E01

Given a string, S , consisting of alphabets and digits, find the number of alphabets and digits in the given string.

Test Case 1:

Sample Input: a11472o5t6

Number of digits: 7

Number of alphabets:3

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

int main() {
    char str[100];
    int alphabets = 0, digits = 0;

    // Input string from user
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

    // Iterate through each character in the string
    for (int i = 0; str[i] != '\0'; ++i) {
        // Check if the character is an alphabet
        if (isalpha(str[i])) {
            alphabets++;
        }
        // Check if the character is a digit
        else if (isdigit(str[i])) {
            digits++;
        }
    }

    // Display the results
    printf("Number of alphabets: %d\n", alphabets);
    printf("Number of digits: %d\n", digits);

    return 0;
}
```

E02

Given a string s containing just the characters like:

'(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if number of opening and closing brackets is same (irrespective of the sequence of opening and closing brackets)

Test Case 1:
Sample Input: ( )
Sample Output: Valid

Test Case 2:
Sample Input: ( )[ ]{ }
Sample Output: Valid

Test Case 3:
Sample Input: ([{ }
Sample Output: Invalid

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

#define MAX_SIZE 100

// Structure for stack
struct Stack {
    int top;
    char items[MAX_SIZE];
};

// Function to initialize the stack
void initialize(struct Stack *stack) {
    stack->top = -1;
}

// Function to check if the stack is empty
bool isEmpty(struct Stack *stack) {
    return (stack->top == -1);
}

// Function to push an item onto the stack
void push(struct Stack *stack, char item) {
    if (stack->top == MAX_SIZE - 1) {
        printf("Stack Overflow\n");
        return;
    }
    stack->items[++stack->top] = item;
}

// Function to pop an item from the stack
char pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stack Underflow\n");
        return '\0';
    }
    return stack->items[stack->top--];
}

// Function to check if the given string is valid
bool isValid(char *s) {
    struct Stack stack;
    initialize(&stack);

    // Iterate through each character in the string
    for (int i = 0; s[i] != '\0'; i++) {
        // If the character is an opening bracket, push it onto the stack
        if (s[i] == '(' || s[i] == '{' || s[i] == '[') {
            push(&stack, s[i]);
        }
        // If the character is a closing bracket, check if it matches the top of the stack
        else if (s[i] == ')' || s[i] == '}' || s[i] == ']') {
            if (isEmpty(&stack)) {
                return false; // Unmatched closing bracket
            }
            char top = pop(&stack);
            if ((s[i] == ')' && top != '(') || (s[i] == '}' && top != '{') || (s[i] == ']' &&
top != '[')) {
                return false; // Mismatched opening and closing brackets
            }
        }
    }

    // Check if the stack is empty after processing the string
    return isEmpty(&stack);
}

int main() {
    char input[100];

    // Input string from user
    printf("Enter a string: ");
    fgets(input, sizeof(input), stdin);

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

    // Check if the string is valid
    if (isValid(input)) {
        printf("The given string is valid.\n");
    } else {
        printf("The given string is not valid.\n");
    }

    return 0;
}
```

E03

Given a string s, perform following operations on a string using function (Without pointer):

1. Find a length of a string.
2. Print string in reverse order.
3. Copy string s into s1 and print it.
4. Accept another string, say s2. Concatenate s and s2 and print concatenated string.

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

// Function to find the length of a string
int findLength(char s[]) {
    int length = 0;
    while (s[length] != '\0') {
        length++;
    }
    return length;
}

// Function to print the string in reverse order
void printReverse(char s[]) {
    int length = findLength(s);
    for (int i = length - 1; i >= 0; i--) {
        printf("%c", s[i]);
    }
    printf("\n");
}

// Function to copy string s into s1 and print it
void copyAndPrint(char s[], char s1[]) {
    strcpy(s1, s);
    printf("Copied string: %s\n", s1);
}

// Function to concatenate s and s2 and print the concatenated string
void concatenateAndPrint(char s[], char s2[]) {
    strcat(s, s2);
    printf("Concatenated string: %s\n", s);
}

int main() {
    char s[100], s1[100], s2[100];

    // Input string from user
    printf("Enter a string: ");
    fgets(s, sizeof(s), stdin);

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

    // Find and print the length of the string
    int length = findLength(s);
    printf("Length of the string: %d\n", length);

    // Print the string in reverse order
    printf("String in reverse order: ");
    printReverse(s);

    // Copy string s into s1 and print it
    copyAndPrint(s, s1);

    // Accept another string s2
    printf("Enter another string (s2): ");
    fgets(s2, sizeof(s2), stdin);
    s2[strcspn(s2, "\n")] = '\0'; // Remove newline character

    // Concatenate s and s2 and print the concatenated string
    concatenateAndPrint(s, s2);

    return 0;
}

```

E04

Given a string s, perform following operations on a string using function (With pointer):

1. Find a length of a string.
2. Print string in reverse order.
3. Copy string s into s1 and print it.

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

// Function to find the length of a string using pointers
int findLength(char *s) {
    int length = 0;
    while (*s != '\0') {
        length++;
        s++;
    }
    return length;
}

// Function to print the string in reverse order using pointers
void printReverse(char *s) {
    int length = findLength(s);
    for (int i = length - 1; i >= 0; i--) {
        printf("%c", *(s + i));
    }
    printf("\n");
}

```

```c
// Function to copy string s into s1 using pointers and print it
void copyAndPrint(char *s, char *s1) {
    while (*s != '\0') {
        *s1 = *s;
        s++;
        s1++;
    }
    *s1 = '\0'; // Null-terminate the destination string
    printf("Copied string: %s\n", s1);
}

int main() {
    char s[100], s1[100];

    // Input string from user
    printf("Enter a string: ");
    fgets(s, sizeof(s), stdin);

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

    // Find and print the length of the string using pointers
    int length = findLength(s);
    printf("Length of the string: %d\n", length);

    // Print the string in reverse order using pointers
    printf("String in reverse order: ");
    printReverse(s);

    // Copy string s into s1 using pointers and print it
    copyAndPrint(s, s1);

    return 0;
}

```

E05

Given a string s, return true if it a **Palindrome** or false otherwise.

Test Cases:

Case 1:

Input: MADAM

Output: true

Case 2:

Input: CAR

Output: false

```c
// C program to check whether a number is palindrome or not
#include <stdio.h>

// Driver code
int main()
{
    // This is our given number
    int original_number = 12321;

    // This variable stored reversed digit
    int reversed = 0;

    int num = original_number;

    // Execute a while loop to reverse
    // digits of given number
    while (num != 0) {
        int r = num % 10;
        reversed = reversed * 10 + r;
        num /= 10;
    }

    // Compare original_number with
    // reversed number
    if (original_number == reversed) {
        printf(" Given number %d is a palindrome number",
            original_number);
    }
    else {
        printf(
            " Given number %d is not a palindrome number",
            original_number);
    }

    return 0;
}
```

Accept limit of array from user. As per the limit, read integer elements in array. Then print :

1. Minimum, Maximum number from array.
2. Search for a particular number from array.

```c
#include <stdio.h>

// Function to find the minimum and maximum numbers in the array
void findMinMax(int arr[], int size, int *min, int *max) {
    *min = arr[0];
    *max = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] < *min) {
            *min = arr[i];
        }

        if (arr[i] > *max) {
            *max = arr[i];
        }
    }
}

// Function to search for a particular number in the array
int searchNumber(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        if (arr[i] == target) {
            return i; // Return the index if the number is found
        }
    }

    return -1; // Return -1 if the number is not found
}

int main() {
    int limit;

    // Input limit from the user
    printf("Enter the limit of the array: ");
    scanf("%d", &limit);

    // Declare an array of the specified limit
    int arr[limit];

    // Input array elements from the user
    printf("Enter %d integer elements:\n", limit);
    for (int i = 0; i < limit; i++) {
        scanf("%d", &arr[i]);
    }

    // Find and print the minimum and maximum numbers in the array
    int min, max;
    findMinMax(arr, limit, &min, &max);
    printf("Minimum number: %d\n", min);
    printf("Maximum number: %d\n", max);

    // Search for a particular number in the array
    int target;
    printf("Enter the number to search: ");
    scanf("%d", &target);

    int index = searchNumber(arr, limit, target);
    if (index != -1) {
        printf("Number %d found at index %d in the array.\n", target, index);
    } else {
        printf("Number %d not found in the array.\n", target);
    }

    return 0;
}

```

E07

You are given a string s and an integer array index of the same length. The string s will be shuffled such that the character at the ith position moves to indices[i] in the shuffled string.

Return *the shuffled string*.

Test Cases:
Input: s = "codeleet", indices = [4,5,6,7,0,2,1,3]
Output: "leetcode"
Explanation: As shown, "codeleet" becomes "leetcode" after shuffling.

Input: s = "abc", indices = [0,1,2]
Output: "abc"
Explanation: After shuffling, each character remains in its position.

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

// Function to shuffle the string based on the given indices
char* shuffleString(char* s, int* indices, int size) {
    char* shuffled = (char*)malloc((size + 1) * sizeof(char)); // +1 for the null terminator
    if (shuffled == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }

    for (int i = 0; i < size; i++) {
        shuffled[indices[i]] = s[i];
    }

    shuffled[size] = '\0'; // Null-terminate the shuffled string

    return shuffled;
}

int main() {
    char s[100];
    int indices[100];
    int size;

    // Input string from user
    printf("Enter the string: ");
    fgets(s, sizeof(s), stdin);
    s[strcspn(s, "\n")] = '\0'; // Remove newline character

    // Input array of indices from user
    printf("Enter the array of indices (space-separated): ");
    for (int i = 0; i < size; i++) {
        scanf("%d", &indices[i]);
    }

    // Get the size of the string and indices array
    size = strlen(s);

    // Check if the size of the string and indices array match
    if (size != sizeof(indices) / sizeof(indices[0])) {
        printf("Size of the string and indices array should be the same.\n");
        return 1;
    }

    // Call the function to shuffle the string based on indices
    char* shuffledString = shuffleString(s, indices, size);

    // Print the shuffled string
    printf("Shuffled String: %s\n", shuffledString);

    // Free the allocated memory
    free(shuffledString);

    return 0;
}
```

E08

1. Write a Program to print the output like:

```
A
A B
A B C
A B C D
A B C D E
A B C D
A B C
A B
A
```

```c
#include <stdio.h>

int main() {
    int n = 5; // You can change the value of n to adjust the number of rows

    // Loop to print the upper half of the pattern
    for (int i = 1; i <= n; i++) {
        for (char ch = 'A'; ch <= 'A' + i - 1; ch++) {
            printf("%c ", ch);
        }
        printf("\n");
    }

    // Loop to print the lower half of the pattern
    for (int i = n - 1; i >= 1; i--) {
        for (char ch = 'A'; ch <= 'A' + i - 1; ch++) {
            printf("%c ", ch);
        }
        printf("\n");
    }

    return 0;
}
```

2. Write a program to print factorial of 1 to 10 numbers.

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

```c

// Function to calculate the factorial of a number
unsigned long long factorial(int num) {
    if (num == 0 || num == 1) {
        return 1;
    } else {
        return num * factorial(num - 1);
    }
}

int main() {
    // Loop to calculate and print factorial for numbers 1 to 10
    for (int i = 1; i <= 10; i++) {
        printf("Factorial of %d: %llu\n", i, factorial(i));
    }

    return 0;
}

```

E09

1. Write a Program to print the output like:

```
1
1 2
1 2 3
1 2 3 4 5
E  D  C  B  A
E  D  C  B
E  D  C
E  D
E
```

```c
#include <stdio.h>

int main() {
    int n = 5; // You can change the value of n to adjust the number of rows

    // Loop to print the upper half of the pattern
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            printf("%d ", j);
        }
        printf("\n");
    }

    // Loop to print the lower half of the pattern
    for (int i = n; i >= 1; i--) {
        for (char ch = 'A' + i - 1; ch >= 'A'; ch--) {
            printf("%c ", ch);
        }
        printf("\n");
    }

    return 0;
}
```

2. Write a program to print prime numbers between two numbers given by user.

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

// Function to check if a number is prime
bool isPrime(int num) {
    if (num <= 1) {
        return false;
    }

    for (int i = 2; i * i <= num; i++) {
        if (num % i == 0) {
            return false; // Not a prime number
        }
    }

    return true; // Prime number
}

int main() {
    int start, end;

    // Input range from user
    printf("Enter the starting number: ");
    scanf("%d", &start);

    printf("Enter the ending number: ");
    scanf("%d", &end);

    // Validate the range
    if (start >= end) {
        printf("Invalid range. Ending number should be greater than the starting number.\n");
        return 1;
    }

    // Print prime numbers between the given range
    printf("Prime numbers between %d and %d are:\n", start, end);
    for (int i = start; i <= end; i++) {
        if (isPrime(i)) {
            printf("%d\n", i);
        }
    }

    return 0;
}

```

E10

Write a C program to find the frequency of each character in a string.

Test Cases: String: This book is very good

Frequency of T:1

Frequency of h:0

Frequency of o:4 and so on for every distinct character.

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

#define MAX_SIZE 100

// Function to find the frequency of each character in a string
```

```c
void findFrequency(char *str) {
    int count[256] = {0}; // Assuming ASCII characters

    // Iterate through each character in the string
    for (int i = 0; str[i] != '\0'; i++) {
        char currentChar = str[i];

        // Increment the frequency count for the current character
        count[currentChar]++;
    }

    // Print the frequency of each character
    for (int i = 0; i < 256; i++) {
        if (count[i] > 0) {
            printf("Frequency of %c: %d\n", i, count[i]);
        }
    }
}

int main() {
    char str[MAX_SIZE];

    // Input string from the user
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

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

    // Find and print the frequency of each character in the string
    findFrequency(str);

    return 0;
}
```

E11

Write a C program to print Fibonacci series up to n terms.

Test Cases: No. of terms 10

Output: 0,1,1,2,3,5,8,13,21,34

```c
#include <stdio.h>

// Function to print Fibonacci series up to n terms
void printFibonacci(int n) {
    int first = 0, second = 1, next;

    printf("Fibonacci series up to %d terms:\n", n);

    for (int i = 0; i < n; i++) {
        printf("%d, ", first);

        next = first + second;
        first = second;
        second = next;
    }
    printf("\n");
}

int main() {
    int n;

    // Input the number of terms from the user
    printf("Enter the number of terms for Fibonacci series: ");
    scanf("%d", &n);

    // Print the Fibonacci series
    printFibonacci(n);

    return 0;
}
```

E12

Write a C program to count the number of Vowels and Consonants.

Test Cases: String: C is a programming language.

Vowels: 9

Consonants: 14

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

// Function to check if a character is a vowel
int isVowel(char ch) {
    ch = toupper(ch); // Convert to uppercase for case-insensitive check
    return (ch == 'A' || ch == 'E' || ch == 'I' || ch == 'O' || ch == 'U');
}

// Function to count the number of vowels and consonants in a string
void countVowelsAndConsonants(char *str, int *vowels, int *consonants) {
    *vowels = 0;
    *consonants = 0;

    // Iterate through each character in the string
    for (int i = 0; str[i] != '\0'; i++) {
        char currentChar = str[i];

        // Check if the character is an alphabet
        if (isalpha(currentChar)) {
            // Check if the alphabet is a vowel or a consonant
            if (isVowel(currentChar)) {
                (*vowels)++;
            } else {
                (*consonants)++;
            }
        }
    }
}

int main() {
    char str[100];
    int vowels, consonants;

    // Input string from the user
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);

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

    // Call the function to count vowels and consonants
    countVowelsAndConsonants(str, &vowels, &consonants);

    // Print the results
    printf("Number of vowels: %d\n", vowels);
    printf("Number of consonants: %d\n", consonants);

    return 0;
}
```

E13

Write a program to convert decimal to binary number.

Test Cases:

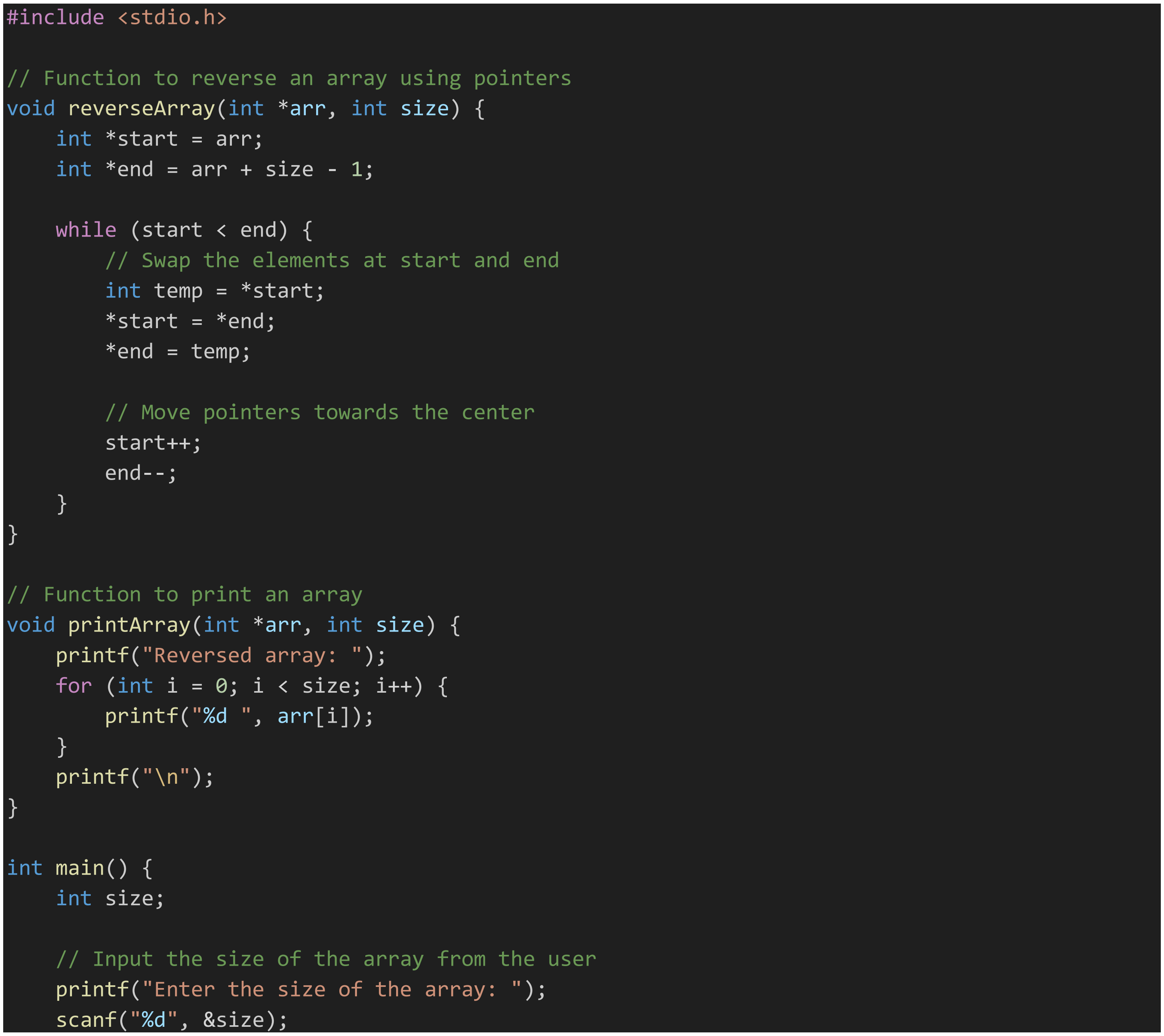
Input : 5

Output: 101

```c
#include <stdio.h>

// Function to convert decimal to binary
void decimalToBinary(int decimal) {
    int binary[32];
    int index = 0;

    // Convert decimal to binary representation
    while (decimal > 0) {
        binary[index] = decimal % 2;
        decimal = decimal / 2;
        index++;
    }

    // Print the binary representation in reverse order
    printf("Binary representation: ");
    for (int i = index - 1; i >= 0; i--) {
        printf("%d", binary[i]);
    }
    printf("\n");
}

int main() {
    int decimal;

    // Input decimal number from the user
    printf("Enter a decimal number: ");
    scanf("%d", &decimal);

    // Check if the entered number is non-negative
    if (decimal < 0) {
        printf("Please enter a non-negative decimal number.\n");
        return 1;
    }

    // Call the function to convert decimal to binary
    decimalToBinary(decimal);

    return 0;
}
```

E14

Write a C program to reverse an array using pointers.



```c
#include <stdio.h>

// Function to reverse an array using pointers
void reverseArray(int *arr, int size) {
    int *start = arr;
    int *end = arr + size - 1;

    while (start < end) {
        // Swap the elements at start and end
        int temp = *start;
        *start = *end;
        *end = temp;

        // Move pointers towards the center
        start++;
        end--;
    }
}

// Function to print an array
void printArray(int *arr, int size) {
    printf("Reversed array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size;

    // Input the size of the array from the user
    printf("Enter the size of the array: ");
    scanf("%d", &size);
    // Check if the size is non-negative
    if (size <= 0) {
        printf("Please enter a valid size for the array.\n");
        return 1;
    }

    // Declare an array of the specified size
    int arr[size];

    // Input array elements from the user
    printf("Enter %d elements for the array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr[i]);
    }

    // Call the function to reverse the array using pointers
    reverseArray(arr, size);

    // Call the function to print the reversed array
    printArray(arr, size);

    return 0;
}
```

E15

Write a function to print all perfect numbers in a given interval in C programming.

Test Cases:1) Enter lower limit to print perfect numbers:1

Enter Upper limit to print perfect numbers:10000

All perfect numbers between 1 to 10000 are:

6,28,496,8128

```
2) Enter lower limit to print perfect numbers: 23

Enter upper limit to print perfect numbers: 450

All perfect numbers between 23 to 450 are:

28

3) Enter lower limit to print perfect numbers: 15

Enter upper limit to print perfect numbers: 70

All perfect numbers between 15 to 70 are: 28
```

```c
#include <stdio.h>

// Function to check if a number is perfect
int isPerfect(int num) {
    int sum = 0;

    // Find the proper divisors and sum them
    for (int i = 1; i <= num / 2; i++) {
        if (num % i == 0) {
            sum += i;
        }
    }

    // Check if the sum of divisors equals the number
    return (sum == num);
}

// Function to print all perfect numbers in a given interval
void printPerfectNumbers(int start, int end) {
    printf("Perfect numbers in the interval [%d, %d]:\n", start, end);

    for (int i = start; i <= end; i++) {
        if (isPerfect(i)) {
            printf("%d\n", i);
        }
    }
}

int main() {
    int start, end;

    // Input interval from the user
    printf("Enter the starting number of the interval: ");
    scanf("%d", &start);

    printf("Enter the ending number of the interval: ");
    scanf("%d", &end);

    // Validate the interval
    if (start < 1 || end <= start) {
        printf("Invalid interval. Please enter a valid range.\n");
        return 1;
    }

    // Call the function to print perfect numbers in the interval
    printPerfectNumbers(start, end);

    return 0;
}
```

E16

Write a C Program to read and print name and other details like mobile number, marks of 5 subjects of n number of students using Structure. Print data of top 5 students (top 5 should be calculated based on the entered marks)

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

#define MAX_STUDENTS 100

// Define a structure to represent student details
struct Student {
    char name[50];
    long long mobileNumber;
    float marks[5];
    float totalMarks;
};

// Function to input details for 'n' students
void inputStudentDetails(struct Student students[], int n) {
    for (int i = 0; i < n; i++) {
        printf("Enter details for student %d:\n", i + 1);

        // Input student name
        printf("Name: ");
        scanf("%s", students[i].name);

        // Input mobile number
        printf("Mobile Number: ");
        scanf("%lld", &students[i].mobileNumber);

        // Input marks for 5 subjects
        printf("Enter marks for 5 subjects:\n");
        students[i].totalMarks = 0;
        for (int j = 0; j < 5; j++) {
            printf("Subject %d: ", j + 1);
            scanf("%f", &students[i].marks[j]);
            students[i].totalMarks += students[i].marks[j];
        }
    }
}

// Function to print details of a student
void printStudentDetails(struct Student student) {
    printf("Name: %s\n", student.name);
    printf("Mobile Number: %lld\n", student.mobileNumber);
    printf("Total Marks: %.2f\n", student.totalMarks);
    printf("\n");
}

// Function to print top 5 students based on marks
void printTop5Students(struct Student students[], int n) {
    // Sort students based on total marks
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (students[j].totalMarks < students[j + 1].totalMarks) {
                // Swap the students
                struct Student temp = students[j];
                students[j] = students[j + 1];
                students[j + 1] = temp;
            }
        }
    }

    // Print details of the top 5 students
    printf("Top 5 Students:\n");
    for (int i = 0; i < 5 && i < n; i++) {
        printf("Rank %d:\n", i + 1);
        printStudentDetails(students[i]);
    }
}

int main() {
    int n;

    // Input the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Check if the number of students is within limits
    if (n <= 0 || n > MAX_STUDENTS) {
        printf("Invalid number of students. Please enter a valid number.\n");
        return 1;
    }

    // Declare an array of structures to store student details
    struct Student students[MAX_STUDENTS];

    // Input details for 'n' students
    inputStudentDetails(students, n);

    // Print details of all students
    printf("Details of all students:\n");
    for (int i = 0; i < n; i++) {
        printf("Student %d:\n", i + 1);
        printStudentDetails(students[i]);
    }

    // Print details of top 5 students based on marks
    printTop5Students(students, n);

    return 0;
}
```

E18

Write a C program to store student data(roll no, name, marks of 5 subjects) in structure. Then calculate the grade of a student using a logical operator.

(76-99 Distinction

60-75 First Class

50-59 Second Class

40-49 Pass Class

Below 40 Fail)

```c
#include <stdio.h>

// Define a structure to represent student details
struct Student {
    int rollNo;
    char name[50];
    float marks[5];
    float totalMarks;
    char grade;
};

// Function to calculate the grade based on total marks
char calculateGrade(float totalMarks) {
    if (totalMarks >= 76 && totalMarks <= 99) {
        return 'A'; // Distinction
    } else if (totalMarks >= 60 && totalMarks <= 75) {
        return 'B'; // First Class
    } else if (totalMarks >= 50 && totalMarks <= 59) {
        return 'C'; // Second Class
    } else if (totalMarks >= 40 && totalMarks <= 49) {
        return 'D'; // Pass Class
    } else {
        return 'F'; // Fail
    }
}

// Function to input details for a student
void inputStudentDetails(struct Student *student) {
    printf("Enter roll number: ");
    scanf("%d", &student->rollNo);

    printf("Enter name: ");
    scanf("%s", student->name);

    printf("Enter marks for 5 subjects:\n");
    student->totalMarks = 0;
    for (int i = 0; i < 5; i++) {
        printf("Subject %d: ", i + 1);
        scanf("%f", &student->marks[i]);
        student->totalMarks += student->marks[i];
    }

    // Calculate the grade based on total marks
    student->grade = calculateGrade(student->totalMarks);
}

// Function to print details of a student
void printStudentDetails(struct Student student) {
    printf("Roll Number: %d\n", student.rollNo);
    printf("Name: %s\n", student.name);
    printf("Total Marks: %.2f\n", student.totalMarks);
    printf("Grade: %c\n", student.grade);
    printf("\n");
}

int main() {
    int n;

    // Input the number of students
    printf("Enter the number of students: ");
    scanf("%d", &n);

    // Check if the number of students is within limits
    if (n <= 0) {
        printf("Invalid number of students. Please enter a valid number.\n");
        return 1;
    }

    // Declare an array of structures to store student details
    struct Student students[n];

    // Input details for each student
    for (int i = 0; i < n; i++) {
        printf("Enter details for student %d:\n", i + 1);
        inputStudentDetails(&students[i]);
    }

    // Print details of all students
    printf("Details of all students:\n");
    for (int i = 0; i < n; i++) {
        printf("Student %d:\n", i + 1);
        printStudentDetails(students[i]);
    }

    return 0;
}
```

E19

Write a c program for swapping of two arrays using call by value function.

```c
#include <stdio.h>

// Function to swap the contents of two arrays
void swapArrays(int arr1[], int arr2[], int size) {
    for (int i = 0; i < size; i++) {
        // Swap elements of arr1 and arr2 at the same index
        int temp = arr1[i];
        arr1[i] = arr2[i];
        arr2[i] = temp;
    }
}
```

```c
// Function to print the elements of an array
void printArray(int arr[], int size) {
    printf("Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size;

    // Input the size of the arrays from the user
    printf("Enter the size of the arrays: ");
    scanf("%d", &size);

    // Check if the size is non-negative
    if (size <= 0) {
        printf("Please enter a valid size for the arrays.\n");
        return 1;
    }

    // Declare two arrays of the specified size
    int arr1[size], arr2[size];

    // Input elements for the first array
    printf("Enter elements for the first array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr1[i]);
    }

    // Input elements for the second array
    printf("Enter elements for the second array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr2[i]);
    }

    // Print the original arrays
    printf("Original Arrays:\n");
    printArray(arr1, size);
    printArray(arr2, size);

    // Call the function to swap the arrays
    swapArrays(arr1, arr2, size);

    // Print the arrays after swapping
    printf("Arrays after swapping:\n");
    printArray(arr1, size);
    printArray(arr2, size);

    return 0;
}
```

E20

Write a C program to count the number of digits in an integer. Then print addition of all digits in the given number.

Test Cases:

1) **input:** 671041

output:
Number of digits:6
Addition of digits:19

```
#include <stdio.h>

// Function to count the number of digits in an integer
int countDigits(int num) {
    int count = 0;

    while (num != 0) {
        num /= 10;
        count++;
```

```c
    }

    return count;
}

// Function to calculate the sum of digits in an integer
int sumOfDigits(int num) {
    int sum = 0;

    while (num != 0) {
        sum += num % 10;
        num /= 10;
    }

    return sum;
}

int main() {
    int number;

    // Input an integer from the user
    printf("Enter an integer: ");
    scanf("%d", &number);

    // Call the function to count the number of digits
    int digitCount = countDigits(number);

    // Call the function to calculate the sum of digits
    int digitSum = sumOfDigits(number);

    // Print the results
    printf("Number of digits: %d\n", digitCount);
    printf("Sum of digits: %d\n", digitSum);

    return 0;
}
```

E21

1. Find the sum of two one-dimensional arrays using Dynamic Memory Allocation and functions. Array should be passed as a function argument and in function should perform addition of passed arrays.

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

// Function to perform addition of two arrays
int* addArrays(const int* arr1, const int* arr2, int size) {
    int* sumArray = (int*)malloc(size * sizeof(int));

    if (sumArray == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }

    for (int i = 0; i < size; i++) {
        sumArray[i] = arr1[i] + arr2[i];
    }

    return sumArray;
}

// Function to print the elements of an array
void printArray(const int* arr, int size) {
    printf("Array: ");
    for (int i = 0; i < size; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main() {
    int size;

    // Input the size of the arrays from the user
    printf("Enter the size of the arrays: ");
    scanf("%d", &size);

    // Check if the size is non-negative
    if (size <= 0) {
        printf("Please enter a valid size for the arrays.\n");
        return 1;
    }

    // Dynamically allocate memory for two arrays
    int* arr1 = (int*)malloc(size * sizeof(int));
    int* arr2 = (int*)malloc(size * sizeof(int));

    if (arr1 == NULL || arr2 == NULL) {
        printf("Memory allocation failed.\n");
        return 1;
    }

    // Input elements for the first array
    printf("Enter elements for the first array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr1[i]);
    }

    // Input elements for the second array
    printf("Enter elements for the second array:\n");
    for (int i = 0; i < size; i++) {
        scanf("%d", &arr2[i]);
    }

    // Call the function to add the arrays
    int* sumArray = addArrays(arr1, arr2, size);

    // Print the original arrays
    printf("Original Arrays:\n");
    printArray(arr1, size);
    printArray(arr2, size);

    // Print the sum array
    printf("Sum Array:\n");
    printArray(sumArray, size);

    // Free dynamically allocated memory
    free(arr1);
    free(arr2);
    free(sumArray);

    return 0;
}

```

E22

Perform following operations on 2D Matrix:

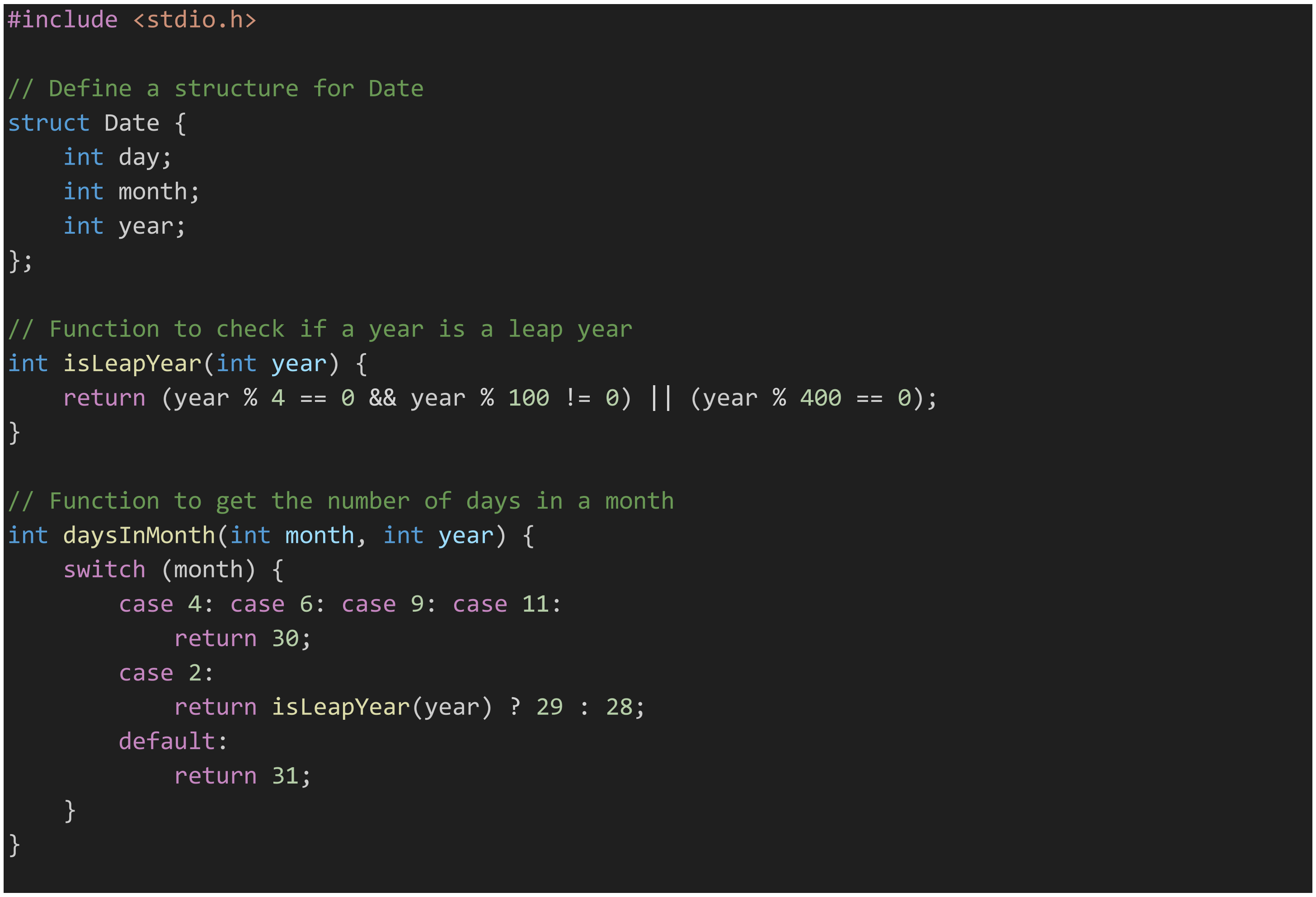
84. Accept number of rows and columns of two matrices and read elements of both matrices.
85. Print Transpose of both matrices.
86. Print Diagonal elements of both matrices.

```c
#include <stdio.h>

// Function to accept elements for a matrix
void inputMatrix(int matrix[][100], int rows, int cols) {
    printf("Enter elements for the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

// Function to print the transpose of a matrix
void printTranspose(int matrix[][100], int rows, int cols) {
    printf("Transpose of the matrix:\n");
    for (int j = 0; j < cols; j++) {
        for (int i = 0; i < rows; i++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

// Function to print the diagonal elements of a matrix
void printDiagonal(int matrix[][100], int rows, int cols) {
    printf("Diagonal elements of the matrix:\n");
    for (int i = 0; i < rows && i < cols; i++) {
        printf("%d ", matrix[i][i]);
    }
    printf("\n");
}

int main() {
    int rows, cols;

    // Input number of rows and columns for the matrices
    printf("Enter number of rows: ");
    scanf("%d", &rows);

    printf("Enter number of columns: ");
    scanf("%d", &cols);

    // Check if the dimensions are non-negative
    if (rows <= 0 || cols <= 0) {
        printf("Invalid dimensions. Please enter valid values.\n");
        return 1;
    }

    int matrix1[100][100], matrix2[100][100];

    // Input elements for the first matrix
    printf("For Matrix 1:\n");
    inputMatrix(matrix1, rows, cols);

    // Input elements for the second matrix
    printf("For Matrix 2:\n");
    inputMatrix(matrix2, rows, cols);

    // Print the transpose of both matrices
    printf("\n");
    printf("Operations on Matrix 1:\n");
    printTranspose(matrix1, rows, cols);
    printDiagonal(matrix1, rows, cols);

    printf("\n");
    printf("Operations on Matrix 2:\n");
    printTranspose(matrix2, rows, cols);
    printDiagonal(matrix2, rows, cols);

    return 0;
}

```

E23

Create a structure named Date having day, month and year as its elements.
Store the current date in the structure. Now add 45 days to the current date and display the final date.

Test Cases:
Input : dd mm yy (e.g 6 /3/23)
Output: dd/mm/yy (20/4/23)



```c
#include <stdio.h>

// Define a structure for Date
struct Date {
    int day;
    int month;
    int year;
};

// Function to check if a year is a leap year
int isLeapYear(int year) {
    return (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);
}

// Function to get the number of days in a month
int daysInMonth(int month, int year) {
    switch (month) {
        case 4: case 6: case 9: case 11:
            return 30;
        case 2:
            return isLeapYear(year) ? 29 : 28;
        default:
            return 31;
    }
}
```

```c
// Function to add days to a given date
struct Date addDays(struct Date currentDate, int daysToAdd) {
    struct Date resultDate = currentDate;

    while (daysToAdd > 0) {
        int daysInCurrentMonth = daysInMonth(resultDate.month, resultDate.year);

        // Calculate remaining days in the current month
        int remainingDaysInMonth = daysInCurrentMonth - resultDate.day + 1;

        if (daysToAdd <= remainingDaysInMonth) {
            // If remaining days are less than or equal to days to add, update the day and
break
            resultDate.day += daysToAdd;
            break;
        } else {
            // Move to the next month
            resultDate.day = 1;
            resultDate.month++;

            if (resultDate.month > 12) {
                resultDate.month = 1;
                resultDate.year++;
            }

            daysToAdd -= remainingDaysInMonth;
        }
    }

    return resultDate;
}

// Function to print a date
void printDate(struct Date date) {
    printf("%02d/%02d/%04d\n", date.day, date.month, date.year);
}

int main() {
    // Get the current date
    struct Date currentDate;
    printf("Enter the current date (DD MM YYYY): ");
    scanf("%d %d %d", &currentDate.day, &currentDate.month, &currentDate.year);

    // Add 45 days to the current date
    struct Date finalDate = addDays(currentDate, 45);

    // Print the final date
    printf("Current Date: ");
    printDate(currentDate);

    printf("Final Date after adding 45 days: ");
    printDate(finalDate);

    return 0;
}
```

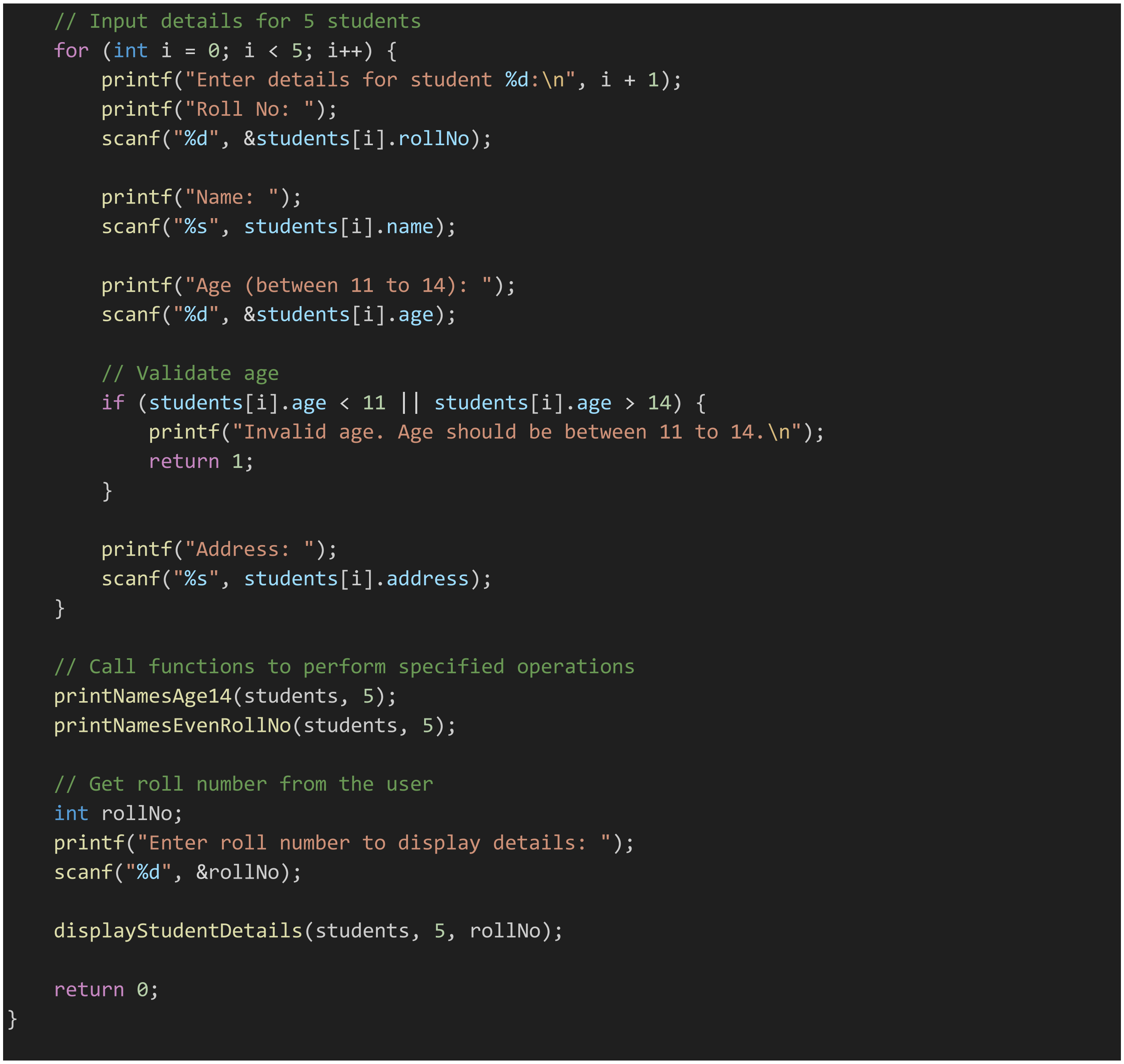
E24

Write a structure to store the roll no., name, age (between 11 to 14) and address of students (5 students). Store the information of the students.

1 - Write a function to print the names of all the students having age 14.
2 - Write a function to print the names of all the students having even roll no.
3 - Write a function to display the details of the student whose roll no is given (i.e. roll no. entered by the user).

```
#include <stdio.h>

// Define a structure for Student
struct Student {
    int rollNo;
    char name[50];
    int age;
    char address[100];
};
```

```c
// Function to print the names of students having age 14
void printNamesAge14(struct Student students[], int size) {
    printf("Names of students with age 14:\n");
    for (int i = 0; i < size; i++) {
        if (students[i].age == 14) {
            printf("%s\n", students[i].name);
        }
    }
    printf("\n");
}

// Function to print the names of students with even roll numbers
void printNamesEvenRollNo(struct Student students[], int size) {
    printf("Names of students with even roll numbers:\n");
    for (int i = 0; i < size; i++) {
        if (students[i].rollNo % 2 == 0) {
            printf("%s\n", students[i].name);
        }
    }
    printf("\n");
}

// Function to display details of a student based on roll number
void displayStudentDetails(struct Student students[], int size, int rollNo) {
    int found = 0;
    for (int i = 0; i < size; i++) {
        if (students[i].rollNo == rollNo) {
            printf("Details of student with roll number %d:\n", rollNo);
            printf("Name: %s\n", students[i].name);
            printf("Age: %d\n", students[i].age);
            printf("Address: %s\n", students[i].address);
            found = 1;
            break;
        }
    }

    if (!found) {
        printf("Student with roll number %d not found.\n", rollNo);
    }

    printf("\n");
}

int main() {
    // Declare an array of structures to store student details
    struct Student students[5];
```



```c
    // Input details for 5 students
    for (int i = 0; i < 5; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("Roll No: ");
        scanf("%d", &students[i].rollNo);

        printf("Name: ");
        scanf("%s", students[i].name);

        printf("Age (between 11 to 14): ");
        scanf("%d", &students[i].age);

        // Validate age
        if (students[i].age < 11 || students[i].age > 14) {
            printf("Invalid age. Age should be between 11 to 14.\n");
            return 1;
        }

        printf("Address: ");
        scanf("%s", students[i].address);
    }

    // Call functions to perform specified operations
    printNamesAge14(students, 5);
    printNamesEvenRollNo(students, 5);

    // Get roll number from the user
    int rollNo;
    printf("Enter roll number to display details: ");
    scanf("%d", &rollNo);

    displayStudentDetails(students, 5, rollNo);

    return 0;
}
```

E25

Write a structure to store the names, salary, and hours of work per day of 10 employees in a company. Write a program to increase the salary depending on the number of hours of work per day as follows and then print the name of all the employees along with their final salaries. Assume:

Hours of work per day	8	10	>=12
Increase in salary	₹4000	₹8000	₹12000

Test Cases: Input

```
A   40000   8
B   10000  10
C   60000  14
```

Output:

```
A   44000   8
B   18000  10
C   72000  12
```

```c
#include <stdio.h>

// Define a structure for Employee
struct Employee {
    char name[50];
    float salary;
    int hoursWorked;
};

// Function to increase salary based on hours worked
void increaseSalary(struct Employee *employee) {
    if (employee->hoursWorked >= 12) {
        employee->salary += 12000;
    } else if (employee->hoursWorked >= 10) {
        employee->salary += 8000;
    } else if (employee->hoursWorked >= 8) {
        employee->salary += 4000;
    }
}
```

```c
// Function to print details of all employees
void printEmployeeDetails(struct Employee employees[], int size) {
    printf("Employee Details:\n");
    for (int i = 0; i < size; i++) {
        printf("Name: %s\n", employees[i].name);
        printf("Final Salary: ₹%.2f\n", employees[i].salary);
        printf("\n");
    }
}

int main() {
    // Declare an array of structures to store employee details
    struct Employee employees[10];

    // Input details for 10 employees
    for (int i = 0; i < 10; i++) {
        printf("Enter details for employee %d:\n", i + 1);
        printf("Name: ");
        scanf("%s", employees[i].name);

        printf("Salary: ₹");
        scanf("%f", &employees[i].salary);

        printf("Hours of work per day: ");
        scanf("%d", &employees[i].hoursWorked);
    }

    // Increase salary based on hours worked
    for (int i = 0; i < 10; i++) {
        increaseSalary(&employees[i]);
    }

    // Print details of all employees
    printEmployeeDetails(employees, 10);

    return 0;
}
```

E26

Write a C program to read the first line from a file.

Test Cases: Suppose the program.txt file contains the following text in the current directory.

Java is Object Oriented Programming.

How are you?

Welcome to VIT

Output: Java is Object Oriented Programming.

```c
#include <stdio.h>

int main() {
    FILE *file;
    char filename[100];
    char line[256]; // Assuming a maximum line length of 255 characters

    // Input the filename from the user
    printf("Enter the filename: ");
    scanf("%s", filename);

    // Open the file for reading
    file = fopen(filename, "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Could not open the file.\n");
        return 1;
    }

    // Read the first line from the file
    if (fgets(line, sizeof(line), file) != NULL) {
        // Print the first line
        printf("First line from the file:\n%s", line);
    } else {
        printf("The file is empty.\n");
    }

    // Close the file
    fclose(file);

    return 0;
}
```

E27

Write a C program to create a student database using file.

Perform following operations:

1. Open file
2. Write five records in file.
3. Read all five records from file.
4. Search for a particular student from file and print his/her details.

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

// Define a structure for Student
struct Student {
    int rollNo;
    char name[50];
    int age;
};
```

```c

// Function to write records to the file
void writeRecords(FILE *file, int numRecords) {
    struct Student student;

    // Input and write five records to the file
    for (int i = 0; i < numRecords; i++) {
        printf("Enter details for student %d:\n", i + 1);
        printf("Roll No: ");
        scanf("%d", &student.rollNo);

        printf("Name: ");
        scanf("%s", student.name);

        printf("Age: ");
        scanf("%d", &student.age);

        // Write the record to the file
        fwrite(&student, sizeof(struct Student), 1, file);
    }

    printf("Records written to the file.\n");
}

// Function to read all records from the file
void readRecords(FILE *file, int numRecords) {
    struct Student student;

    printf("Reading all records from the file:\n");

    // Read and print all records from the file
    for (int i = 0; i < numRecords; i++) {
        fread(&student, sizeof(struct Student), 1, file);

        printf("Record %d:\n", i + 1);
        printf("Roll No: %d\n", student.rollNo);
        printf("Name: %s\n", student.name);
        printf("Age: %d\n", student.age);
        printf("\n");
    }
}

// Function to search for a particular student and print details
void searchStudent(FILE *file, int rollNo, int numRecords) {
    struct Student student;

    // Search for the student with the given roll number
    fseek(file, 0, SEEK_SET); // Move the file pointer to the beginning

    while (fread(&student, sizeof(struct Student), 1, file) == 1) {
        if (student.rollNo == rollNo) {
            printf("Details of student with Roll No %d:\n", rollNo);
            printf("Name: %s\n", student.name);
            printf("Age: %d\n", student.age);
            return;
        }
    }

    printf("Student with Roll No %d not found.\n", rollNo);
}

int main() {
    FILE *file;
    char filename[100];
    int numRecords = 5;

    // Input the filename from the user
    printf("Enter the filename: ");
    scanf("%s", filename);

    // Open the file for writing in binary mode
    file = fopen(filename, "wb");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Could not open the file.\n");
        return 1;
    }

    // Perform operations
    writeRecords(file, numRecords);
    fclose(file);

    // Open the file for reading in binary mode
    file = fopen(filename, "rb");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Could not open the file for reading.\n");
        return 1;
    }

    readRecords(file, numRecords);

    // Close the file
    fclose(file);
```

```c

        // Get roll number from the user to search
        int searchRollNo;
        printf("Enter the Roll No to search: ");
        scanf("%d", &searchRollNo);

        // Open the file for reading in binary mode
        file = fopen(filename, "rb");

        // Check if the file was opened successfully
        if (file == NULL) {
            printf("Could not open the file for searching.\n");
            return 1;
        }

        // Search for a particular student and print details
        searchStudent(file, searchRollNo, numRecords);

        // Close the file
        fclose(file);

        return 0;
    }

```

E28

Write a C program to copy one file contents to another file using character by character. Consider a small source file of 4-5 lines only.

```c
#include <stdio.h>

int main() {
    FILE *sourceFile, *destinationFile;
    char sourceFilename[100], destinationFilename[100];
    char character;

    // Input the source filename from the user
    printf("Enter the source filename: ");
    scanf("%s", sourceFilename);

    // Open the source file for reading
    sourceFile = fopen(sourceFilename, "r");

    // Check if the source file was opened successfully
    if (sourceFile == NULL) {
        printf("Could not open the source file.\n");
        return 1;
    }

    // Input the destination filename from the user
    printf("Enter the destination filename: ");
    scanf("%s", destinationFilename);

    // Open the destination file for writing
    destinationFile = fopen(destinationFilename, "w");

    // Check if the destination file was opened successfully
    if (destinationFile == NULL) {
        printf("Could not open the destination file.\n");
        fclose(sourceFile);
        return 1;
    }

    // Copy character by character from source to destination
    while ((character = fgetc(sourceFile)) != EOF) {
        fputc(character, destinationFile);
    }

    // Close the files
    fclose(sourceFile);
    fclose(destinationFile);

    printf("File contents copied successfully.\n");

    return 0;
}
```

E29

Perform following operations on 2D Matrix:

159. Accept number of rows and columns of two matrices and read elements of both matrices.
160. Print Transpose of both matrices.
161. Print Addition of two matrices.

```c
#include <stdio.h>

// Function to accept elements for a matrix
void inputMatrix(int matrix[][100], int rows, int cols) {
    printf("Enter elements for the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

// Function to print a matrix
void printMatrix(int matrix[][100], int rows, int cols) {
    printf("Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }
}

// Function to calculate transpose of a matrix
void transposeMatrix(int matrix[][100], int transposedMatrix[][100], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            transposedMatrix[j][i] = matrix[i][j];
        }
    }
}

// Function to calculate addition of two matrices
void addMatrices(int matrix1[][100], int matrix2[][100], int resultMatrix[][100], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            resultMatrix[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

int main() {
    int rows, cols;

    // Input number of rows and columns for the matrices
    printf("Enter number of rows: ");
    scanf("%d", &rows);

    printf("Enter number of columns: ");
    scanf("%d", &cols);

    // Check if the dimensions are non-negative
    if (rows <= 0 || cols <= 0) {
        printf("Invalid dimensions. Please enter valid values.\n");
        return 1;
    }

    int matrix1[100][100], matrix2[100][100], transposedMatrix1[100][100], transposedMatrix2[100][100];
    int resultMatrix[100][100];

    // Input elements for the first matrix
    printf("For Matrix 1:\n");
    inputMatrix(matrix1, rows, cols);

    // Input elements for the second matrix
    printf("For Matrix 2:\n");
    inputMatrix(matrix2, rows, cols);

    // Print the original matrices
    printf("Original Matrices:\n");
    printMatrix(matrix1, rows, cols);
    printMatrix(matrix2, rows, cols);

    // Calculate and print the transpose of both matrices
    transposeMatrix(matrix1, transposedMatrix1, rows, cols);
    transposeMatrix(matrix2, transposedMatrix2, rows, cols);

    printf("Transpose of Matrix 1:\n");
    printMatrix(transposedMatrix1, cols, rows);

    printf("Transpose of Matrix 2:\n");
    printMatrix(transposedMatrix2, cols, rows);

    // Calculate and print the addition of two matrices
    addMatrices(matrix1, matrix2, resultMatrix, rows, cols);
```

```c

        printf("Addition of Matrices:\n");
        printMatrix(resultMatrix, rows, cols);

        return 0;
    }

```

E30

Accept number of rows and columns and read elements of matrix.

1. Print matrix in row major format.
2. Print matrix in column major format.

Test Case 1:

Number of rows:2

Number of columns: 3

Matrix Elements: 1 2 3 4 5 6

Row Major:

```
1   2   3

4   5   6
```

Column Major

```
1    3    5
2    4    6
```

```c
#include <stdio.h>

// Function to accept elements for a matrix
void inputMatrix(int matrix[][100], int rows, int cols) {
    printf("Enter elements for the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
```

```c
}

// Function to print matrix in row major format
void printRowMajor(int matrix[][100], int rows, int cols) {
    printf("Matrix in Row Major Format:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
    }
    printf("\n");
}

// Function to print matrix in column major format
void printColumnMajor(int matrix[][100], int rows, int cols) {
    printf("Matrix in Column Major Format:\n");
    for (int j = 0; j < cols; j++) {
        for (int i = 0; i < rows; i++) {
            printf("%d\t", matrix[i][j]);
        }
    }
    printf("\n");
}

int main() {
    int rows, cols;

    // Input number of rows and columns for the matrix
    printf("Enter number of rows: ");
    scanf("%d", &rows);

    printf("Enter number of columns: ");
    scanf("%d", &cols);

    // Check if the dimensions are non-negative
    if (rows <= 0 || cols <= 0) {
        printf("Invalid dimensions. Please enter valid values.\n");
        return 1;
    }

    int matrix[100][100];

    // Input elements for the matrix
    inputMatrix(matrix, rows, cols);

    // Print the original matrix
    printf("Original Matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            printf("%d\t", matrix[i][j]);
        }
        printf("\n");
    }

    // Print the matrix in row major format
    printRowMajor(matrix, rows, cols);

    // Print the matrix in column major format
    printColumnMajor(matrix, rows, cols);

    return 0;
}
```

E31

Write a C Program to count number of characters in the file and print every character on new line on screen.

```c
#include <stdio.h>

int main() {
    FILE *file;
    char filename[100];
    char character;
    int count = 0;

    // Input the filename from the user
    printf("Enter the filename: ");
    scanf("%s", filename);

    // Open the file for reading
    file = fopen(filename, "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Could not open the file.\n");
        return 1;
    }

    // Count the number of characters in the file
    while ((character = fgetc(file)) != EOF) {
        count++;
    }

    // Print the count
    printf("Number of characters in the file: %d\n", count);

    // Close the file
    fclose(file);

    // Open the file again for reading
    file = fopen(filename, "r");

    // Check if the file was opened successfully
    if (file == NULL) {
        printf("Could not open the file.\n");
        return 1;
    }

    // Print every character on a new line
    printf("Characters in the file:\n");
    while ((character = fgetc(file)) != EOF) {
        printf("%c\n", character);
    }

    // Close the file
    fclose(file);

    return 0;
}
```

E32

Write a c program to Delete all occurrences of Character from the String.

Test case: Computer_engineering

Enter character to delete: e

Output: Computr_nginring

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

// Function to delete all occurrences of a character from the string
void deleteChar(char *str, char ch) {
    int i, j;
    int len = strlen(str);

    for (i = j = 0; i < len; i++) {
        if (str[i] != ch) {
            str[j++] = str[i];
        }
    }

    str[j] = '\0';  // Null-terminate the modified string
}

int main() {
    char inputString[100], characterToDelete;

    // Input the string
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);

    // Input the character to delete
    printf("Enter the character to delete: ");
    scanf(" %c", &characterToDelete);

    // Delete all occurrences of the character
    deleteChar(inputString, characterToDelete);

    // Print the modified string
    printf("String after deleting '%c': %s\n", characterToDelete, inputString);

    return 0;
}
```

E33

Write a c program to insert a sub-string in to given main string.

Enter First String: Life is beautiful

Enter Second String: very

Enter the position to insert second string in first: 9

Output: Life is very beautiful

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

// Function to insert a sub-string into a main string
```

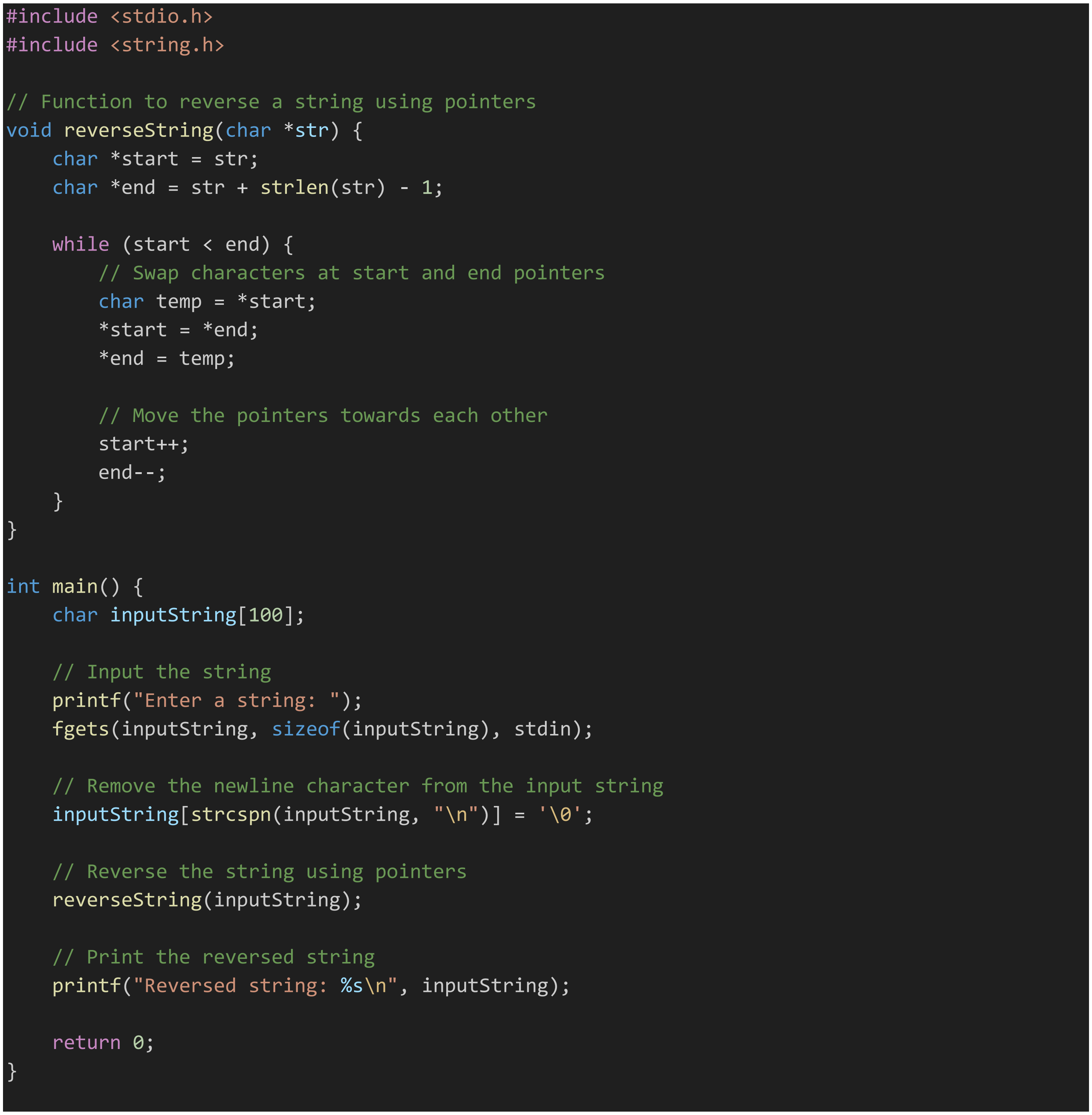
```c
void insertSubstring(char *mainStr, const char *subStr, int position) {
    int mainLen = strlen(mainStr);
    int subLen = strlen(subStr);

    // Ensure the position is within bounds
    if (position < 0 || position > mainLen) {
        printf("Invalid position for insertion.\n");
        return;
    }

    // Shift characters to make space for the sub-string
    for (int i = mainLen; i >= position; i--) {
        mainStr[i + subLen] = mainStr[i];
    }

    // Copy the sub-string into the main string
    for (int i = 0; i < subLen; i++) {
        mainStr[position + i] = subStr[i];
    }
}

int main() {
    char mainString[100], subString[50];
    int position;

    // Input the main string
    printf("Enter the main string: ");
    fgets(mainString, sizeof(mainString), stdin);

    // Remove the newline character from the main string
    mainString[strcspn(mainString, "\n")] = '\0';

    // Input the sub-string
    printf("Enter the sub-string to insert: ");
    fgets(subString, sizeof(subString), stdin);

    // Remove the newline character from the sub-string
    subString[strcspn(subString, "\n")] = '\0';

    // Input the position to insert the sub-string
    printf("Enter the position to insert the sub-string: ");
    scanf("%d", &position);

    // Insert the sub-string into the main string
    insertSubstring(mainString, subString, position);

    // Print the modified main string
    printf("String after insertion: %s\n", mainString);
    return 0;
}
```

E34

1. Write a C program to print a given string in upper case using C

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

// Function to convert a string to uppercase
void toUpperCase(char *str) {
    for (int i = 0; str[i] != '\0'; i++) {
        str[i] = toupper(str[i]);
    }
}

int main() {
    char inputString[100];

    // Input the string
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);

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

    // Convert the string to uppercase
    toUpperCase(inputString);

    // Print the string in uppercase
    printf("Uppercase string: %s\n", inputString);

    return 0;
}

```

2.Write a C program to Reverse a string using pointers



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

// Function to reverse a string using pointers
void reverseString(char *str) {
    char *start = str;
    char *end = str + strlen(str) - 1;

    while (start < end) {
        // Swap characters at start and end pointers
        char temp = *start;
        *start = *end;
        *end = temp;

        // Move the pointers towards each other
        start++;
        end--;
    }
}

int main() {
    char inputString[100];

    // Input the string
    printf("Enter a string: ");
    fgets(inputString, sizeof(inputString), stdin);

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

    // Reverse the string using pointers
    reverseString(inputString);

    // Print the reversed string
    printf("Reversed string: %s\n", inputString);

    return 0;
}
```

E35

Write a C program to evaluate a^b using function.

```c
#include <stdio.h>

// Function to calculate power a^b
double power(double a, int b) {
    double result = 1.0;

    for (int i = 0; i < b; i++) {
        result *= a;
    }

    return result;
}

int main() {
    double base;
    int exponent;

    // Input base and exponent
    printf("Enter the base (a): ");
    scanf("%lf", &base);

    printf("Enter the exponent (b): ");
    scanf("%d", &exponent);

    // Calculate and print the result
    double result = power(base, exponent);
    printf("%.2lf ^ %d = %.2lf\n", base, exponent, result);

    return 0;
}
```

Write a C program to find out the maximum number in an array using function.

```c
#include <stdio.h>

// Function to find the maximum number in an array
int findMax(int arr[], int size) {
    int max = arr[0];

    for (int i = 1; i < size; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    return max;
}

int main() {
    int size;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check if the size is non-negative
    if (size <= 0) {
        printf("Invalid array size. Please enter a valid size.\n");
        return 1;
    }

    int array[size];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }

    // Find and print the maximum number
    int max = findMax(array, size);
    printf("Maximum number in the array: %d\n", max);

    return 0;
}

```

E36

Write a C program to find HCF and LCM of two numbers given by user.

```c
#include <stdio.h>

// Function to calculate HCF (GCD) of two numbers
int calculateHCF(int a, int b) {
    if (b == 0) {
        return a;
    } else {
        return calculateHCF(b, a % b);
    }
}

// Function to calculate LCM of two numbers
int calculateLCM(int a, int b) {
    return (a * b) / calculateHCF(a, b);
}

int main() {
    int num1, num2;

    // Input two numbers
    printf("Enter the first number: ");
    scanf("%d", &num1);

    printf("Enter the second number: ");
    scanf("%d", &num2);

    // Check if numbers are non-negative
    if (num1 < 0 || num2 < 0) {
        printf("Please enter non-negative numbers.\n");
        return 1;
    }

    // Calculate and print HCF and LCM
    int hcf = calculateHCF(num1, num2);
    int lcm = calculateLCM(num1, num2);

    printf("HCF of %d and %d: %d\n", num1, num2, hcf);
    printf("LCM of %d and %d: %d\n", num1, num2, lcm);

    return 0;
}
```

E37

Write a C program to accept two matrices and check if they are equal or not. Order of both matrices must be accepted from user at run time.

```c
#include <stdio.h>

// Function to input elements for a matrix
void inputMatrix(int matrix[][100], int rows, int cols) {
    printf("Enter elements for the matrix:\n");
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            scanf("%d", &matrix[i][j]);
        }
    }
}

// Function to check if two matrices are equal
int areMatricesEqual(int matrix1[][100], int matrix2[][100], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix1[i][j] != matrix2[i][j]) {
                return 0; // Matrices are not equal
            }
        }
    }
    return 1; // Matrices are equal
}

int main() {
    int rows, cols;

    // Input the number of rows and columns for the matrices
    printf("Enter the number of rows: ");
    scanf("%d", &rows);

    printf("Enter the number of columns: ");
    scanf("%d", &cols);

    // Check if the dimensions are non-negative
    if (rows <= 0 || cols <= 0) {
        printf("Invalid dimensions. Please enter valid values.\n");
        return 1;
    }

    int matrix1[100][100], matrix2[100][100];
    // Input elements for the first matrix
    printf("For Matrix 1:\n");
    inputMatrix(matrix1, rows, cols);

    // Input elements for the second matrix
    printf("For Matrix 2:\n");
    inputMatrix(matrix2, rows, cols);

    // Check if the matrices are equal
    if (areMatricesEqual(matrix1, matrix2, rows, cols)) {
        printf("Matrices are equal.\n");
    } else {
        printf("Matrices are not equal.\n");
    }

    return 0;
}
```

E38

1. Write a C program to insert a given number in the array at given position.

```
#include <stdio.h>

// Function to insert a number into an array at a given position
void insertNumber(int arr[], int *size, int position, int number) {
    // Check if the position is valid
    if (position < 0 || position > *size) {
        printf("Invalid position for insertion.\n");
        return;
    }

    // Shift elements to make space for the new number
    for (int i = *size; i > position; i--) {
        arr[i] = arr[i - 1];
    }

    // Insert the number at the specified position
    arr[position] = number;

    // Update the size of the array
    (*size)++;
}
```

```c

int main() {
    int size, position, number;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check if the size is non-negative
    if (size <= 0) {
        printf("Invalid array size. Please enter a valid size.\n");
        return 1;
    }

    int array[100];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array[i]);
    }

    // Input the position and number to insert
    printf("Enter the position to insert the number: ");
    scanf("%d", &position);

    printf("Enter the number to insert: ");
    scanf("%d", &number);

    // Insert the number into the array
    insertNumber(array, &size, position, number);

    // Print the modified array
    printf("Array after insertion:\n");
    for (int i = 0; i < size; i++) {
        printf("%d ", array[i]);
    }
    printf("\n");

    return 0;
}

```

Write a C program to remove a number in the array from a given position.

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

```c

// Function to remove a number from an array at a given position
void removeNumber(int arr[], int *size, int position) {
    // Check if the position is valid
    if (position < 0 || position >= *size) {
        printf("Invalid position for removal.\n");
        return;
    }

    // Shift elements to fill the gap left by the removed number
    for (int i = position; i < *size - 1; i++) {
        arr[i] = arr[i + 1];
    }

    // Update the size of the array
    (*size)--;
}

int main() {
    int size, position;

    // Input the size of the array
    printf("Enter the size of the array: ");
    scanf("%d", &size);

    // Check if the size is non-negative
    if (size <= 0) {
        printf("Invalid array size. Please enter a valid size.\n");
        return 1;
    }

    int array[100];

        // Input elements for the array
        printf("Enter %d elements for the array:\n", size);
        for (int i = 0; i < size; i++) {
            scanf("%d", &array[i]);
        }

        // Input the position to remove the number
        printf("Enter the position to remove the number: ");
        scanf("%d", &position);

        // Remove the number from the array
        removeNumber(array, &size, position);

        // Print the modified array
        printf("Array after removal:\n");
        for (int i = 0; i < size; i++) {
            printf("%d ", array[i]);
        }
        printf("\n");

        return 0;
    }

```

E39

Write a c program for swapping of two arrays using function and check if both arrays are equal or not. Limits and numbers of both arrays must be accepted from user at run time.

```c
#include <stdio.h>

// Function to swap two arrays
void swapArrays(int arr1[], int arr2[], int size) {
    // Swap elements of the two arrays
    for (int i = 0; i < size; i++) {
        int temp = arr1[i];
        arr1[i] = arr2[i];
        arr2[i] = temp;
    }
}

// Function to check if two arrays are equal
int areArraysEqual(int arr1[], int arr2[], int size) {
    for (int i = 0; i < size; i++) {
        if (arr1[i] != arr2[i]) {
            return 0; // Arrays are not equal
        }
    }
    return 1; // Arrays are equal
}

int main() {
    int size;

    // Input the size of the arrays
    printf("Enter the size of the arrays: ");
    scanf("%d", &size);
    // Check if the size is non-negative
    if (size <= 0) {
        printf("Invalid array size. Please enter a valid size.\n");
        return 1;
    }

    int array1[100], array2[100];

    // Input elements for the first array
    printf("Enter %d elements for the first array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array1[i]);
    }

    // Input elements for the second array
    printf("Enter %d elements for the second array:\n", size);
    for (int i = 0; i < size; i++) {
        scanf("%d", &array2[i]);
    }

    // Swap the arrays
    swapArrays(array1, array2, size);

    // Check if the swapped arrays are equal
    if (areArraysEqual(array1, array2, size)) {
        printf("Arrays are equal after swapping.\n");
    } else {
        printf("Arrays are not equal after swapping.\n");
    }

    return 0;
}
```

E40

Write a C program for swapping of two string

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

```c
// Function to swap two strings
void swapStrings(char str1[], char str2[]) {
    char temp[100]; // Assuming a maximum length of 100 characters for each string

    // Copy content of str1 to temp
    strcpy(temp, str1);

    // Copy content of str2 to str1
    strcpy(str1, str2);

    // Copy content of temp to str2
    strcpy(str2, temp);
}

int main() {
    char string1[100], string2[100];

    // Input the first string
    printf("Enter the first string: ");
    fgets(string1, sizeof(string1), stdin);

    // Remove the newline character from the first string
    string1[strcspn(string1, "\n")] = '\0';

    // Input the second string
    printf("Enter the second string: ");
    fgets(string2, sizeof(string2), stdin);

    // Remove the newline character from the second string
    string2[strcspn(string2, "\n")] = '\0';

    // Displaying strings before swapping
    printf("\nStrings before swapping:\n");
    printf("String 1: %s\n", string1);
    printf("String 2: %s\n", string2);

    // Swapping the strings
    swapStrings(string1, string2);

    // Displaying strings after swapping
    printf("\nStrings after swapping:\n");
    printf("String 1: %s\n", string1);
    printf("String 2: %s\n", string2);

    return 0;
}
```

E41

Write a C program for sorting list of elements using bubble sort.

```c
#include <stdio.h>

// Function to perform Bubble Sort on an array
void bubbleSort(int arr[], int n) {
    int temp;
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            // Swap if the element found is greater than the next element
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main() {
    int n;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Check if the number of elements is non-negative
    if (n <= 0) {
        printf("Invalid number of elements. Please enter a valid number.\n");
        return 1;
    }

    int arr[100];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Perform Bubble Sort
    bubbleSort(arr, n);
    // Print the sorted array
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

E42

Write a C program for sorting list of elements using selection sort

```
#include <stdio.h>

// Function to perform Selection Sort on an array
void selectionSort(int arr[], int n) {
    int minIndex, temp;
    for (int i = 0; i < n - 1; i++) {
        minIndex = i;
        for (int j = i + 1; j < n; j++) {
            // Find the index of the minimum element in the unsorted part
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }

        // Swap the found minimum element with the first element
        temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

int main() {
    int n;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);
```

```c
    // Check if the number of elements is non-negative
    if (n <= 0) {
        printf("Invalid number of elements. Please enter a valid number.\n");
        return 1;
    }

    int arr[100];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Perform Selection Sort
    selectionSort(arr, n);

    // Print the sorted array
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```

E43

Write a C program to search element in given list using linear search . also find largest element in given array.

```c
#include <stdio.h>

// Function to perform linear search and find the index of an element
int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            return i; // Return the index if the element is found
        }
    }
    return -1; // Return -1 if the element is not found
}
```

```c
// Function to find the largest element in an array
int findLargestElement(int arr[], int n) {
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }
    return max;
}

int main() {
    int n, key;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Check if the number of elements is non-negative
    if (n <= 0) {
        printf("Invalid number of elements. Please enter a valid number.\n");
        return 1;
    }

    int arr[100];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Input the element to search
    printf("Enter the element to search: ");
    scanf("%d", &key);

    // Perform linear search
    int index = linearSearch(arr, n, key);

    // Check and print the result of linear search
    if (index != -1) {
        printf("Element %d found at index %d using linear search.\n", key, index);
    } else {
        printf("Element %d not found in the array using linear search.\n", key);
    }

    // Find and print the largest element in the array
    int largest = findLargestElement(arr, n);
    printf("Largest element in the array: %d\n", largest);

    return 0;
}
```

E44

Write a C program to find duplicate element in an array.

```c
#include <stdio.h>

// Function to find duplicate elements in an array
void findDuplicates(int arr[], int n) {
    printf("Duplicate elements in the array are: ");

    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            // If duplicate elements are found, print one of them
            if (arr[i] == arr[j]) {
                printf("%d ", arr[i]);
                break; // Break to avoid printing the same element multiple times
            }
        }
    }

    printf("\n");
}

int main() {
    int n;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Check if the number of elements is non-negative
    if (n <= 0) {
        printf("Invalid number of elements. Please enter a valid number.\n");
        return 1;
    }

    int arr[100];
    // Input elements for the array
    printf("Enter %d elements for the array:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Find and print duplicate elements in the array
    findDuplicates(arr, n);

    return 0;
}
```

E45

Write a C program to insert element in an array on given specific position.

```c
#include <stdio.h>

// Function to insert an element into an array at a given position
void insertElement(int arr[], int *n, int position, int element) {
    // Check if the position is valid
    if (position < 0 || position > *n) {
        printf("Invalid position for insertion.\n");
        return;
    }

    // Shift elements to make space for the new element
    for (int i = *n; i > position; i--) {
        arr[i] = arr[i - 1];
    }

    // Insert the element at the specified position
    arr[position] = element;

    // Update the size of the array
    (*n)++;
}
```

```c
int main() {
    int n, position, element;

    // Input the number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Check if the number of elements is non-negative
    if (n <= 0) {
        printf("Invalid number of elements. Please enter a valid number.\n");
        return 1;
    }

    int arr[100];

    // Input elements for the array
    printf("Enter %d elements for the array:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Input the position and element to insert
    printf("Enter the position to insert the element: ");
    scanf("%d", &position);

    printf("Enter the element to insert: ");
    scanf("%d", &element);

    // Insert the element into the array
    insertElement(arr, &n, position, element);

    // Print the modified array
    printf("Array after insertion:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    return 0;
}
```