Basic C Programs

1.Program: Selection Sort

Aim: To write a C program to sort elements of an array in ascending order using the Selection Sort technique.

Algorithm:

1. Start the program.
2. Read the number of elements n.
3. Input array elements.
4. For each element, find the smallest element in the unsorted part.
5. Swap it with the current element.
6. Repeat until the array is sorted.
7. Display the sorted array.
8. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int a[100], n, i, j, temp;

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

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }
    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(a[i] > a[j]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
        }
    }

    printf("Sorted array in ascending order:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }

    return 0;
}
```

Output

```
Enter number of elements: 5
Enter 5 elements:
22 74 99 14 46
Sorted array in ascending order:
14 22 46 74 99

=== Code Execution Successful ===
```

Input:enter no.of elements:5
Enter 5 elements : 22 74 99 14 46
Output:
Sorted array in ascending order: 14 22 46 74 99

Result:
Thus, the C program for Selection Sort was successfully executed and verified.

2.Program: Factorial Using Recursion

Aim: To find the factorial of a given number using recursion in C.

Algorithm:

1. Start the program.
2. Read an integer n.
3. If n == 0 or n == 1, return 1.
4. Else return n * factorial(n - 1).
5. Display the result.
6. Stop.

main.c

```c
#include <stdio.h>

int factorial(int n) {
    if(n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}

int main() {
    int num;

    printf("Enter a number: ");
    scanf("%d", &num);

    printf("Factorial of %d is: %d\n", num, factorial(num));

    return 0;
}
```

Output

```
Enter a number: 7
Factorial of 7 is: 5040


=== Code Execution Successful ===
```

Input:
enter a number : 7
Output:
Factorial of 7 is : 5040

Result:
Thus, the factorial of a number using recursion was successfully implemented.

3.Program: Fibonacci Series

Aim: To write a program for Fibonacci series.

Algorithm:

1. Start the program.
2. Read the number of terms n.
3. Initialize a = 0, b = 1.
4. Print a and b.
5. Use a loop to generate next term c = a + b.
6. Update a = b, b = c.
7. Repeat until all terms are printed.
8. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int n, i;
    int a = 0, b = 1, c;
    printf("Enter number of terms: ");
    scanf("%d", &n);

    if(n <= 0) {
        printf("Enter a positive number.\n");
        return 0;
    }
    printf("Fibonacci series up to %d terms:\n", n);

    if(n >= 1)
        printf("%d ", a);
    if(n >= 2)
        printf("%d ", b);

    for(i = 3; i <= n; i++) {
        c = a + b;
        printf("%d ", c);
        a = b;
        b = c;
    }

    return 0;
}
```

Output

```
Enter number of terms: 7
Fibonacci series up to 7 terms:
0 1 1 2 3 5 8

=== Code Execution Successful ===
```

Input:
Enter no.of terms : 7
Output:
Fibonacci series up to 7 terms: 0 1 1 2 3 5 8

Result:
Thus, the Fibonacci series without recursion was successfully generated.

4.Program: Palindrome

Aim: To check whether a given number is a palindrome or not.

Algorithm:

1. Start the program.
2. Read an integer n.
3. Reverse the number.
4. Compare reversed and original number.
5. If equal → palindrome; else → not palindrome.
6. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int num, rev = 0, rem, temp;

    printf("Enter a number: ");
    scanf("%d", &num);

    temp = num;
    while(num > 0) {
        rem = num % 10;
        rev = rev * 10 + rem;
        num = num / 10;
    }

    if(temp == rev)
        printf("%d is a palindrome.\n", temp);
    else
        printf("%d is not a palindrome.\n", temp);

    return 0;
}
```

Output

```
Enter a number: 12321
12321 is a palindrome.


=== Code Execution Successful ===
```

Input:
Enter a number : 12321
Output:
12321 is a palindrome

Result:
Thus, the C program to check for a palindrome was executed successfully.

5.Program: Prime Number

Aim: To check whether a given number is prime or not.

Algorithm:

1. Start the program.
2. Read an integer n.
3. Check divisibility from 2 to n/2.
4. If divisible → not prime; else → prime.
5. Display the result.
6. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int n, i, count = 0;

    printf("Enter a number: ");
    scanf("%d", &n);

    for(i = 1; i <= n; i++) {
        if(n % i == 0)
            count++;
    }

    if(count == 2)
        printf("Prime number\n");
    else
        printf("Not a prime number\n");

    return 0;
}
```

Output

```
Enter a number: 7
Prime number


=== Code Execution Successful ===
```

Input:
Enter a number : 7
Output:
Prime number

Result:
Thus, the given number was successfully checked for primality.

6.Program: Two Order Homogeneous Recursion

Aim: To implement a C program demonstrating a two-order homogeneous recurrence relation.

Algorithm:

1. Start the program.
2. Read number of terms n.
3. Initialize a0 and a1 as first two terms.
4. For i = 2 to n, calculate a[i] = a[i-1] + a[i-2].
5. Display all the terms.
6. Stop.

main.c

```c
#include <stdio.h>
int sequence(int n, int T0, int T1, int p, int q) {
    if(n == 0)
        return T0;
    else if(n == 1)
        return T1;
    else
        return p * sequence(n - 1, T0, T1, p, q) + q * sequence(n - 2, T0, T1, p
            , q);
}
int main() {
    int n, T0, T1, p, q, i;

    printf("Enter initial term T0: ");
    scanf("%d", &T0);
    printf("Enter initial term T1: ");
    scanf("%d", &T1);
    printf("Enter constants p and q: ");
    scanf("%d %d", &p, &q);
    printf("Enter number of terms: ");
    scanf("%d", &n);

    printf("Sequence generated:\n");
    for(i = 0; i < n; i++) {
        printf("%d ", sequence(i, T0, T1, p, q));
    }

    return 0;
}
```

Output

```
Enter initial term T0: 0
Enter initial term T1: 1
Enter constants p and q: 1 1
Enter number of terms: 7
Sequence generated:
0 1 1 2 3 5 8

=== Code Execution Successful ===
```

Input:
Enter initial term T0: 0
Enter initial term T1: 1
Enter constants p and q : 1 1
Enter no.of terms: 7
Output:
Sequence generated: 0 1 1 2 3 5 8

Result:
Thus, the two-order homogeneous recursion program was successfully executed.

7.Program: Biggest Number in Series

Aim: To find the biggest number in a given series of integers.

Algorithm:

1. Start the program.
2. Read the number of elements n.
3. Input all array elements.
4. Initialize max = first element.
5. Compare each element with max and update if larger.
6. Display the biggest number.
7. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int a[100], n, i, max;

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

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    max = a[0];

    for(i = 1; i < n; i++) {
        if(a[i] > max) {
            max = a[i];
        }
    }

    printf("The biggest number is: %d\n", max);

    return 0;
}
```

Output

```
Enter number of elements: 6
Enter 6 elements:
4 85 56 3 77 23
The biggest number is: 85


=== Code Execution Successful ===
```

Input:
Enter no.of elements : 6
Enter the 6 elements : 4 85 56 3 77 23
Output:
The biggest number is : 85

Result:
Thus, the biggest number in the series was successfully found.

8.Program: Leap Year

Aim: To check whether a given year is a leap year or not.

Algorithm:

1. Start the program.
2. Read the year value.
3. If year is divisible by 400 → leap year.
4. Else if divisible by 100 → not leap year.
5. Else if divisible by 4 → leap year.
6. Else not leap year.
7. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int year;

    printf("Enter a year: ");
    scanf("%d", &year);

    if((year % 400 == 0) || (year % 4 == 0 && year % 100 != 0))
        printf("%d is a leap year.\n", year);
    else
        printf("%d is not a leap year.\n", year);

    return 0;
}
```

Output

```
Enter a year: 2025
2025 is not a leap year.


=== Code Execution Successful ===
```

Input:
Enter a year: 2025
Output:
2025 is not a leap year.

Result:
Thus, the leap year checking program was executed successfully.

9.Program: Swapping

Aim: To swap two numbers using a temporary variable in C.

Algorithm:

1. Start the program.
2. Read two numbers a and b.
3. Use temp = a; a = b; b = temp.
4. Display the swapped values.
5. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int a, b, temp;

    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);

    printf("Before swapping: a = %d, b = %d\n", a, b);

    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d\n", a, b);

    return 0;
}
```

Output

```
Enter two numbers: 4 6
Before swapping: a = 4, b = 6
After swapping: a = 6, b = 4


=== Code Execution Successful ===
```

Input:
Enter two numbers : 4 6
Output:
Before Swapping : a=4 , b=6
After Swapping : a=6 , b=4

Result:
Thus, the swapping of two numbers was successfully performed.

10.Program: Duplicate in a List

Aim: To find and display duplicate elements in a list of integers.

Algorithm:

1. Start the program.
2. Read the number of elements n.
3. Input array elements.
4. Use nested loops to compare each pair of elements.
5. If any two elements are equal, display as duplicate.
6. Stop.

main.c

```c
#include <stdio.h>

int main() {
    int a[100], n, i, j;

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

    printf("Enter %d elements:\n", n);
    for(i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    printf("Duplicate elements are:\n");
    for(i = 0; i < n - 1; i++) {
        for(j = i + 1; j < n; j++) {
            if(a[i] == a[j]) {
                printf("%d\n", a[i]);
                break;
            }
        }
    }

    return 0;
}
```

Output

```
Enter number of elements: 6
Enter 6 elements:
4 8 4 6 9 8
Duplicate elements are:
4
8


=== Code Execution Successful ===
```

Input:
Enter no. of elements : 6
Enter 6 elements: 4 8 4 6 9 8
Output:
Duplicate elements are : 4 8

Result:
Thus, the program to find duplicates in a list was successfully executed.