Easy programs

1.Magic number:

```
#include <iostream>

using namespace std;


int main() {

   int n;

   bool isSymmetric = true;


   cout << "Enter the size of the matrix (n x n): ";

   cin >> n;


   int matrix[n][n];


   cout << "Enter the elements of the matrix:\n";

   for (int i = 0; i < n; i++) {

      for (int j = 0; j < n; j++) {

         cin >> matrix[i][j];

      }

   }

   for (int i = 0; i < n; i++) {

      for (int j = 0; j < n; j++) {

         if (matrix[i][j] != matrix[j][i]) {

            isSymmetric = false;

            break;

         }

      }

      if (!isSymmetric) break;
```

```
    }

    if (isSymmetric) {
        cout << "The matrix is symmetric.";
    } else {
        cout << "The matrix is not symmetric.";
    }

    return 0;
}
```

Output:

```
19 is an magic number.
```

2.sum of square series:

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

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

    cout << "Enter the value of n: ";
    cin >> n;

    for (int i = 1; i <= n; i++) {
        sum += i*i;
    }

    cout << "The sum of the series " << n << " is: " << sum ;
    return 0;
}
```

Output:

```
Enter the value of n: 5

The sum of the series 1 + 2 + ... + 5 is: 55
```

3. Right angle triangle:

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

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

    cout << "Enter the value of n: ";
    cin >> n;

    for (int i = 1; i <= n; i++) {
        sum += i*i;
    }

    cout << "The sum of the series " << n << " is: " << sum ;

    return 0;
}
```

Output:

```
*
```

```
**
***
****
*****
```

4.Palindrome pattern:

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

int main() {
    int rows;

    cout << "Enter the number of rows: ";
    cin >> rows;

    for (int i = 1; i <= rows; i++) {
        for (int j = 1; j <= i; j++) {
            cout << j;
        }
        for (int j = i - 1; j >= 1; j--) {
            cout << j;
        }
        cout << "\n";
    }

    return 0;
}
```

Output:

```
1
```

```
121

12321

1234321

123454321
```

5.Sum of even and odd numbers:

```
#include <iostream>

using namespace std;


int main() {

   int n, sumEven = 0, sumOdd = 0;


   cout << "Enter the value of n: ";

   cin >> n;


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

      if (i % 2 == 0) {

         sumEven += i;

      } else {

         sumOdd += i;

      }

   }


   cout << "Sum of even numbers: " << sumEven ;

   cout << "Sum of odd numbers: " << sumOdd ;


   return 0;

}
```

```
Output:
Enter the value of n: 10
Sum of even numbers: 30
Sum of odd numbers: 25
```

Medium:

6.symmetric matrix:

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

int main() {
    int n;
    bool isSymmetric = true;

    cout << "Enter the size of the matrix (n x n): ";
    cin >> n;

    int matrix[n][n];

    cout << "Enter the elements of the matrix:\n";
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] != matrix[j][i]) {
                isSymmetric = false;
                break;
            }
        }
        if (!isSymmetric) break;
    }

    if (isSymmetric) {
        cout << "The matrix is symmetric.";
    } else {
        cout << "The matrix is not symmetric.";
    }

    return 0;
}
```

Output:

```
Enter the size of the matrix (n x n): 3
Enter the elements of the matrix:
1 2 3
2 4 5
3 5 6
The matrix is symmetric.
```

7.structure for student:

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

```
struct Student {
   string name;
   int age;
   float grade;
};
int main() {
   Student student;
   cout << "Enter student's name: ";
   getline(cin, student.name);
   cout << "Enter student's age: ";
   cin >> student.age;
   cout << "Enter student's grade: ";
   cin >> student.grade;
   cout << "\nStudent Information:\n";
   cout << "Name: " << student.name ;
   cout << "Age: " << student.age ;
   cout << "Grade: " << student.grade ;

   return 0;
}
```

Output:

```
Enter student's name: John Doe
Enter student's age: 20
Enter student's grade: 85.5
Student Information:
Name: John Doe
Age: 20
Grade: 85.5
```

Hard

8. subsets of an array that sum up to a target value:

```
#include <iostream>
using namespace std;
bool subsetSum(int arr[], int n, int target) {
    if (target == 0) return true;
    if (n == 0) return false;
    if (arr[n-1] > target) return subsetSum(arr, n-1, target);

    return subsetSum(arr, n-1, target) || subsetSum(arr, n-1, target - arr[n-1]);
}

int main() {
    int n, target;

    cout << "Enter the number of elements in the array: ";
    cin >> n;

    int arr[n];

    cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    cout << "Enter the target sum: ";
    cin >> target;
    if (subsetSum(arr, n, target)) {
        cout << "There is a subset with the given target sum." ;
    } else {
        cout << "There is no subset with the given target sum." ;
    }

    return 0;
}
```

Output:

```
Enter the number of elements in the array: 5
Enter the elements of the array:
3 34 4 12 5
Enter the target sum: 9
There is a subset with the given target sum.
```

Create a program to determine the least common multiple (LCM) of two numbers provided by the user.

Program:

```
#include<iostream>
using namespace std;
int main(){
        int num1,num2,lcm,max;
        cout<<"enter two positive integers:";
        cin>>num1>>num2;
        max=(num1>num2)?num1:num2;
        lcm=max;
        while(true){
                if(lcm%num1==0 && lcm%num2==0){
                        cout<< "LCM of" << num1 << "and" << num2 << "is" << lcm;
                        break;
                }
                ++lcm;
        }
        return 0;
}
```

Output:

```
enter two positive integers:3 4
LCM of3and4is12
```

create a program that prints all factors of a number provided by the user

Program:

```
#include<iostream>
using namespace std;
int main(){
        int n;
        cout<<"enter a positive integer:";
        cin>>n;
        cout<<"factors of"<<n<<"are:";
        for(int i=1;i<=n;i++){
```

```
			if(n%i==0){
				cout<<i<<" ";
			}
		}
		return 0;
}
```

Output:

```
enter a positive integer:20
factors of20are:1 2 4 5 10 20
```

Develop a program that prints the Fibonacci series up to n terms with the value of n provided by the user.

Program:

```
#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: " << t1 << ", " << t2;

   for(int i = 3; i <= n; ++i) {
      nextTerm = t1 + t2;
      cout << ", " << nextTerm;
      t1 = t2;
      t2 = nextTerm;
   }

   return 0;
}
```

OUTPUT:

```
Enter the number of terms: 6
```

Fibonacci Series: 0, 1, 1, 2, 3, 5

Create a program to print the following number pattern

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

PROGRAM:

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

int main() {
   int n = 5;

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

   return 0;
}
```

OUTPUT:

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

5) create a program to print the following pyramid pattern

```
      *
    * * *
   * * * * *
  * * * * * * *
```

PROGRAM:

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

int main() {
    int n = 4;
    for(int i = 1; i <= n; ++i) {
        for(int j = i; j < n; ++j) {
            cout << "  ";
        }
        for(int j = 1; j <= (2*i - 1); ++j) {
            cout << "* ";
        }
        cout << endl;
    }

    return 0;
}
```

OUTPUT:

```
      *
    * * *
  * * * * *
* * * * * * *
```

Create a class Engine with an attribute horsepower and a method start().create another class Transmission with an attribute type.

PROGRAM:

```
#include<iostream>
using namespace std;
class Engine {
public:
    int horsepower;
    void start() {
        cout << "Engine with " << horsepower << " horsepower is starting." << endl;
    }
};
class Transmission {
public:
    string type;
    void displayType() {
        cout << "Transmission type: " << type << endl;
    }
};

int main() {
    Engine engine1;
    engine1.horsepower = 300;
    engine1.start();
    Transmission transmission1;
    transmission1.type = "Automatic";
    transmission1.displayType();

    return 0;
}
```

OUTPUT:

```
Engine with 300 horsepower is starting.
Transmission type: Automatic
```

Create a program that finds the maximum sum of a circular subarray.

PROGRAM:

```
#include<iostream>
#include<vector>
#include<algorithm>
using namespace std;
int kadaneMaxSum(const vector<int>& nums) {
    int maxSum = nums[0], currentSum = nums[0];
    for(size_t i = 1; i < nums.size(); ++i) {
        currentSum = max(nums[i], currentSum + nums[i]);
        maxSum = max(maxSum, currentSum);
    }
    return maxSum;
}
int kadaneMinSum(const vector<int>& nums) {
    int minSum = nums[0], currentSum = nums[0];
    for(size_t i = 1; i < nums.size(); ++i) {
        currentSum = min(nums[i], currentSum + nums[i]);
        minSum = min(minSum, currentSum);
    }
    return minSum;
}
int maxCircularSubarraySum(const vector<int>& nums) {
    int maxKadane = kadaneMaxSum(nums);
    int totalSum = 0;
    for(size_t i = 0; i < nums.size(); ++i) {
        totalSum += nums[i];
    }
    int minKadane = kadaneMinSum(nums)
    int maxCircularSum = totalSum - minKadane;
    if(maxCircularSum == 0) {
        return maxKadane;
    }
   return max(maxKadane, maxCircularSum);
}
int main() {
    int arr[] = {5, -2, 3, 4};
    vector<int> nums(arr, arr + sizeof(arr) / sizeof(arr[0]))
    int result = maxCircularSubarraySum(nums);
```

```
cout << "The maximum sum of a circular subarray is: " << result << endl;

return 0;

}
```

OUTPUT:

```
The maximum sum of a circular subarray is: 12
```

Create a point with constructor overloading to initialize the object with different parameters: x-coordinate ,y-coordinate ,z-coordinate .calculate and print the distance from the origin(0,0,0)

PROGRAM:

```
#include <iostream>
#include <cmath>
class Point {
private:
   double x, y, z;

public:
   Point() : x(0), y(0), z(0) {}
   Point(double xCoord, double yCoord) : x(xCoord), y(yCoord), z(0) {}
   Point(double xCoord, double yCoord, double zCoord) : x(xCoord), y(yCoord), z(zCoord) {}
   double distanceFromOrigin() const {
      return sqrt(x * x + y * y + z * z);
   }
   void print() const {
      std::cout << "Point coordinates: (" << x << ", " << y << ", " << z << ")\n";
      std::cout << "Distance from origin: " << distanceFromOrigin() << "\n";
   }
};

int main() {
   Point p1;
   Point p2(3, 4);
   Point p3(1, 2, 3);
   p1.print();
   p2.print();
   p3.print();
  return 0;
}
```

OUTPUT:

```
Point coordinates: (0, 0, 0)
Distance from origin: 0
Point coordinates: (3, 4, 0)
Distance from origin: 5
Point coordinates: (1, 2, 3)
Distance from origin: 3.74166
```

MODEL EXAMINATION-2

1.write a c++ program to find the smallest and largest digit in a number entered by the user.

```
#include <iostream>

#include <limits>

#include <cmath>


int main()

   long long number;

   int smallest = std::numeric_limits<int>::max();

   int largest = std::numeric_limits<int>::min();


   std::cout << "Enter an integer number: ";

   std::cin >> number;

   number = std::abs(number);

   while (number > 0) {

      int digit = number % 10;

      if (digit < smallest) {

         smallest = digit;

      }

      if (digit > largest) {

         largest = digit;

      }

      number /= 10;

   }

   std::cout << "Smallest digit: " << smallest << std::endl;

   std::cout << "Largest digit: " << largest << std::endl;
```

```
    return 0;
}
```

2.write a c++ program to count the frequency of each digit in a number entered by the user.

```
#include <iostream>
#include <vector>
#include <cmath>  // For std::abs function

int main() {
    // Array to store the frequency of each digit
    std::vector<int> digitFrequency(10, 0);

    // Input number from the user
    long long number;
    std::cout << "Enter an integer number: ";
    std::cin >> number;

    // Handle negative numbers
    number = std::abs(number);

    // Count frequency of each digit
    while (number > 0) {
        int digit = number % 10;  // Extract the last digit
        digitFrequency[digit]++; // Increment the count for this digit
        number /= 10;  // Remove the last digit
    }

    // Display the frequency of each digit
    std::cout << "Digit frequencies:\n";
    for (int i = 0; i < 10; ++i) {
        std::cout << "Digit " << i << ": " << digitFrequency[i] << " times\n";
    }

    return 0;
}
```

3.Write a c++ program to find sum of prime number.

```
#include <iostream>
#include <cmath>  // For std::sqrt function

// Function to check if a number is prime
bool isPrime(int number) {
    if (number <= 1) return false;
    if (number <= 3) return true;
    if (number % 2 == 0 || number % 3 == 0) return false;

    for (int i = 5; i * i <= number; i += 6) {
        if (number % i == 0 || number % (i + 2) == 0)
            return false;
    }
    return true;
}

int main() {
    int N;

    // Input the upper limit
    std::cout << "Enter the upper limit (N): ";
    std::cin >> N;

    // Variable to store the sum of prime numbers
    int sum = 0;

    // Calculate the sum of all prime numbers up to N
    for (int i = 2; i <= N; ++i) {
        if (isPrime(i)) {
            sum += i;
        }
    }

    // Output the result
    std::cout << "The sum of prime numbers up to " << N << " is " << sum << std::endl;

    return 0;
}
```

4.write a c++ program to print hollow square pattern.

```
#include <iostream>

int main() {
    int size;

    // Input the size of the square
    std::cout << "Enter the size of the square: ";
    std::cin >> size;
    // Generate the hollow square pattern
    for (int i = 0; i < size; ++i) {
        for (int j = 0; j < size; ++j) {
            // Print '*' for borders and ' ' for the hollow part
            if (i == 0 || i == size - 1 || j == 0 || j == size - 1) {
                std::cout << "*";
            } else {
                std::cout << " ";
            }
        }
        std::cout << std::endl; // Move to the next line after each row
    }

    return 0;
}
```

5.write a c++ program to print inverted pyramind pattern .

```
#include <iostream>

int main() {
    int height;

    // Input the height of the pyramid
    std::cout << "Enter the height of the inverted pyramid: ";
    std::cin >> height;

    // Generate the inverted pyramid pattern
    for (int i = 0; i < height; ++i) {
        // Print leading spaces
        for (int j = 0; j < i; ++j) {
            std::cout << " ";
        }
        // Print stars
        for (int k = 0; k < (height - i); ++k) {
            std::cout << "*";
        }
        std::cout << std::endl; // Move to the next line after each row
    }

    return 0;
}
```

6.write a c++ program to class vehicle.

```
#include <iostream>
#include <string>

// Define the Vehicle class
class Vehicle {
private:
    std::string make;
    std::string model;
    int year;

public:
    // Constructor to initialize the Vehicle object
    Vehicle(const std::string& make, const std::string& model, int year)
        : make(make), model(model), year(year) {}

    // Getter methods
    std::string getMake() const { return make; }
    std::string getModel() const { return model; }
    int getYear() const { return year; }

    // Method to display vehicle information
    void displayInfo() const {
        std::cout << "Make: " << make << std::endl;
        std::cout << "Model: " << model << std::endl;
        std::cout << "Year: " << year << std::endl;
    }

    // Method to set vehicle information
    void setInfo(const std::string& make, const std::string& model, int year) {
        this->make = make;
        this->model = model;
        this->year = year;
    }
};

int main() {
    // Create a Vehicle object
    Vehicle myCar("Toyota", "Corolla", 2022);

    // Display information about the vehicle
    myCar.displayInfo();

    // Change vehicle information
    myCar.setInfo("Honda", "Civic", 2023);

    // Display updated information about the vehicle
    myCar.displayInfo();

    return 0;
}
```

7.write a c++ program to missing positive numbers.

```
#include <iostream>
#include <vector>
#include <unordered_set>

// Function to find the smallest missing positive integer
int findMissingPositive(const std::vector<int>& nums) {
    std::unordered_set<int> numSet;

    // Insert all positive numbers into the set
    for (int num : nums) {
        if (num > 0) {
            numSet.insert(num);
        }
    }

    // Find the smallest missing positive integer
    int smallestMissing = 1;
    while (numSet.find(smallestMissing) != numSet.end()) {
        ++smallestMissing;
    }

    return smallestMissing;
}

int main() {
    // Input array
    std::vector<int> nums = {3, 4, -1, 1};

    // Find and output the smallest missing positive integer
    int result = findMissingPositive(nums);
    std::cout << "The smallest missing positive integer is: " << result << std::endl;

    return 0;
}
```

8.write a c++ program to power.

```
#include <iostream>

// Function to calculate power using a loop
double power(double base, int exponent) {
    double result = 1.0;
    for (int i = 0; i < exponent; ++i) {
        result *= base;
    }
    return result;
}
```

```cpp
int main() {
    double base;
    int exponent;

    // Input base and exponent
    std::cout << "Enter base: ";
    std::cin >> base;
    std::cout << "Enter exponent: ";
    std::cin >> exponent;

    // Calculate and display the result
    double result = power(base, exponent);
    std::cout << base << "^" << exponent << " = " << result << std::endl;

    return 0;
}
```

Easy program 1. Hollow diamond

Program:

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

Int main() {
   Int n;
   Cout << “Enter the number of rows: “;
   Cin >> n;

   For (int I = 1; I <= n; i++) {
      For (int j = I; j < n; j++) {
         Cout << “ “;
      }
      For (int j = 1; j <= (2 * I – 1); j++) {
         If (j == 1 | | j == (2 * I – 1)) {
            Cout << “*”;
         } else {
            Cout << “ “;
         }
      }
      Cout << endl;
   }

   For (int I = n – 1; I >= 1; i--) {
      For (int j = n; j > I; j--) {
         Cout << “ “;
      }
```

```
    For (int j = 1; j <= (2 * I – 1); j++) {
      If (j == 1 || j == (2 * I – 1)) {
        Cout << “*”;
      } else {
        Cout << “ “;
      }
    }
    Cout << endl;
  }

  Return 0;
}.
```

Easy program 2

Automorphic number

Program :

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

Bool isAutomorphic(int num) {
  Int square = num * num;

  While (num > 0) {
    If (num % 10 != square % 10) {
      Return false;
    }
    Num /= 10;
    Square /= 10;
  }
```

```
   Return true;
}

Int main() {
   Int num;
   Cout << "Enter a number: ";
   Cin >> num;

   If (isAutomorphic(num)) {
      Cout << num << " is an automorphic number." << endl;
   } else {
      Cout << num << " is not an automorphic number." << endl;
   }

   Return 0;
}
```

Easy program 3

Perfect number :

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

Bool isPerfect(int num) {
   Int sum = 0;
   For (int I = 1; I <= num / 2; i++) {
      If (num % I == 0) {
         Sum += I;
      }
   }
   Return sum == num;
}

Int main() {
   Int num;
   Cout << "Enter a number: ";
   Cin >> num;

   If (isPerfect(num)) {
      Cout << num << " is a perfect number." << endl;
   } else {
      Cout << num << " is not a perfect number." << endl;
   }

   Return 0;
}
```

Easy program 4

Pyramid pattern

Program:

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

Int main() {
   Int n;
   Cout << "Enter the number of rows: ";
   Cin >> n;

   For (int I = 1; I <= n; i++) {
      For (int j = I; j < n; j++) {
         Cout << " ";
```

```
        }
        For (int j = 1; j <= (2 * I – 1); j++) {
            Cout << "*";
        }
        Cout << endl;
    }

    Return 0;
}
```

Easy program 5

Polindrome pattern

Program :

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

Int main() {
    Int n;
    Cout << "Enter the number of rows: ";
    Cin >> n;

    For (int I = 1; I <= n; i++) {
        For (int j = I; j < n; j++) {
            Cout << " ";
        }

        For (int j = 1; j <= I; j++) {
            Cout << j;
        }
        For (int j = I – 1; j >= 1; j--) {
            Cout << j;
        }

        Cout << endl;
    }

    Return 0;
}
```

Medium

Sorted array ascending order

Program:

```
#include <iostream>
#include <algorithm>
Using namespace std;

Int main() {
    Int n;
    Cin >> n;

    Int arr[n];

    For (int I = 0; I < n; i++) {
        Cin >> arr[i];
    }

    Sort(arr, arr + n);

    For (int I = 0; I < n; i++) {
        Cout << arr[i] << " ";
    }

    Return 0;
}
```

Medium -2

Reverse string

```
#include <iostream>
#include <string>
Using namespace std;

Int main() {
    String str;
    Getline(cin, str);

    Int left = 0;
    Int right = str.length() – 1;

    While (left < right) {
        Swap(str[left], str[right]);
        Left++;
        Right--;
    }

    Cout << str << endl;

    Return 0;
}
```

Hard 1

Armstrong number using recursion

Program:

```
#include<iostream>

Using namespace std;

Int ams(int n){

            Int r,sum=0;

            Int a=n;

            While(n>0){

                        R=n%10;

                        Sum=sum+(r*r*r);

                        N=n/10;

            }

            Return sum;

}

Int main(){

            Int n;

            Cin>>n;

            Int number=ams(n);

            If(number==n){

                        Cout<<”amstrong”;

            }

            Else{

                        Cout<<”not amstrong”;

            }

}
```

MODEL ASSESMENT 2

1.odd or even

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

int main() {
    int num;
    cout << "Enter an integer: ";
    cin >> num;

    // Check if the number is even or odd
    if (num % 2 == 0) {
        cout << num << " is even." << endl;
    } else {
        cout << num << " is odd." << endl;
    }

    return 0;
}
```

Output

```
Enter an integer: 7
7 is odd
```

2.binary to decimal

```
#include <iostream>
#include <string>
```

```
#include <cmath> // For pow function

using namespace std;

int binaryToDecimal(const string& binaryStr) {
    int decimalValue = 0;
    int length = binaryStr.length();

    // Process each bit of the binary string
    for (int i = 0; i < length; ++i) {
        // Convert character '0' or '1' to integer
        int bit = binaryStr[length - 1 - i] - '0';
        // Calculate its decimal value and add to result
        decimalValue += bit * pow(2, i);
    }

    return decimalValue;
}

int main() {
    string binaryStr;

    // Prompt the user to enter a binary number
    cout << "Enter a binary number: ";
    cin >> binaryStr;

    // Convert binary to decimal
    int decimalValue = binaryToDecimal(binaryStr);
    // Print the result
    cout << "The decimal value is: " << decimalValue << endl;

    return 0;
}
```

Output

```
Enter a binary number: 1011
The decimal value is: 11
```

3.inverted pyramid

```
#include<iostream>
using namespace std;
int main()
{
        int n;
        cin>>n;
        for(int i=n-2;i>0;i--){
                for(int j=0;j<n-i-1;j++){
                        cout<<" ";
                }
                for(int k=0;k<2*i-1;k++){
                        cout<<"*";
                }
                cout<<"\n";
        }
}
```

Output

```
*************
  ***********
   *********
    *******
     *****
      ***
       *
```

4.amstrong using recursion

```
#include <iostream>
#include <cmath> // For pow and log10 functions

using namespace std;

// Function to calculate the number of digits
int countDigits(int num) {
    if (num == 0) return 1; // To handle zero
    return log10(num) + 1;
}

// Recursive function to compute the sum of digits raised to the power of digit count
int armstrongSum(int num, int power) {
    if (num == 0) return 0;
    int digit = num % 10;
    return pow(digit, power) + armstrongSum(num / 10, power);
}

// Function to check if a number is an Armstrong number
```

```
bool isArmstrong(int num) {
   int digits = countDigits(num);
   return num == armstrongSum(num, digits);
}

int main() {
   int num;

   // Prompt the user to enter a number
   cout << "Enter a number: ";
   cin >> num;

   // Check if the number is an Armstrong number
   if (isArmstrong(num)) {
      cout << num << " is an Armstrong number." << endl;
   } else {
      cout << num << " is not an Armstrong number." << endl;
   }

   return 0;
}
```

Output

```
153 is an Armstrong number
```

5.hollow square

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

```
int main() {
    int n;
    cin >> n;

    // Loop over rows
    for (int i = 0; i < n; i++) {
        // Loop over columns
        for (int j = 0; j < n; j++) {
            // Check if we are on the border
            if (i == 0 || i == n - 1 || j == 0 || j == n - 1) {
                cout << "*";
            } else {
                cout << " ";
            }
        }
        // Move to the next line after printing each row
        cout << endl;
    }

    return 0;
}
```

Output

```
*****
*   *
*   *
*   *
*****
```

6.sum of positive and negative numbers in array

```
#include<iostream>

using namespace std;

int main()

{

        int n;

        cin>>n;

        int arr[n];

        for(int i=0;i<n;i++){

                cin>>arr[i];

        }

        int sum=0,count=0;

        for(int j=0;j<n;j++){

                if(arr[j]>0){

                        sum=sum+arr[j];

                }

        }

        for(int k=0;k<n;k++){

                if(arr[k]<0){

                        count=count+arr[k];

                }

        }

        cout<<sum;

        cout<<count;

}
```

Output

```
5

1 -3 5 4 6

16-3
```

7.frequency of each element in array

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

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter the elements: ";
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }

    // Array to keep track of elements already counted
    bool counted[n] = {false};

    for (int i = 0; i < n; i++) {
        if (!counted[i]) {
            int count = 1;
            for (int j = i + 1; j < n; j++) {
                if (arr[i] == arr[j]) {
                    count++;
                    counted[j] = true; // Mark this element as counted
                }
            }
            // Print frequency of the current element
            cout << "Element " << arr[i] << " appears " << count << " times" << endl;
        }
    }

    return 0;
}
```

Output

```
1 2 2 3 3 3
1 1 times
2 2 times
3 3 times
```

8.calculates the digital root of a given number

```
#include <iostream>
using namespace std;
int digitalRoot(int n)
{
    int sum = 0;
    while (n != 0)
              {
        sum += n % 10;
        n /= 10;
    }
    if (sum > 9)
              {
        return digitalRoot(sum);
    } else
```

```
            {
        return sum;
    }
}

int main()
 {
    int num;
    cout << "Enter a number: ";
    cin >> num;
    cout << "Digital root of " << num << " is " << digitalRoot(num);
    return 0;
}
```

Output

```
Enter a number: 123456
Digital root (iterative method):3
  Digital root (modulo method): 3
```