SOLUTION BOOK

❤️ From SIDDHARTH SINGH

LOOP

1) **Program to Find Factorial**

THEORY:

The factorial of a positive integer n is equal to 1*2*3*...n.

Note:

The factorial of a negative number doesn't exist. And the factorial of 0 is 1.

n! = 1 if n = 0 or n = 1

CODE:

```
#include <iostream>
using namespace std;

int main() {
    int n;
    long double factorial = 1.0;

    cout << "Enter a positive integer: ";
    cin >> n;

    if (n < 0)
        cout << "Error! Factorial of a negative number doesn't exist.";
    else {
        for(int i = 1; i <= n; ++i) {
            factorial *= i;
        }
        cout << "Factorial of " << n << " = " << factorial;
    }

    return 0;
}
```

Output

```
Enter a positive integer: 12
Factorial of 12 = 479001600
```

NOTE:

SOLUTION BOOK

❤️ From SIDDHARTH SINGH

This program can calculate the factorial only up to 1754 or some integer value close to it. Beyond that, the program can no longer calculate the factorial as the results exceed the range of long double data type.

2) Program to Generate Multiplication Table

CODE:

```
#include <iostream>
using namespace std;

int main()
{
    int n;

    cout << "Enter a positive integer: ";
    cin >> n;

    for (int i = 1; i <= 10; ++i) {
        cout << n << " * " << i << " = " << n * i << endl;
    }

    return 0;
}
```

Output

```
Enter an integer: 5
5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45
5 * 10 = 50
```

3) Program to Display Fibonacci Series upto Nth Term

THEORY:

The Fibonacci sequence is a series where the next term is the sum of previous two terms. The first two terms of the Fibonacci sequence is 0 followed by 1.

The Fibonacci sequence: 0, 1, 1, 2, 3, 5, 8, 13, 21

CODE:

```
#include <iostream>
using namespace std;

int main() {
   int n, t1 = 0, t2 = 1, nextTerm = 0;

   cout << "Enter the number of terms: ";
   cin >> n;

   cout << "Fibonacci Series: ";

   for (int i = 1; i <= n; ++i) {
      // Prints the first two terms.
      if(i == 1) {
         cout << t1 << ", ";
         continue;
      }
      if(i == 2) {
         cout << t2 << ", ";
         continue;
      }
      nextTerm = t1 + t2;
      t1 = t2;
      t2 = nextTerm;

      cout << nextTerm << ", ";
   }
   return 0;
}
```

Output

```
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34
```

4) **Program to Generate Fibonacci Sequence Up to a Certain Number**

CODE:

```
#include <iostream>
```

```
using namespace std;

int main() {
   int t1 = 0, t2 = 1, nextTerm = 0, n;

   cout << "Enter a positive number: ";
   cin >> n;

   // displays the first two terms which is always 0 and 1
   cout << "Fibonacci Series: " << t1 << ", " << t2 << ", ";

   nextTerm = t1 + t2;

   while(nextTerm <= n) {
      cout << nextTerm << ", ";
      t1 = t2;
      t2 = nextTerm;
      nextTerm = t1 + t2;
   }
   return 0;
}
```

Output

```
Enter a positive integer: 100
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,
```

5) Program to Find GCD

Method 1: Using for loop

```
#include <iostream>
using namespace std;

int main() {
  int n1, n2, hcf;
  cout << "Enter two numbers: ";
  cin >> n1 >> n2;

  // swapping variables n1 and n2 if n2 is greater than n1.
  if ( n2 > n1) {
    int temp = n2;
    n2 = n1;
    n1 = temp;
```

```
  }

  for (int i = 1; i <=  n2; ++i) {
    if (n1 % i == 0 && n2 % i ==0) {
      hcf = i;
    }
  }

  cout << "HCF = " << hcf;

  return 0;
}
```

IMP:

Method 2: Using while loop

```
#include <iostream>
using namespace std;

int main() {
  int n1, n2;

  cout << "Enter two numbers: ";
  cin >> n1 >> n2;

  while(n1 != n2) {
    if(n1 > n2)
      n1 -= n2;
    else
      n2 -= n1;
  }

  cout << "HCF = " << n1;

  return 0;
}
```

Output

```
Enter two numbers: 16
76
HCF = 4
```

CONCEPT:

In the above program, the smaller number is subtracted from the larger number and that number is stored in place of the larger number.

Here, n1 -= n2 is the same as n1 = n1 - n2. Similarly, n2 -= n1 is the same as n2 = n2 - n1.

This process is continued until the two numbers become equal which will be HCF.

Let us look at how this program works when n1 = 16 and n2 = 76.

n1	n2	n1 > n2	n1 -= n2	n2 -= n1	n1 != n2
16	76	false	-	60	true
16	60	false	-	44	true
16	44	false	-	28	true
16	28	false	-	12	true
16	12	true	4	-	true
4	12	false	-	8	true
4	8	false	-	4	false

Here, the loop terminates when n1 != n2 becomes false.

After the final iteration of the loop, n1 = n2 = 4. This is the value of the GCD/HCF since this is the greatest number that can divide both 16 and 76.

6) Program to Find LCM

CODE:

```
#include <iostream>
using namespace std;

int main()
{
   int n1, n2, max;

   cout << "Enter two numbers: ";
   cin >> n1 >> n2;

   // maximum value between n1 and n2 is stored in max
   max = (n1 > n2) ? n1 : n2;

   do
   {
```

```
        if (max % n1 == 0 && max % n2 == 0)
        {
            cout << "LCM = " << max;
            break;
        }
        else
            ++max;
    } while (true);

    return 0;
}
```

Output

```
Enter two numbers: 12
18
LCM = 36
```

EXPLANATION:

In the above program, the user is asked to integer two integers n1 and n2 and the largest of those two numbers is stored in max.

It is checked whether max is divisible by n1 and n2, if it's divisible by both numbers, max (which contains LCM) is printed and loop is terminated.

If not, value of max is incremented by 1 and the same process goes on until max is divisible by both n1 and n2.

OTHER METHOD:

The LCM of two numbers is given by:
LCM = (n1 * n2) / HCF
So we can calculate LCM by calculating HCF (above question)

7) Program to Reverse a Number

CODE:

```
#include <iostream>
using namespace std;

int main() {
    int n, reversedNumber = 0, remainder;

    cout << "Enter an integer: ";
```

```
    cin >> n;

    while(n != 0) {
        remainder = n%10;
        reversedNumber = reversedNumber*10 + remainder;
        n /= 10;
    }

    cout << "Reversed Number = " << reversedNumber;

    return 0;
}
```

Output

```
Enter an integer: 12345
Reversed number = 54321
```

8) **Program to Sum digits of Number**

CODE:

```
#include <iostream>
using namespace std;

int main() {
    int n, sum = 0, remainder;

    cout << "Enter an integer: ";
    cin >> n;

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

    cout << "Sum of Digits = " << sum;
    return 0;
}
```

Output

```
Enter an integer: 12345
Sum of Digits =15
```

SOLUTION BOOK

❤️ From SIDDHARTH SINGH

---

9) Program to Calculate Power of a Number using pow()

CODE:

```
#include <iostream>
#include <cmath>

using namespace std;

int main()
{
   float base, exponent, result;

   cout << "Enter base and exponent respectively:  ";
   cin >> base >> exponent;

   result = pow(base, exponent);

   cout << base << "^" << exponent << " = " << result;

   return 0;
}
```

Output

```
Enter base and exponent respectively:  2.3
4.5
2.3^4.5 = 42.44
```

CONCEPT:

We have included the cmath header file in order to use the pow() function. We then use the pow() function to calculate the power. The first argument is the base, and the second argument is the exponent.

10) Program to Calculate Power of a Number without using pow()

CODE:

```
#include <iostream>
using namespace std;
```

```
int main()
{
    int exponent;
    float base, result = 1;

    cout << "Enter base and exponent respectively:  ";
    cin >> base >> exponent;

    cout << base << "^" << exponent << " = ";

    while (exponent != 0) {
        result *= base;
        --exponent;
    }

    cout << result;
    return 0;
}
```

Output

```
Enter base and exponent respectively:  3.4
5
3.4^5 = 454.354
```

11) Program to Check Whether a Number is Palindrome or Not

THEORY:

If the reversed integer is equal to the integer entered by user then, that number is a palindrome if not that number is not a palindrome.

CODE:

```
#include <iostream>
using namespace std;
int main()
{
    int n, num, digit, rev = 0;

    cout << "Enter a positive number: ";
    cin >> num;

    n = num;

    do
```

```
    {
        digit = num % 10;
        rev = (rev * 10) + digit;
        num = num / 10;
    } while (num != 0);

    cout << " The reverse of the number is: " << rev << endl;

    if (n == rev)
        cout << " The number is a palindrome.";
    else
        cout << " The number is not a palindrome.";

    return 0;
}
```

Output 1:

```
Enter a positive number: 12321
The reverse of the number is: 12321
The number is a palindrome.
```

Output 2:

```
Enter a positive number: 12331
The reverse of the number is: 13321
The number is not a palindrome.
```

12) Program to Check Whether a Number is Prime or Not

CODE:

```
#include <iostream>
using namespace std;
int main() {
    int i, n;
    bool isPrime = true;

    cout << "Enter a positive integer: ";
    cin >> n;

    // 0 and 1 are not prime numbers
    if (n == 0 || n == 1) {
        isPrime = false;
    }
    else {
```

```
        for (i = 2; i <= n / 2; ++i) {
            if (n % i == 0) {
                isPrime = false;
                break;
            }
        }
    }
    if (isPrime)
        cout << n << " is a prime number";
    else
        cout << n << " is not a prime number";

    return 0;
}
```

Output:

Enter a positive integer: **29**

29 is a prime number.

13) Program to Display Prime Numbers Between Two Intervals

CODE:

```
#include <iostream>
using namespace std;

int main() {
    int low, high, i;
    bool isPrime = true;

    cout << "Enter two numbers: ";
    cin >> low >> high;

    cout << "\nPrime numbers between " << low << " and " << high << " are: " << endl;

    while (low < high) {
        isPrime = true;
        if (low == 0 || low == 1) {
            isPrime = false;
        }
        else {
            for (i = 2; i <= low / 2; ++i) {
                if (low % i == 0) {
                    isPrime = false;
```

```
                break;
            }
        }
    }

    if (isPrime)
        cout << low << " ";

    ++low;
  }

  return 0;
}
```

Output

```
Enter two numbers: 0 20
Prime numbers between 0 and 20 are:
2 3 5 7 11 13 17 19
```

14) Program to Check Armstrong Number

THEORY:

A positive integer is called an Armstrong number (of order n) if

abcd... = a^n + b^n + c^n + d^n + ...

For example, 153 is an Armstrong number because

153 = 1*1*1 + 5*5*5 + 3*3*3

CODE:

```
#include <cmath>
#include <iostream>
using namespace std;
int main() {
  int num, originalNum, remainder, n = 0, result = 0, power;
  cout << "Enter an integer: ";
  cin >> num;

  originalNum = num;

  while (originalNum != 0) {
    originalNum /= 10;
    ++n;
  }
  originalNum = num;
```

```
  while (originalNum != 0) {
    remainder = originalNum % 10;

    // pow() returns a double value
    // round() returns the equivalent int
    power = round(pow(remainder, n));
    result += power;
    originalNum /= 10;
  }

  if (result == num)
    cout << num << " is an Armstrong number.";
  else
    cout << num << " is not an Armstrong number.";
  return 0;
}
```

Output

Enter an integer: 1634
1634 is an Armstrong number.

EXPLANATION:

Step 1: Calculated the number of digits of the entered number
Step 2: The pow() function computes the power of individual digits in each iteration of the while loop.

15) Program to Display Factors of a Number

CODE:

```
#include <iostream>
using namespace std;

int main() {
   int n, i;

   cout << "Enter a positive integer: ";
   cin >> n;

   cout << "Factors of " << n << " are: ";
   for(i = 1; i <= n; ++i) {
      if(n % i == 0)
         cout << i << " ";
   }
```

```
    return 0;
}
```

Output

```
Enter a positive integer: 60
Factors of 60 are: 1 2 3 4 5 6 10 12 15 20 30 60
```