence of the substring sub in the string str, or NULL if the
substring is not found.*/
 #include<stdio.h>
 #include<string.h>
 char* findSubstring(const char *str, const char *sub);
 void main()
 {
    char string[20]="Hello world";
    char sub[]="rahul";
    const char *str,*s;
    char * result;
    str=string;
    s=sub;
    result=findSubstring(str,s);
    if(result !=NULL)
    printf("starting address of  sub string is %p",result);
    else
    printf("Substring not found");
 }
 char* findSubstring(const char *str, const char *sub)
 {
```

```
    char *r;
    int len=strlen(sub);
    int i;
    while(*str!='\0')
    {   if(*str==sub[0])
        {
            r=(char *)str;
        for(i=0;i<len;i++)
        {
            if(*(str+i)!=sub[i])
            break;
        }
        if(i==len)
        return r;
        }
        str++;
    }
    return NULL;
}
```

```
PS D:\projects\quest\C> cd "d:
Substring not found
PS D:\projects\quest\C>
```

```
/*Write a function void replaceChar(char *str, char oldChar, char newChar)
that replaces all occurrences of oldChar with newChar in the given
string*/
#include <stdio.h>
void replaceChar(char *str, char oldChar, char newChar);
int main() {
    char string[100];
    char oldChar, newChar;
    printf("Enter the string: ");
    scanf("%99[^\n]", string);
    printf("Enter the character to replace: ");
    scanf(" %c", &oldChar);
    printf("Enter the replacement character: ");
    scanf(" %c", &newChar);
    replaceChar(string, oldChar, newChar);
    printf("Modified string: %s\n", string);
```

```c
}
void replaceChar(char *str, char oldChar, char newChar) {
    while (*str) {
        if (*str == oldChar) {
            *str = newChar;
        }
        str++;
    }
}
```

```
PS D:\projects\quest\C> cd "d:\projects
Enter the string: helloworld
Enter the character to replace: h
Enter the replacement character: c
Modified string: celloworld
PS D:\projects\quest\C>
```

```c
/*Create a function void copyString(char *dest, const char *src) that
copies
the content of the source string src to the destination string dest.*/
#include <stdio.h>
void copyString(char *dest, const char *src);
void main() {
    char src[100], dest[100];
    printf("Enter the source string: ");
    scanf("%99[^\n]", src);
    copyString(dest, src);
    printf("Copied string: %s\n", dest);
}
void copyString(char *dest, const char *src) {
    while (*src) {
        *dest = *src;
        dest++;
        src++;
    }
    *dest = '\0';
```

```
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Enter the source string: helloworld
Copied string: helloworld
PS D:\projects\quest\C>
```

```c
/*Implement int countVowels(const char *str) that counts and returns the
number of vowels in a given string.*/
#include <stdio.h>
int countVowels(const char *str);
int main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%99[^\n]", str);
    int vowelsCount = countVowels(str);
    printf("Number of vowels: %d\n", vowelsCount);
}
int countVowels(const char *str)
{
    int count = 0;
    char vowels[] = "aeiouAEIOU";
    while (*str)
     {
        for (int i = 0; vowels[i] != '\0'; i++)
         {
            if (*str == vowels[i])
            {
                count++;
                break;
            }
        }
        str++;
    }
    return count;
}
```

```
PS D:\projects\quest\C> cd "d:\proj
Enter a string: helloworld
Number of vowels: 3
PS D:\projects\quest\C>
```

```c
/*Write a function int isPalindrome(const char *str) that checks if a
given string is a palindrome and returns 1 if true, otherwise 0*/
#include <stdio.h>
#include <string.h>
int isPalindrome(const char *str);
void main() {
    char str[100];
    printf("Enter a string: ");
    scanf("%s", str);
    if (isPalindrome(str)) {
        printf("The string is a palindrome.\n");
    } else {
        printf("The string is not a palindrome.\n");
    }
}
int isPalindrome(const char *str) {
    int length = strlen(str);
    int start = 0, end = length - 1;
    while (start < end) {
        if (str[start] != str[end]) {
            return 0;
        }
        start++;
        end--;
    }
    return 1;
}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter a string: malayalam
The string is a palindrome.
PS D:\projects\quest\C>
```

```c
/*Create a function void tokenizeString(char *str, const char *delim, void
(*processToken)(const char *))
that tokenizes the string str using delimiters in delim, and for each
token, calls processToken*/
#include<stdio.h>
#include<string.h>
void tokenizeString(char *str, const char *delim, void
(*processToken)(const char *))
{
    char *token=strtok(str,delim);
    while(token!=NULL)
    {
        processToken(token);
        token=strtok(NULL,delim);
    }
}
void print(const char * str)
{
    printf("%s\n",str);
}
void main()
{
    char string[100];
    char delim[]=" ,.-";
    printf("Enter the string\n");
    fgets(string,sizeof(string),stdin);
     string[strcspn(string, "\n")] = '\0';
     tokenizeString(string,delim,print);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\
Enter the string
hello world,how are you
hello
world
how
are
you
PS D:\projects\quest\C>
```

```c
/*Write a program that dynamically allocates memory for
an array of integers, fills it with values from 1 to n, and then frees the
allocated memory.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n;
    printf("Enter the size of array\n");
    scanf("%d",&n);
    int * array=(int *)malloc(n*sizeof(int));
    printf("Enter the elements\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
     for(int i=0;i<n;i++)
    {
        printf("%d ",array[i]);
    }
    free(array);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest
Enter the size of array
5
Enter the elements
1 2 3 4 5
1 2 3 4 5
PS D:\projects\quest\C>
```

```c
/*Implement a function that dynamically allocates memory for a string, reads a
string input from the user, and then prints the string. Free the memory after use.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n;
    printf("Enter size of string\n");
    scanf("%d",&n);
    char *str=(char *)malloc(n * sizeof(char));
    printf("Enter string\n");
     getchar();
    fgets(str,n,stdin);
    printf("%s",str);
    free(str);

}
```

```
PS D:\projects\quest\C> cd "d:\pr
Enter size of string
20
Enter string
hello world
hello world
PS D:\projects\quest\C>
```

```c
/*Write a program that dynamically allocates memory for an array of n integers, fills it with values,
 resizes the array to 2n using realloc(), and fills the new elements with values.*/
 #include<stdio.h>
 #include<stdlib.h>
 void main()
 {
    int n;
    printf("Enter the array size\n");
    scanf("%d",&n);
    int *array=(int *)malloc(n * sizeof(int));
    printf("Enter the elements\n");
    for(int i=0;i<n;i++)
    {
        scanf("%d",&array[i]);
    }
    printf("Elements are\n");
    for(int i=0;i<n;i++)
    {
        printf("%d ",array[i]);
    }
    int * narray=(int *)realloc(array,2*n*sizeof(int));
    printf("\nEnter the rest of the elements\n");
    for(int i=n;i<2*n;i++)
    scanf("%d",&narray[i]);
    printf("Elements are\n");
```

```
    for(int i=0;i<2*n;i++)
    printf("%d ",narray[i]);
}
```

```
PS D:\projects\quest\C> cd "d:\projec
Enter the array size
5
Enter the elements
1 2 3 4 5
Elements are
1 2 3 4 5
Enter the rest of the elements
6 7 8 9 10
Elements are
1 2 3 4 5 6 7 8 9 10
PS D:\projects\quest\C>
```

```
/*Create a function that dynamically allocates memory for a 2D array
(matrix) of size m x n, fills it with values, and then deallocates the
memory.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,m;
    printf("Enter the size of matrix\n");
    scanf("%d %d",&n,&m);
    int **matrix=(int **)malloc(m * sizeof(int *));
    if (matrix == NULL)
    printf("Memory allocation failed for rows.\n");
    for(int i=0;i<m;i++)
    {
        matrix[i]=(int *)malloc(n * sizeof(int));
        if(matrix[i]==NULL)
        {
            printf("Memmory allocation failed");
            for(int j=0;i<i;j++)
```

```
            free(matrix[i]);
            free(matrix);
        }
    }
    printf("Enter the elements\n");
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        scanf("%d",&matrix[i][j]);
    }
    printf("The matrix is\n");
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<n;j++)
        printf("%d ",matrix[i][j]);
        printf("\n");
    }
    for(int i=0;i<m;i++)
    free(matrix[i]);
    free(matrix);
}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter the size of matrix
3 3
Enter the elements
1 2 3 4 5 6 7 8 9
The matrix is
1 2 3
4 5 6
7 8 9
PS D:\projects\quest\C>
```

```
/*Implement a function that takes two strings, dynamically allocates
memory to concatenate them,
and returns the new concatenated string. Ensure to free the memory after
use.*/
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
```

```c
void main()
{
    char *str1,*str2;
    int n,len1,len2;
    printf("Enter the length \n");
    scanf("%d",&n);
    str1=(char *)malloc(n * sizeof(char));
    str2=(char *)malloc(n * sizeof(char));
    printf("Enter string 1\n");
    scanf("%s",str1);
    printf("Enter string 2\n");
    scanf("%s",str2);
    len1=strlen(str1);
    len2=strlen(str2);
    char*result=(char *)malloc((len1+len2+1)*sizeof(char));
    for(int i=0;i<len1;i++)
    {
        result[i]=str1[i];
    }
    for(int i=0;i<len2;i++)
    {
        result[len1+i]=str2[i];
    }
    result[len1+len2]='\0';
    printf("%s",result);
    free(result);
    free(str1);
    free(str2);

}
```

```
PS D:\projects\quest\C> cd "d:
Enter the length
10
Enter string 1
hello
Enter string 2
world
helloworld
PS D:\projects\quest\C>
```

```c
/*Define a struct for a student with fields like name, age, and grade.
Write a program that dynamically
allocates memory for a student, fills in the details, and then frees the
memory.*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct st{
    char name[20];
    int age;
    char grade;
};
void main()
{
    struct st *student=(struct st*)malloc(sizeof(struct st));
    if(student==NULL)
    {
        printf("failed to allocate memmory\n");
    }
    printf("Enter the name of student\n");
    fgets(student->name,sizeof(student->name),stdin);
    printf("Age of student\n");
    scanf("%d",&student->age);
    printf("Enter grade of student\n");
    getchar();
    scanf("%c",&student->grade);
    printf("\n");
```

```
    printf("Name of students : %s\n",student->name);
    printf("Age of student : %d\n",student->age);
    printf("Grade of student : %c",student->grade);
    free(student);


}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter the name of student
rahul
Age of student
12
Enter grade of student
A

Name of students : rahul
Age of student : 12
Grade of student : A
PS D:\projects\quest\C>
```

```
/*Write a program that dynamically allocates memory for an array of
pointers to integers,
fills each integer with values, and then frees all the allocated memory.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n;
    printf("Enter the number of integers\n");
    scanf("%d",&n);
    int  **num=(int **)malloc(n*sizeof(int *));
    if(num==NULL)
    {printf("Allocation failed\n");
    return;
    }
    for(int i=0;i<n;i++)
    {
       num[i]=(int *)malloc(sizeof(int));
```

```
    }
    printf("Enter the values\n");
    for (int i = 0; i < n; i++)
    {
        scanf("%d",num[i]);
    }
    printf("The values in the array are\n");
    for(int i=0;i<n;i++)
    printf("%d ",*num[i]);
    for(int i=0;i<n;i++)
    free(num[i]);
    free(num);
}
```

```
PS D:\projects\quest\C> cd "d:\pr
Enter the number of integers
5
Enter the values
1 2 3 4 5
The values in the array are
1 2 3 4 5
PS D:\projects\quest\C>
```

```
/*Create a program that dynamically allocates memory for a 3D array of
integers, fills it with values, and deallocates the memory.*/
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int n,m,p;
    printf("Enter the the three dimensions\n");
    scanf("%d %d %d",&n,&m,&p);
    int ***matrix=(int ***)malloc(n*sizeof(int **));
    if(matrix==NULL)
```

```
    printf("memory allocation failed\n");
    for(int i=0;i<n;i++)
    {
        matrix[i]=(int **)malloc(m*sizeof(int *));
        for(int j=0;j<m;j++)
        {
            matrix[i][j]=(int *)malloc(p*sizeof(int));
        }
    }
    printf("Enter the elements\n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            for(int k=0;k<p;k++)
            {
                scanf("%d",&matrix[i][j][k]);
            }
        }
    }
    printf("The matrix is \n");
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            for(int k=0;k<p;k++)
            printf("%d ",matrix[i][j][k]);
            printf("\n");

        }
        printf("\n");
    }


    for(int i=0;i<n;i++)
    {
        for(int j=0;j<m;j++)
        {
            free(matrix[i][j]);
        }
```

```
        free(matrix[i]);
    }
    free(matrix);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRun
Enter the the three dimensions
3 3 3
Enter the elements
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
The matrix is
1 2 3
4 5 6
7 8 9

10 11 12
13 14 15
16 17 18

19 20 21
22 23 24
25 26 27

PS D:\projects\quest\C>
```

```
/*Write a function void swap(int **a, int **b) that swaps the values of
two integer pointers using double pointers.*/
#include<stdio.h>
void swap(int **a, int **b);
void main()
{
    int a,b;
    int *i,*j;
    int **m,**n;
    printf("Enter the two numberrs\n");
    scanf("%d %d",&a,&b);
    i=&a;
    j=&b;
    m=&i;
    n=&j;
    swap(m,n);

}
```

```c
void swap(int **a,int **b)
{
    int temp;
    temp=**a;
    **a=**b;
    **b=temp;
    printf("The first num is %d and second num is %d",**a,**b);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\ques
Enter the two numberrs
5 6
The first num is 6 and second num is 5
PS D:\projects\quest\C>
```

```c
/*Implement a function void allocateArray(int **arr, int size)
that dynamically allocates memory for an array of integers using a double
pointer.*/
#include<stdio.h>
#include<stdlib.h>
void allocateArray(int **arr, int size);
void main()
{
int n;
int **arr;
printf("Enter the size of array\n");
scanf("%d",&n);
allocateArray(arr,n);
}
void allocateArray(int **arr, int size)
{
arr=(int **)malloc(size*sizeof(int *));
for(int i=0;i<size;i++)
{
    arr[i]=(int *)malloc(sizeof(int));
    scanf("%d",arr[i]);
}
printf("Array is \n");
```

```c
for(int i=0;i<size;i++)
{
    printf("%d ",*arr[i]);
}
for(int i=0;i<size;i++)
{
    free(arr[i]);
}
free(arr);
}
```

```
PS D:\projects\quest\C> cd "d
Enter the size of array
5
1 2 3 4 5
Array is
1 2 3 4 5
PS D:\projects\quest\C>
```

```c
/*Write a function void modifyString(char **str) that takes a double
pointer to a string,
 dynamically allocates a new string, assigns it to the pointer, and
modifies the original string.*/
 #include <stdio.h>
#include <stdlib.h>
#include <string.h>
void modifyString(char **str) {
    *str = (char *)malloc(50 * sizeof(char));
    if (*str == NULL) {
        printf("Memory allocation failed!\n");
        return;
    }
    strcpy(*str, "This is the modified string!");
}
void main() {
    char *origin = "This is the original string";
    printf("Original string: %s\n", origin);
```

```
    modifyString(&origin);
    printf("Modified string: %s\n", origin);
    free(origin);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if
Original string: This is the original string
Modified string: This is the modified string!
PS D:\projects\quest\C>
```

```
/*Create a simple program that demonstrates how to use a pointer to a
pointer to access and modify the value of an integer.*/
#include<stdio.h>
void main()
{
    int a=5;
    int *b=&a;
    int **c=&b;
    printf("Value of a is %d\n",a);
    printf("Enter the new value of a\n");
    scanf("%d",*c);
    printf("value of a is %d",a);
}
```

```
PS D:\projects\quest\C> cd "d:\project
Value of a is 5
Enter the new value of a
10
value of a is 10
PS D:\projects\quest\C>
```

```
/*Write a function int** create2DArray(int rows, int cols) that
dynamically allocates memory for
a 2D array of integers using a double pointer and returns the pointer to
the array.*/
#include<stdio.h>
```

```c
#include<stdlib.h>
int** create2DArray(int rows, int cols);
void main()
{
    int row,col;
    int **m;
    printf("Enter the rows and columns\n");
    scanf("%d %d",&row,&col);
    m=create2DArray(row,col);
    printf("Address of first elemnt is %p",m);
}
int** create2DArray(int rows, int cols)
{
    int **matrix=(int **)malloc(rows*sizeof(int *));
    for(int i=0;i<rows;i++)
    {
        matrix[i]=(int *)malloc(cols*sizeof(int));
    }
    return matrix;

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if (
Enter the rows and columns
3 3
Address of first elemnt is 00721458
PS D:\projects\quest\C>
```

```c
/*Implement a function void free2DArray(int **arr, int rows)
that deallocates the memory allocated for a 2D array using a double
pointer*/
#include<stdio.h>
#include<stdlib.h>
void free2DArray(int **arr, int rows);
void main()
{
    int row=3;
```

```c
    int **ar=(int **)malloc(sizeof(int *));
    for(int i=0;i<row;i++)
    ar[i]=(int *)malloc(sizeof(int));
    free2DArray(ar,row);
}
void free2DArray(int **arr, int rows)
{
    for(int i=0;i<rows;i++)
    free(arr[i]);
    free(arr);
}
/*Write a function void setPointer(int **ptr) that sets the pointer passed
to it to point to a dynamically allocated integer.*/
#include<stdlib.h>
#include<stdio.h>
void setPointer(int **ptr);
void main()
{
    int **ptr;
    setPointer(ptr);

}
void setPointer(int **ptr)
{
    int *num=(int *)malloc(sizeof(int));
    *num=5;
    ptr=&num;

    printf("address of num is %p\n",ptr);
    printf("value of num is %p\n",*ptr);
    printf("valiu of address num pointing to is %d",**ptr);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; i
address of num is 0061FEEC
value of num is 00942FB8
valiu of address num pointing to is 5
PS D:\projects\quest\C>
```

```c
/*Create a function void allocateStringArray(char ***arr, int n) that
dynamically allocates memory for an array of n strings using a double
pointer.*/
#include<stdio.h>
#include<stdlib.h>
void allocateStringArray(char ***arr, int n);
void main()
{
    int n;
    char ***arr;
    printf("Enter the number of strings\n");
    scanf("%d",&n);
    allocateStringArray(arr,n);
}
void allocateStringArray(char ***arr, int n)
{
    char **ar;
    arr=&ar;
    ar=(char **)malloc(n*sizeof(char*));
    for(int i=0;i<n;i++)
    {
        ar[i]=(char *)malloc(10*sizeof(char));
    }
    printf("Enter string of length less than 10\n");
    for(int i=0;i<n;i++)
    scanf("%s",ar[i]);
    printf("Strings are\n");
    for(int i=0;i<n;i++)
    printf("%s\n",ar[i]);
    for(int i=0;i<n;i++)
```

```
    free(ar[i]);
    free(ar);
}
```

```
Enter the number of strings
3
Enter string of length less than 10
hello
world
how
Strings are
hello
world
how
PS D:\projects\quest\C>
```

```c
/*Implement a function void modifyStringArray(char **arr, int n)
that modifies each string in an array of strings using a double pointer.*/
#include<stdio.h>
#include<stdlib.h>
void modifyStringArray(char **arr, int n);
void main()
{
    int n;
    char **arr;
    printf("Enter the number of strings\n");
    scanf("%d",&n);
    arr=(char **)malloc(n*sizeof(char*));
    for(int i=0;i<n;i++)
    arr[i]=(char *)malloc(10*sizeof(char));
    printf("Enter string of length less than 10\n");
    for(int i=0;i<n;i++)
    scanf("%s",arr[i]);
    printf("Strings are\n");
    for(int i=0;i<n;i++)
    printf("%s\n",arr[i]);
    modifyStringArray(arr,n);
    printf("After modification\n");
```

```
    for(int i=0;i<n;i++)
    printf("%s\n",arr[i]);
}
void modifyStringArray(char **arr, int n)
{
    printf("Enter new string\n");
    for(int i=0;i<n;i++)
    {
        scanf("%s",arr[i]);
    }
}
```

```
PS D:\projects\quest\C> cd "d:\proje
Enter the number of strings
3
Enter string of length less than 10
hello
how
are
Strings are
hello
how
are
Enter new string
hi
where
you
After modification
hi
where
you
PS D:\projects\quest\C>
```

```
/*Write a program that declares a function pointer for a function
int add(int, int) and uses it to call the function and print the result.*/
#include<stdio.h>
int add(int, int);
```

```c
void main()
{
    int (*sum)(int,int);
    sum=&add;
    int a,b;
    printf("Enter the numbers\n");
    scanf("%d %d",&a,&b);
    printf("sum is %d",sum(a,b));
}
int add(int a, int b)
{
    return a+b;
}
```

```
PS D:\projects\quest\C> cd "d:
Enter the numbers
2 3
sum is 5
PS D:\projects\quest\C>
```

```c
/*Implement a function void performOperation(int (*operation)(int, int),
int a, int b) that takes a
function pointer as an argument and applies it to two integers, printing
the result.*/
#include<stdio.h>
void performOperation(int (*operation)(int, int), int a, int b);
int add(int a, int b);
void main()
{
    int a,b;
    printf("Enter the numbers\n");
    scanf("%d %d",&a,&b);
    performOperation(add,a,b);
}
void performOperation(int (*operation)(int , int ), int a, int b)
{

printf("The result is %d",operation(a,b));
}
```

```c
int add(int a, int b)
{
    return a+b;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter the numbers
3 4
The result is 7
PS D:\projects\quest\C>
```

```c
/*Write a program with a function int* max(int *a, int *b) that returns a
pointer to
the larger of two integers, and use a function pointer to call this
function.*/
#include<stdio.h>
int* max(int *a, int *b);
void main()
{
    int x,y;
    int *a=&x,*b=&y;
    int * (*m)(int *,int *);
    m=&max;
    printf("Enter the two numbers\n");
    scanf("%d %d",&x,&y);
    int *result;
    result=m(a,b);
    printf("Largest number is %d",*result);

}
int* max(int *a, int *b)
{
    if(*a>*b)
    return a;
    else
```

```
    return b;
}
```

```
PS D:\projects\quest\C> cd "d:\p
Enter the two numbers
5 6
Largest number is 6
PS D:\projects\quest\C>
```

```c
/*Create a program that defines two functions int add(int, int) and int
multiply(int, int) and uses a
function pointer to dynamically switch between these functions based on
user input.*/
#include<stdio.h>
int add(int, int);
int multiply(int, int);
void main()
{
    int (*ptr)(int,int);
    int a,b,c;

    do
    {
        printf("Enter the option(1.add,2.multiply,3.exit)\n");
    scanf("%d",&c);
        switch(c)
    {
        case 1:
        ptr=&add;
        printf("Enter the two numbers\n");
        scanf("%d %d",&a,&b);
        printf("Sum is %d\n",ptr(a,b));
        break;
        case 2:
         ptr=&multiply;
        printf("Enter the two numbers\n");
        scanf("%d %d",&a,&b);
        printf("product is %d\n",ptr(a,b));
        break;
```

```
        case 3:
        printf("EXITING.......\n");
        break;
        default:printf("Enter valid input\n");
        break;
    }
    }while (c!=3);
}
int add(int a, int b)
{
    return a+b;
}
int multiply(int a, int b)
{
    return a*b;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C
Enter the option(1.add,2.multiply,3.exit)
1
Enter the two numbers
5 3
Sum is 8
Enter the option(1.add,2.multiply,3.exit)
2
Enter the two numbers
4 4
product is 16
Enter the option(1.add,2.multiply,3.exit)
3
EXITING.......
PS D:\projects\quest\C>
```

```
/*Implement a program that creates an array of function pointers for basic
arithmetic operations
(addition, subtraction, multiplication, division) and allows the user to
select and execute one operation.*/
#include<stdio.h>
```

```c
int add(int a,int b)
{
    return a+b;
}
int sub(int a,int b)
{
    return a-b;
}
int mult(int a,int b)
{
    return a*b;
}
int div(int a,int b)
{
    if(b==0)
    {
        printf("Division by zero not possible\n");
        return 0;
    }
    return a/b;
}
void main()
{
    int (*ptr[])(int,int)={add,sub,mult,div};
    int c,a,b;
    do
    {
        printf("Enter the
option(1.add,2.subtract,3.multiply,4.divide,5.exit)\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Sum is %d\n",ptr[0](a,b));
            break;
            case 2:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Difference is %d\n",ptr[1](a,b));
            break;
```

```
            case 3:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Product is %d\n",ptr[2](a,b));
            break;
            case 4:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Quotient is %d\n",ptr[3](a,b));
            break;
            case 5:printf("EXIT.....");
            break;
            default:
            printf("Enter a valid input\n");
            break;
        }
    } while (c!=5);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gc
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
1
Enter the two numbers
2 2
Sum is 4
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
2
Enter the two numbers
4 3
Difference is 1
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
3
Enter the two numbers
6 7
Product is 42
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
4
Enter the two numbers
8  2
Quotient is 4
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
5
EXIT.....
PS D:\projects\quest\C>
```

```c
/*Write a function void sort(int *arr, int size, int (*compare)(int, int))
that uses a function
 pointer to compare elements, allowing for both ascending and descending
order sorting.*/
#include <stdio.h>
int compareAscending(int a, int b)
{
    return a > b;
}
int compareDescending(int a, int b)
{
```

```
    return a < b;
}
void sort(int *arr, int size, int (*compare)(int, int))
{
    for (int i = 0; i < size - 1; i++)
    {
        for (int j = 0; j < size - i - 1; j++)
        {
            if (compare(arr[j], arr[j + 1]))
            {
                int temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
void main()
{
    int arr[6] = {5, 2, 9, 1, 5, 6};
    int size = 6;
    printf("Original array:\n");
    for(int i=0;i<size;i++)
    printf("%d ",arr[i]);
    printf("\nSorting in ascending order:\n");
    sort(arr, size, compareAscending);
    for(int i=0;i<size;i++)
    printf("%d ",arr[i]);
    printf("\nSorting in descending order:\n");
    sort(arr, size, compareDescending);
    for(int i=0;i<size;i++)
    printf("%d ",arr[i]);
}
```

```
PS D:\projects\quest\C> cd "d:\projects\q
Original array:
5 2 9 1 5 6
Sorting in ascending order:
1 2 5 5 6 9
Sorting in descending order:
9 6 5 5 2 1
PS D:\projects\quest\C>
```

```c
/*Create a program with a function void execute(int x, int
(*callback)(int)) that applies a callback
function to an integer and prints the result. Demonstrate with multiple
callback functions (e.g., square, cube)*/
#include <stdio.h>
int square(int a);
int cube(int a);
void execute(int a, int (*callback)(int));
void main() {
    int num;
    printf("Enter an integer\n");
    scanf("%d", &num);
    printf("Executing square function\n");
    execute(num, square);
    printf("Executing cube function\n");
    execute(num, cube);
}
void execute(int a, int (*callback)(int)) {
    printf("Result: %d\n", callback(a));
}
int square(int a)
{
    return a*a;
}
int cube(int a)
{
    return a*a*a;
}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\"
Enter an integer
4
Executing square function
Result: 16
Executing cube function
Result: 64
PS D:\projects\quest\C>
```

```c
/*Implement a simple menu system where each menu option corresponds to a
different function,
and a function pointer array is used to call the selected function based
on user input.*/
#include<stdio.h>
int add(int a,int b)
{
    return a+b;
}
int sub(int a,int b)
{
    return a-b;
}
int mult(int a,int b)
{
    return a*b;
}
int div(int a,int b)
{
    if(b==0)
    {
        printf("Division by zero not possible\n");
        return 0;
    }
    return a/b;
```

```
}
void main()
{
    int (*ptr[])(int,int)={add,sub,mult,div};
    int c,a,b;
    do
    {
        printf("Enter the
option(1.add,2.subtract,3.multiply,4.divide,5.exit)\n");
        scanf("%d",&c);
        switch(c)
        {
            case 1:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Sum is %d\n",ptr[0](a,b));
            break;
            case 2:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Difference is %d\n",ptr[1](a,b));
            break;
            case 3:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Product is %d\n",ptr[2](a,b));
            break;
            case 4:printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Quotient is %d\n",ptr[3](a,b));
            break;
            case 5:printf("EXIT.....");
            break;
            default:
            printf("Enter a valid input\n");
            break;
        }
    } while (c!=5);

}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc t
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
1
Enter the two numbers
8 9
Sum is 17
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
2
Enter the two numbers
21 3
Difference is 18
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
3
Enter the two numbers
6 9
Product is 54
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
4
Enter the two numbers
12 3
Quotient is 4
Enter the option(1.add,2.subtract,3.multiply,4.divide,5.exit)
5
EXIT.....
PS D:\projects\quest\C>
```

```c
/*Write a program where the user inputs an operation symbol (+, -, *, /)
and the program uses a function pointer to call the corresponding
function.*/
#include<stdio.h>
int add(int a,int b)
{
    return a+b;
}
int sub(int a,int b)
{
    return a-b;
```

```
}
int mult(int a,int b)
{
    return a*b;
}
int div(int a,int b)
{
    if(b==0)
    {
        printf("Division by zero not possible\n");
        return 0;
    }
    return a/b;
}
void main()
{
    int (*ptr[])(int,int)={add,sub,mult,div};
    int a,b;
    char c;

        printf("Enter the
option(1.+->add,2.-->subtract,3.*->multiply,4./->divide,5.e->exit)\n");
        scanf("%c",&c);
        switch(c)
        {
            case '+':printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Sum is %d\n",ptr[0](a,b));
            break;
            case '-':printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Difference is %d\n",ptr[1](a,b));
            break;
            case '*':printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Product is %d\n",ptr[2](a,b));
            break;
            case '/':printf("Enter the two numbers\n");
            scanf("%d %d",&a,&b);
            printf("Quotient is %d\n",ptr[3](a,b));
```

```
            break;
            default:
            printf("Enter a valid input\n");
            break;
        }


}
```

```
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRun
Enter the option(1.+->add,2.-->subtract,3.*->multiply,4./->divide,5.e->exit)
+
Enter the two numbers
2 3
Sum is 5
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRun
Enter the option(1.+->add,2.-->subtract,3.*->multiply,4./->divide,5.e->exit)
-
Enter the two numbers
5 2
Difference is 3
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRun
Enter the option(1.+->add,2.-->subtract,3.*->multiply,4./->divide,5.e->exit)
*
Enter the two numbers
4 4
Product is 16
PS D:\projects\quest\C> cd "d:\projects\quest\C\" ; if ($?) { gcc tempCodeRun
Enter the option(1.+->add,2.-->subtract,3.*->multiply,4./->divide,5.e->exit)
/
Enter the two numbers
4 2
Quotient is 2
PS D:\projects\quest\C>
PS D:\projects\quest\C>
```

```
/*Design a simple state machine where each state is represented by a
function, and transitions are handled using function pointers.
 For example, implement a traffic light system with states like Red,
Green, and Yellow*/
 #include <stdio.h>
#include <stdlib.h>
```

```c
#include <unistd.h>
void redState(void);
void greenState(void);
void yellowState(void);
void (*currentState)(void) = NULL;
int main() {
    currentState = redState;
    while (1) {
        currentState();
    }
}
void redState(void) {
    printf("RED: Stop!\n");
    sleep(3);
    currentState = greenState;
}
void greenState(void) {
    printf("GREEN: Go!\n");
    sleep(3);
    currentState = yellowState;
}
void yellowState(void) {
    printf("YELLOW: Get Ready to Stop!\n");
    sleep(2);
    currentState = redState;
}
```

```
PS D:\projects\quest\C> cd "d:\proje
RED: Stop!
GREEN: Go!
YELLOW: Get Ready to Stop!
RED: Stop!
GREEN: Go!
YELLOW: Get Ready to Stop!
PS D:\projects\quest\C>
```