Operators

1) Write a C program to swap two numbers without using a temporary variable.

Swap Two Numbers Without Using a Temporary Variable

Algorithm

1. Input two numbers, a and b.
2. Swap them using arithmetic operations:
   - a = a + b
   - b = a - b
   - a = a - b
3. Print the swapped values.

C Code

```
#include <stdio.h>

int main() {

   int a, b;

   printf("Enter two numbers: ");

   scanf("%d %d", &a, &b);

   a = a + b;

   b = a - b;

   a = a - b;

   printf("Swapped numbers: a = %d, b = %d\n", a, b);

   return 0;

}
```

2) Write a C program to check whether a given number is even or odd using bitwise operators.

Check if a Number is Even or Odd Using Bitwise Operators

Algorithm

1. Input an integer n.
2. Use bitwise AND: n & 1
    - If the result is 0, the number is even.
    - If the result is 1, the number is odd.
3. Print the result.

C Code

```
#include <stdio.h>

int main() {

   int n;

   printf("Enter a number: ");

   scanf("%d", &n);

   if (n & 1)

      printf("%d is Odd\n", n);

   else

      printf("%d is Even\n", n);

   return 0;

}
```

3) Write a C program to perform bitwise AND, OR, XOR, operations on two given integers.

Perform Bitwise AND, OR, XOR on Two Numbers

Algorithm

1. Input two integers a and b.
2. Compute:
    - Bitwise AND (a & b).
    - Bitwise OR (a | b).
    - Bitwise XOR (a ^ b).
3. Print the results.

C Code

```
#include <stdio.h>

int main() {

   int a, b;

   printf("Enter two numbers: ");

   scanf("%d %d", &a, &b);

   printf("AND: %d\n", a & b);

   printf("OR: %d\n", a | b);

   printf("XOR: %d\n", a ^ b);

   return 0;

}
```

4) Write a C program to count the number of set bits (1s) in the binary representation of a given integer.

Count the Number of Set Bits in an Integer

Algorithm

1. Input an integer n.
2. Initialize count = 0.
3. While n > 0:
    - Increment count if n & 1 is 1.
    - Right shift n (n = n >> 1).
4. Print count.

C Code

```
#include <stdio.h>

int countSetBits(int n) {
   int count = 0;
   while (n) {
      count += n & 1;
      n >>= 1;
   }
   return count;
}

int main() {
   int n;
   printf("Enter a number: ");
   scanf("%d", &n);
   printf("Number of set bits: %d\n", countSetBits(n));
   return 0;
}
```

5) Write a C program to check whether a given number is a power of two or not, using bitwise operators.

Check if a Number is a Power of Two

Algorithm

1. Input an integer n.
2. If n > 0 and n & (n - 1) == 0, it is a power of two.
3. Print the result.

C Code

```
#include <stdio.h>

int isPowerOfTwo(int n) {
   return (n > 0) && ((n & (n - 1)) == 0);
}

int main() {
   int n;
   printf("Enter a number: ");
   scanf("%d", &n);
   if (isPowerOfTwo(n))
      printf("%d is a power of two.\n", n);
   else
      printf("%d is not a power of two.\n", n);
   return 0;
}
```

6) Write a C program to check if a given number is positive, negative, or zero using conditional operators.

Check if a Number is Positive, Negative, or Zero

Algorithm

1. Input n.
2. Use a conditional operator to check:
   - If n > 0: Positive.
   - If n < 0: Negative.
   - Otherwise: Zero.
3. Print the result.

C Code

```
#include <stdio.h>

int main() {

  int n;

  printf("Enter a number: ");

  scanf("%d", &n);

  (n > 0) ? printf("Positive\n") : (n < 0) ? printf("Negative\n") : printf("Zero\n");

  return 0;

}
```

7) Write a C program to check if a given number is a strong number or not using arithmetic operators.

Check if a Number is a Strong Number

Algorithm

1. Input n, store original = n.
2. Find the factorial sum of digits.
3. Compare sum with original.
4. If equal, print "Strong Number"; else, "Not a Strong Number".

C Code

```
#include <stdio.h>

int factorial(int n) {

   int fact = 1;

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

      fact *= i;

   return fact;

}

int isStrongNumber(int n) {

   int original = n, sum = 0;

   while (n > 0) {

      sum += factorial(n % 10);

      n /= 10;

   }

   return sum == original;

}

int main() {

   int n;

   printf("Enter a number: ");

   scanf("%d", &n);
```

```
    if (isStrongNumber(n))
        printf("%d is a Strong Number\n", n);
    else
        printf("%d is not a Strong Number\n", n);
    return 0;
}
```

8) Write a C program to check if a given number is a palindrome or not using arithmetic operators.

Check if a Number is a Strong Number

Algorithm

1. Input n, store original = n.
2. Find the factorial sum of digits.
3. Compare sum with original.
4. If equal, print "Strong Number"; else, "Not a Strong Number".

C Code

```
#include <stdio.h>

int factorial(int n) {

    int fact = 1;

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

        fact *= i;

    return fact;

}

int isStrongNumber(int n) {

    int original = n, sum = 0;

    while (n > 0) {

        sum += factorial(n % 10);

        n /= 10;

    }

    return sum == original;

}

int main() {

    int n;

    printf("Enter a number: ");

    scanf("%d", &n);
    if (isStrongNumber(n))
        printf("%d is a Strong Number\n", n);
    else
        printf("%d is not a Strong Number\n", n);


    return 0;
}
```

9) Write a C program to check if a given number is a prime number or not using a combination of arithmetic and logical operators.

Check if a Number is Prime

Algorithm

1. If n < 2, return false.
2. Check divisibility from 2 to sqrt(n).
3. If no divisors found, print "Prime"; else, "Not Prime".

C Code

```
#include <stdio.h>

#include <math.h>

int isPrime(int n) {

   if (n < 2) return 0;

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

      if (n % i == 0)

         return 0;

   }

   return 1;

}

int main() {

   int n;

   printf("Enter a number: ");

   scanf("%d", &n);

   if (isPrime(n))

      printf("%d is a Prime Number\n", n);

   else

      printf("%d is not a Prime Number\n", n);

   return 0;

}
```

10) Write a C program to check if a given year is a leap year or not using logical operators.

Check if a Year is a Leap Year

Algorithm

1. If (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0), it is a leap year.
2. Print result.

C Code

```
#include <stdio.h>

int main() {

   int year;

   printf("Enter a year: ");

   scanf("%d", &year);

   if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))

      printf("%d is a Leap Year\n", year);

   else

      printf("%d is not a Leap Year\n", year);

   return 0;

}
```