Q.1 WAP to find given number is even or odd.

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

int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;

    cout << ((n % 2 == 0) ? "Even" : "Odd") << endl;
    return 0;
}
```

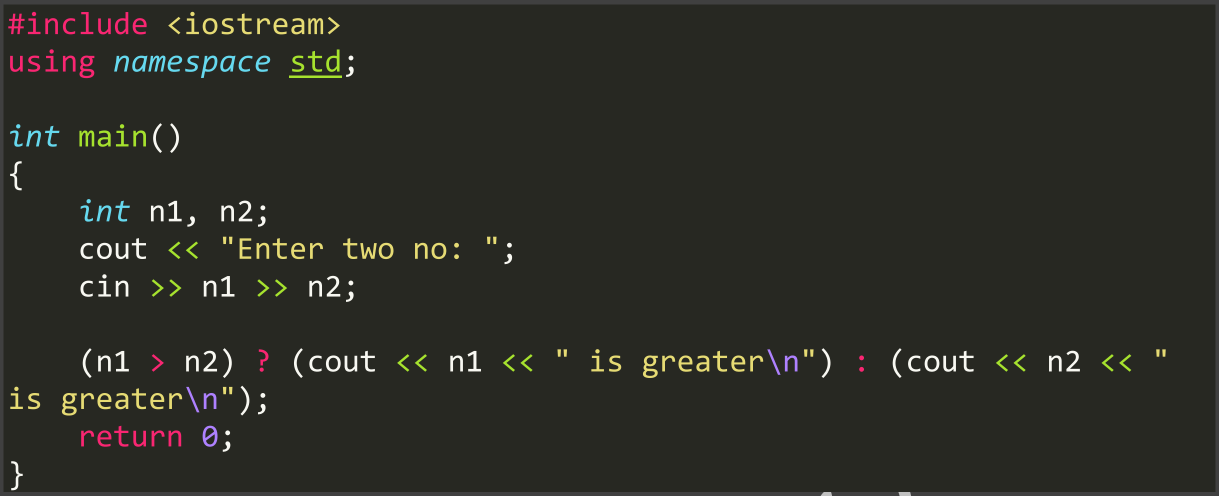
Q.2 WAP to find given number is prime or composite.

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

int main()
{
    int n;
    cout << "Enter n: ";
    cin >> n;

    bool is_prime = true;
    for (int i = 2; i * i <= n; i++)
    {
        if (n % i == 0)
 {
            is_prime = false;
            break;
        }
    }

    if (is_prime)
    {
        cout << "Prime";
    }
    else
    {
        cout << "Composite";
    }

    return 0;
}
```

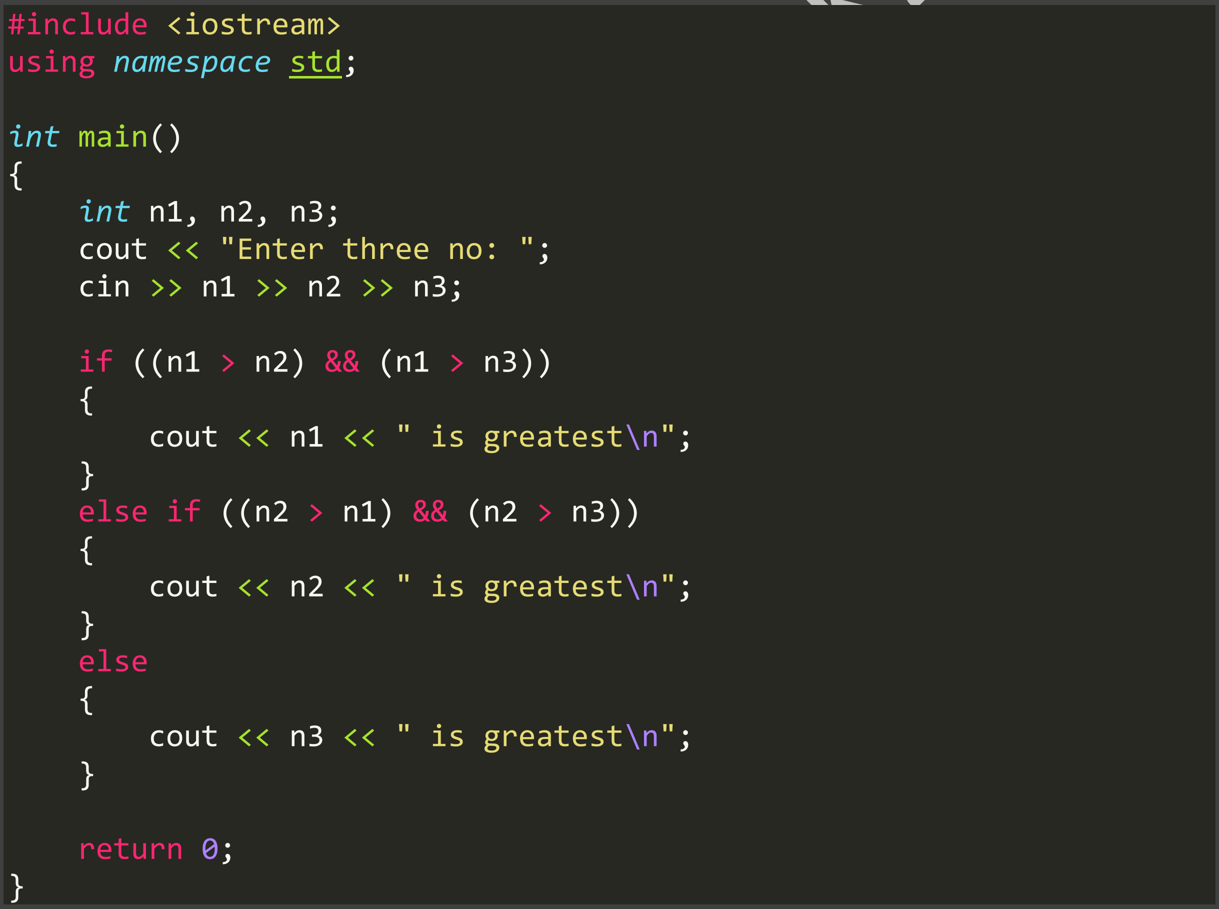
Q.3 WAP to print table of a given number up to 'n' multiples.

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

int main()
{
    int num, mul;
    cout << "Enter number and its multiples: ";
    cin >> num >> mul;

    for (int i = 1; i <= mul; i++)
    {
        cout << num << " X " << i << " = " << (num * i) << endl;
    }

    return 0;
}
```

Q.4 WAP to find
i) greater of two numbers.
ii) greatest of three numbers.



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

int main()
{
    int n1, n2;
    cout << "Enter two no: ";
    cin >> n1 >> n2;

    (n1 > n2) ? (cout << n1 << " is greater\n") : (cout << n2 << "
is greater\n");
    return 0;
}
```



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

int main()
{
    int n1, n2, n3;
    cout << "Enter three no: ";
    cin >> n1 >> n2 >> n3;

    if ((n1 > n2) && (n1 > n3))
    {
        cout << n1 << " is greatest\n";
    }
    else if ((n2 > n1) && (n2 > n3))
    {
        cout << n2 << " is greatest\n";
    }
    else
    {
        cout << n3 << " is greatest\n";
    }

    return 0;
}
```

Q.5 WAP to find sum of first 'n' natural numbers.

```cpp
#include <iostream>

using namespace std;

int main()
{
    int n, sum = 0;
    cout << "Enter n: ";
    cin >> n;

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

    cout << "sum of first " << n << " natural no.s is: " << sum << endl;

    return 0;
}
```

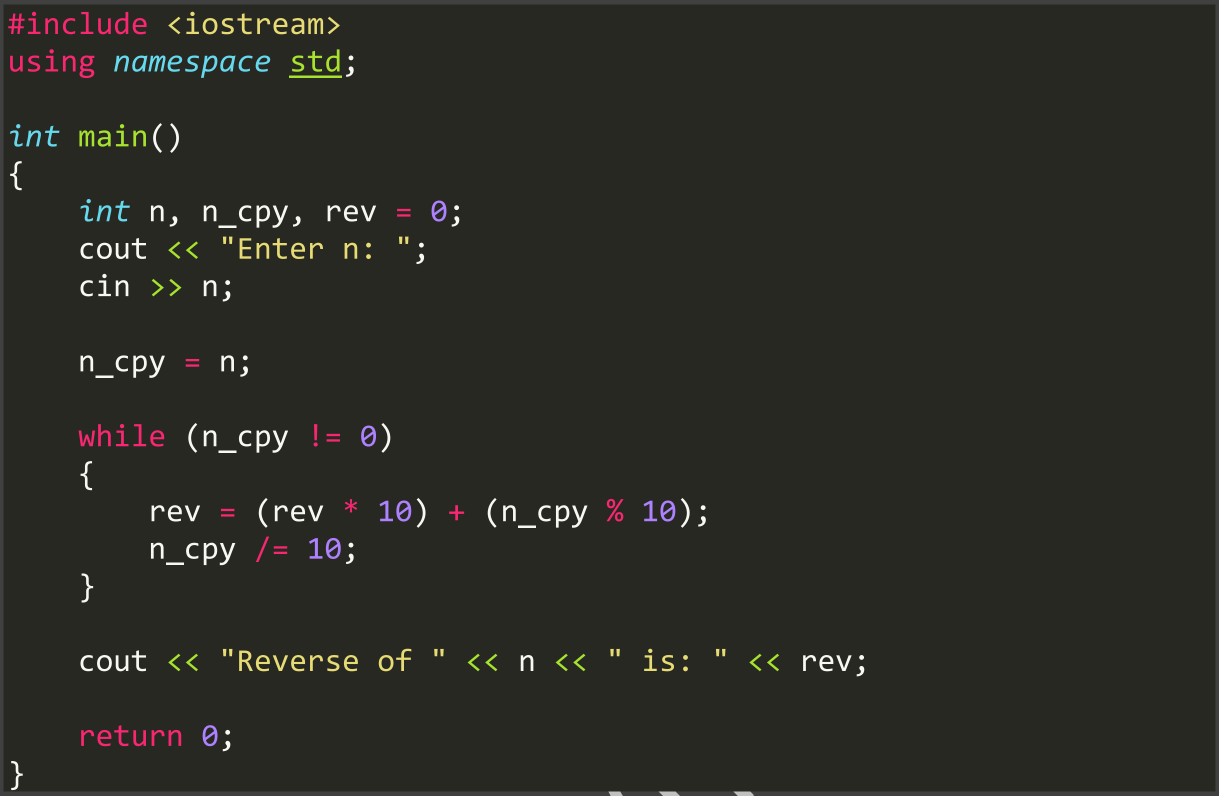
Q.6 WAP to find factorial of given number.

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

int main()
{
    int n = 5;
    long long fact = 1;

    while (n >= 1)
    {
        fact *= n--;
    }

    cout << "Factorial of " << n << " is: " << fact << endl;

    return 0;
}
```

Q.7 WAP to find sum of digits of 'n' digit number.

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

int digitSum(int n)
{
    int sum = 0;

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

int main()
{
    int n = 1212;
    cout << "Sum of digits of " << n << " is: " << digitSum(n);
    return 0;
}
```

Q.8 WAP to find reverse of a number.

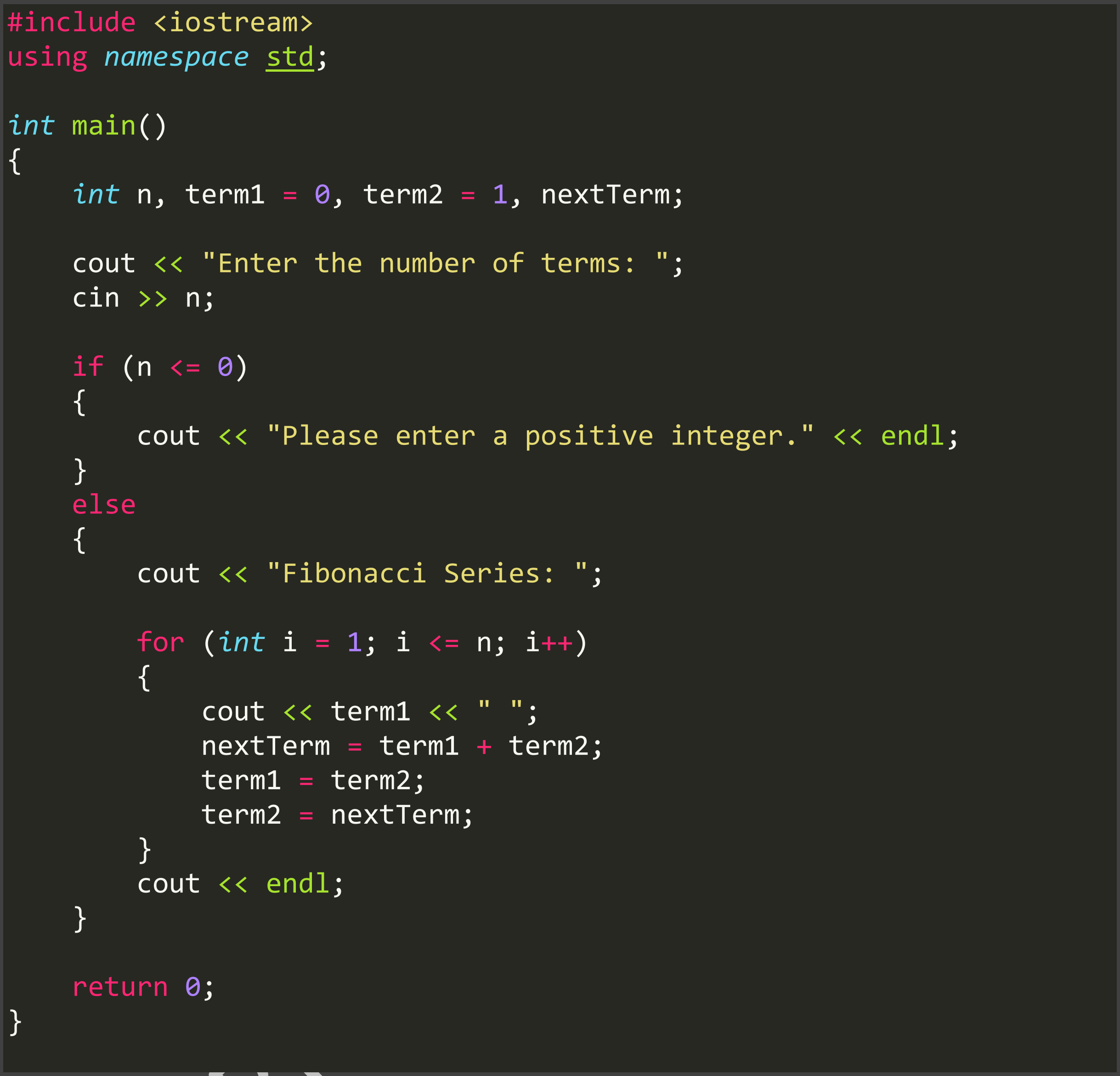


```
#include <iostream>
using namespace std;

int main()
{
    int n, n_cpy, rev = 0;
    cout << "Enter n: ";
    cin >> n;

    n_cpy = n;

    while (n_cpy != 0)
    {
        rev = (rev * 10) + (n_cpy % 10);
        n_cpy /= 10;
    }

    cout << "Reverse of " << n << " is: " << rev;

    return 0;
}
```

Q.9 WAP to determine given number is palindrome or not.

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

int main()
{
    int n = 12321;
    int n_copy = n, n_rev = 0;

    while (n_copy != 0)
    {
        // n_rev *= 10;
        // n_rev += (n_copy % 10);
        n_rev = (n_rev * 10) + (n_copy % 10);
        n_copy /= 10;
    }

    cout << "n_rev: " << n_rev << '\n';
    if (n_rev == n)
    {
        cout << "Palindrome\n";
    }

    return 0;
}
```

Q.10 WAP to print Fibonacci series up to 'n' terms.



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

int main()
{
    int n, term1 = 0, term2 = 1, nextTerm;

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

    if (n <= 0)
    {
        cout << "Please enter a positive integer." << endl;
    }
    else
    {
        cout << "Fibonacci Series: ";

        for (int i = 1; i <= n; i++)
        {
            cout << term1 << " ";
            nextTerm = term1 + term2;
            term1 = term2;
            term2 = nextTerm;
        }
        cout << endl;
    }

    return 0;
}
```

Q.11 WAP to determine given 'n' digit number is Armstrong number or not.

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

int main()
{
    int n, n_cpy, arm = 0;
    cout << "Enter n: ";
    cin >> n;
    n_cpy = n;

    int no_of_digit = 0;
    while (n_cpy != 0)
    {
        no_of_digit++;
        n_cpy /= 10;
    }
    n_cpy = n;

    while (n_cpy != 0)
    {
        arm += round(pow((n_cpy % 10), no_of_digit));
        n_cpy /= 10;
    }
    if (n == arm)
    {
        cout << "Armstrong\n";
    }
    else
    {
        cout << "Not Armstrong\n";
    }

    return 0;
}
```

Q.12 WAP to print all even numbers between 100 & 200.

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

int main()
{
    cout << "Even numbers b/w 100 and 200 are: ";
    for (int i = 100; i <= 200; i++)
    {
        if (i % 2 == 0)
        {
            cout << i << " ";
        }
    }

    return 0;
}
```

Q.13 WAP to print first 50 prime numbers.

```
#include <iostream>

using namespace std;

int main()
{
    int count = 0, i = 2;

    cout << "First 50 primes are: \n";
    while (count < 50)
    {
        bool is_i_prime = true;
        for (int j = 2; j < i; j++)
        {
            if (i % j == 0)
            {
                is_i_prime = false;
                break;
            }
        }

        if (is_i_prime)
        {
            cout << i << " ";
            count++;
        }
        i++;
    }

    return 0;
}
```

Q.14 WAP to print all 4-digit Armstrong numbers.

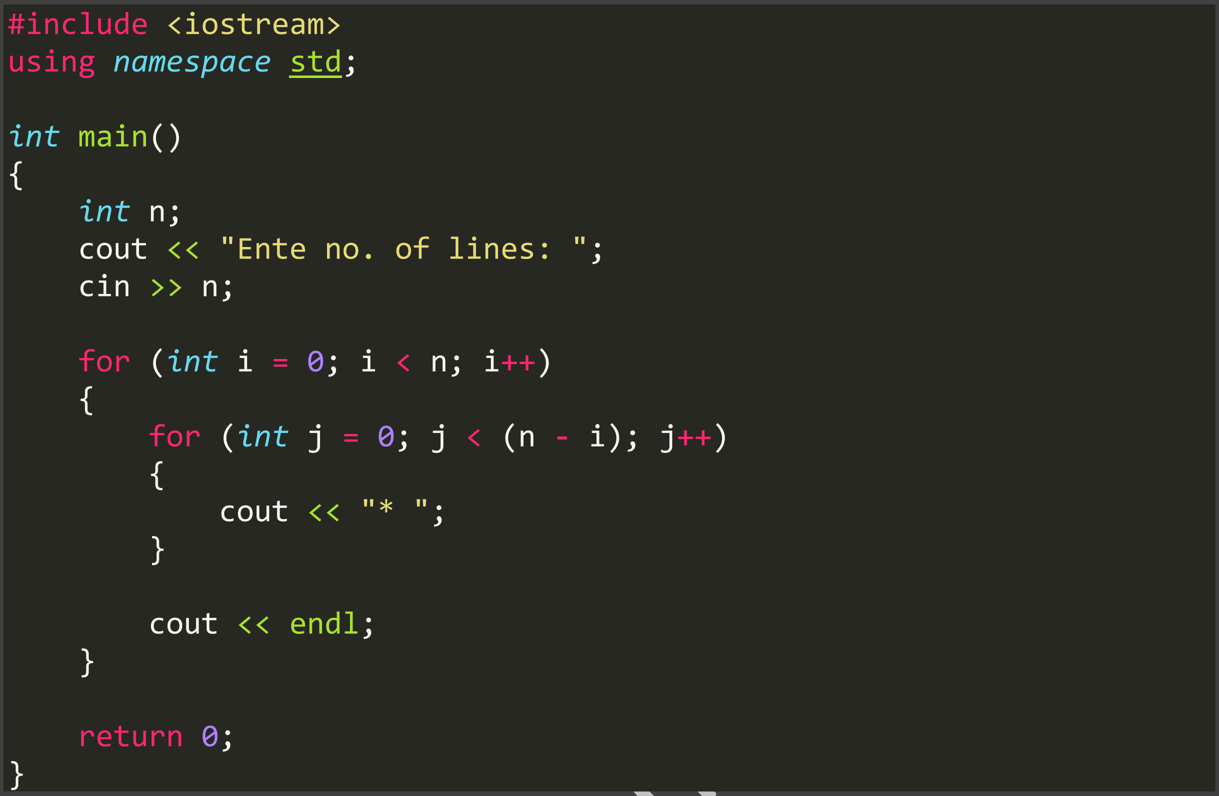
```
#include <iostream>
using namespace std;

int main()
{
    for (int i = 1000; i <= 9999; i++)
    {
        int arm = 0, i_cpy = i;

        while (i_cpy != 0)
        {
            arm += ((i_cpy % 10) * (i_cpy % 10) * (i_cpy % 10) *
(i_cpy % 10));
            i_cpy /= 10;
        }

        if (arm == i)
        {
            cout << "arm: " << i << endl;
            // cout << i << " ";
        }
    }

    return 0;
}
```

Q.15 WAP to print following patterns:

i) (Pattern of left-aligned triangle of stars)

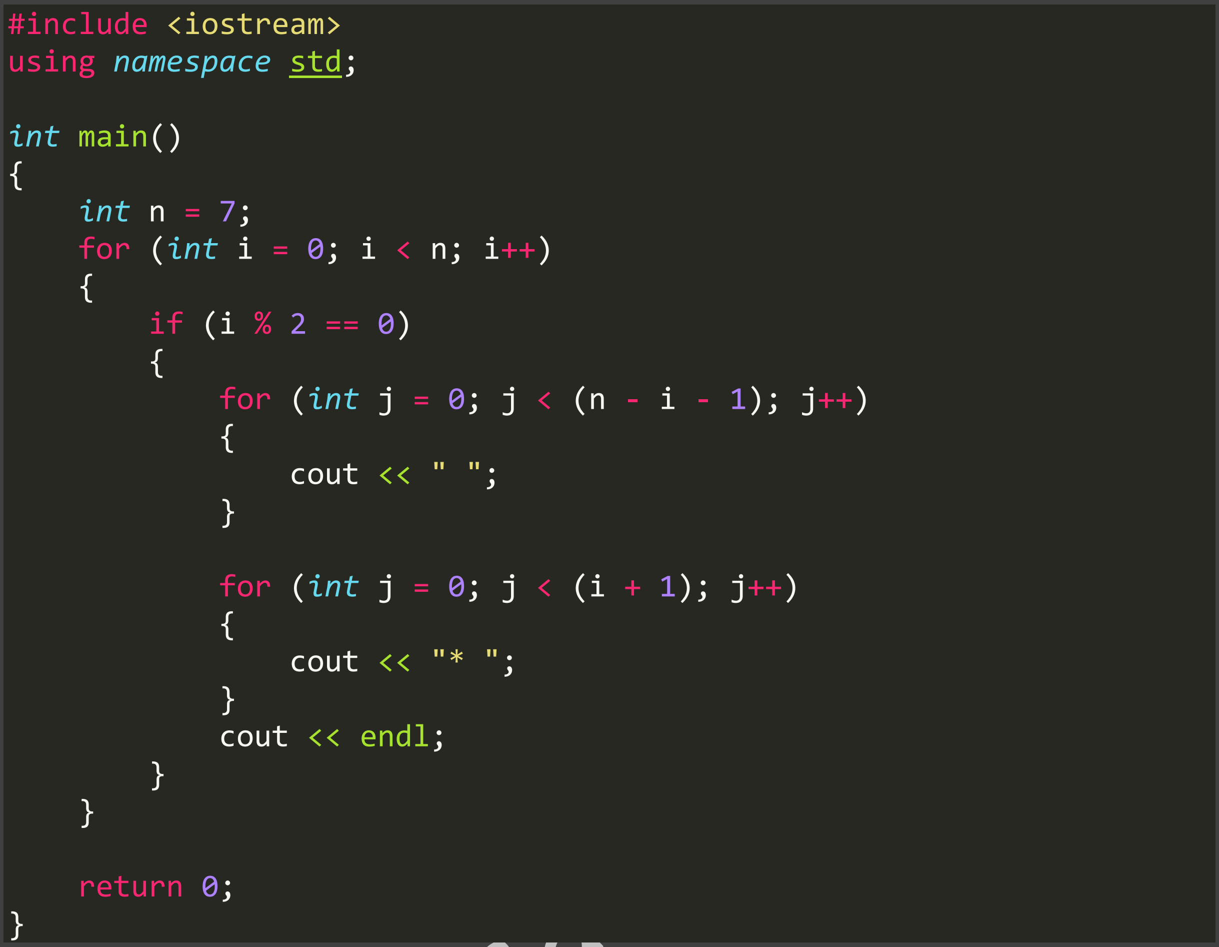
```
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Ente no. of lines: ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < (i + 1); j++)
        {
            cout << "* ";
        }

        cout << endl;
    }

    return 0;
}
```

ii) (Pattern of right-aligned triangle of stars)



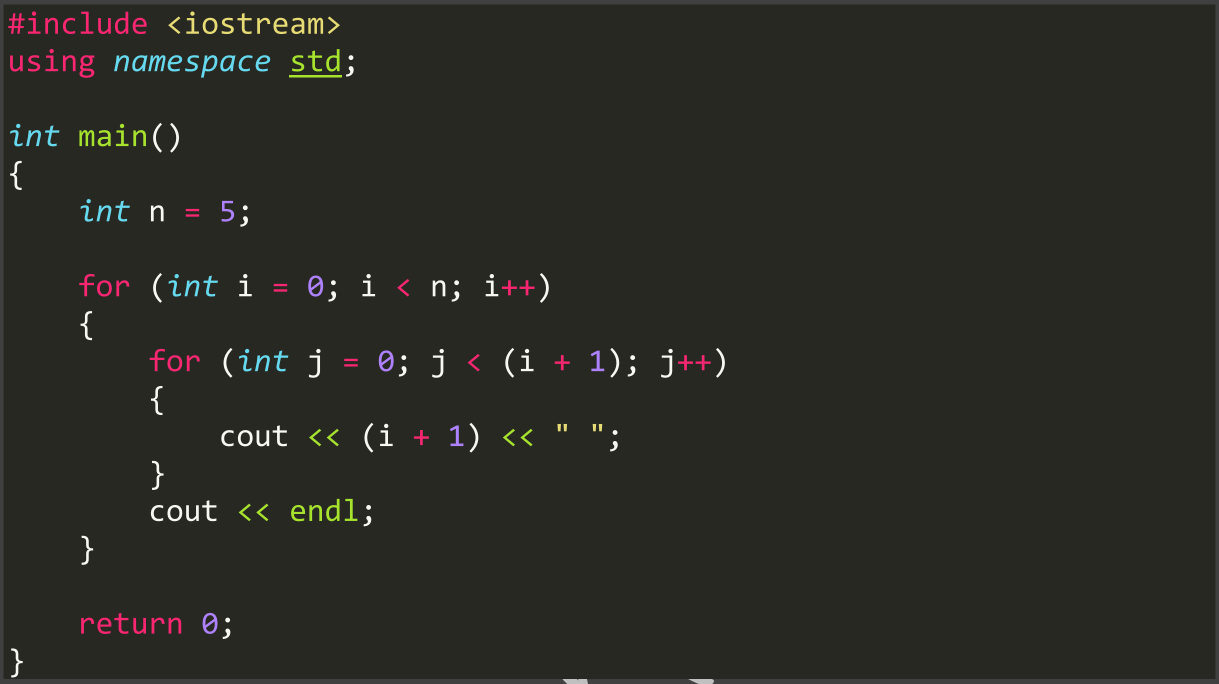
```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Ente no. of lines: ";
    cin >> n;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < (n - i); j++)
        {
            cout << "* ";
        }

        cout << endl;
    }

    return 0;
}
```

iii) (Pyramid pattern of stars)



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

int main()
{
    int n = 7;
    for (int i = 0; i < n; i++)
    {
        if (i % 2 == 0)
        {
            for (int j = 0; j < (n - i - 1); j++)
            {
                cout << " ";
            }

            for (int j = 0; j < (i + 1); j++)
            {
                cout << "* ";
            }
            cout << endl;
        }
    }

    return 0;
}
```

iv)
1
22
333
4444



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

int main()
{
    int n = 5;

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < (i + 1); j++)
        {
            cout << (i + 1) << " ";
        }
        cout << endl;
    }

    return 0;
}
```

v) Pascal’s triangle

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

int main()
{
    int n = 5;

    for (int i = 0; i < n; i++)
    {
        int coeff = 1;
        for (int j = 0; j < (n - i - 1); j++)
        {
            cout << " ";
        }

        for (int j = 0; j < (i + 1); j++)
        {
            cout << coeff << ' ';
            coeff = coeff * (i - j) / (j + 1);
        }

        cout << endl;
    }

    return 0;
}
```

vi) Floyd’s triangle

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

int main()
{
    int n;
    cout << "Ente no. of lines: ";
    cin >> n;

    int count = 1;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < (i + 1); j++)
        {
            cout << count << ' ';
            count++;
        }

        cout << endl;
    }

    return 0;
}
```

Q.16 Using functions, write following C++ programs:

i) To print all palindromes for a range 500-1000.

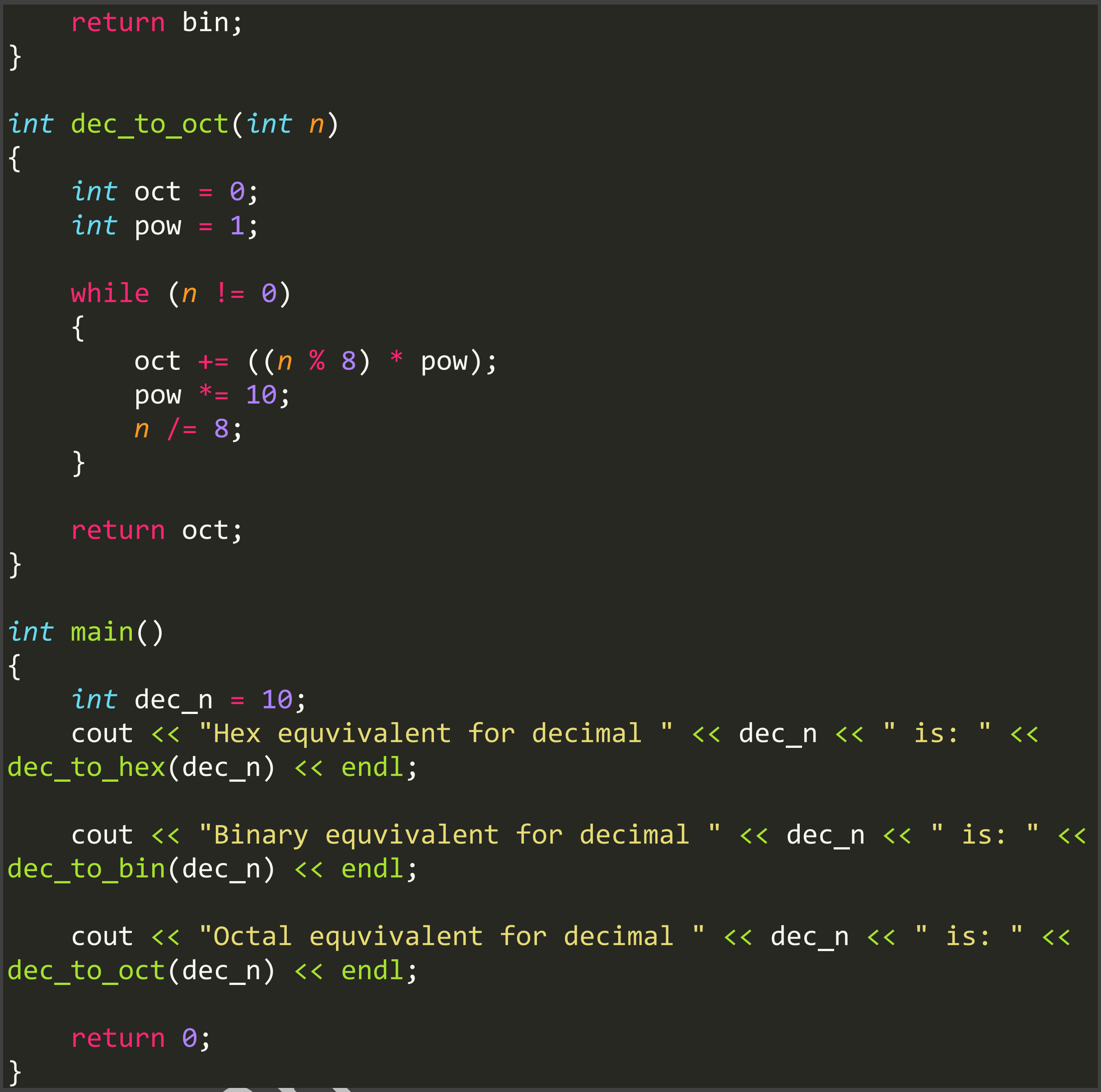
```cpp
#include <iostream>
using namespace std;

void pali_500_to_1000()
{
    cout << "Pali form 500 to 1000 are: \n";
    for (int i = 500; i <= 1000; i++)
    {
        int rev = 0, i_cpy = i;
        while (i_cpy != 0)
        {
            rev = (rev * 10) + (i_cpy % 10);
            i_cpy /= 10;
        }
        if (rev == i)
        {
            cout << i << " ";
        }
    }
}

int main()
{
    pali_500_to_1000();

    return 0;
}
```

ii) To print first 100 odd numbers.

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

void first_100_odd()
{
    int count = 0;
    cout << "First 100 odd no.s are: ";

    int i = 1;
    while (count < 100)
    {
        cout << (i) << " ";
        i += 2;
        count++;
    }
}

int main()
{
    first_100_odd();
    return 0;
}
```

iii) To find binary, octal, hexadecimal equivalent of a given decimal number.

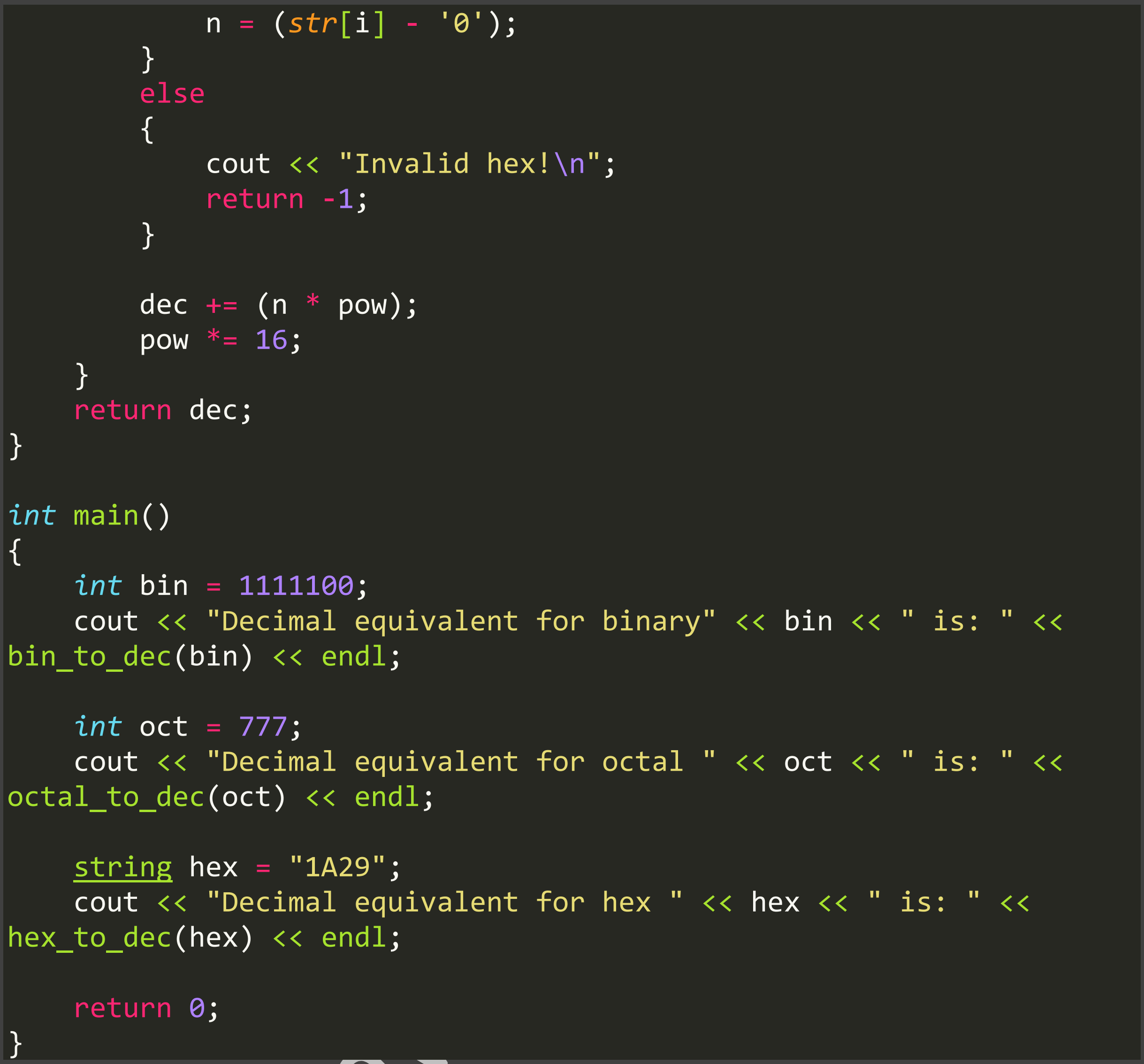
```
#include <iostream>
using namespace std;

string dec_to_hex(int n)
{
    string hex;

    int hex_len = 0;
    int n_cpy = n;
    while (n_cpy != 0)
    {
        hex_len++;
        n_cpy /= 16;
    }
    hex.resize(hex_len);

    while (n != 0)
    {
        if (((n % 16) >= 10) && ((n % 16) <= 15))
        {
            hex[--hex_len] = (char)((n % 16) + 55);
        }
        else if (((n % 16) >= 0) && ((n % 16) <= 9))
        {
            hex[--hex_len] = (char)((n % 16) + '0');
        }
        else
        {
            return "Negative Decimal no. can't be converted into 
Hex\n";
        }

        n /= 16;
    }
    return hex;
}

int dec_to_bin(int n)
{
    int bin = 0;
    int pow = 1;
    while (n != 0)
    {
        bin += ((n % 2) * pow);
        pow *= 10;
        n /= 2;
    }
```



```
    return bin;
}

int dec_to_oct(int n)
{
    int oct = 0;
    int pow = 1;

    while (n != 0)
    {
        oct += ((n % 8) * pow);
        pow *= 10;
        n /= 8;
    }

    return oct;
}

int main()
{
    int dec_n = 10;
    cout << "Hex equvivalent for decimal " << dec_n << " is: " <<
dec_to_hex(dec_n) << endl;

    cout << "Binary equvivalent for decimal " << dec_n << " is: " <<
dec_to_bin(dec_n) << endl;

    cout << "Octal equvivalent for decimal " << dec_n << " is: " <<
dec_to_oct(dec_n) << endl;

    return 0;
}
```

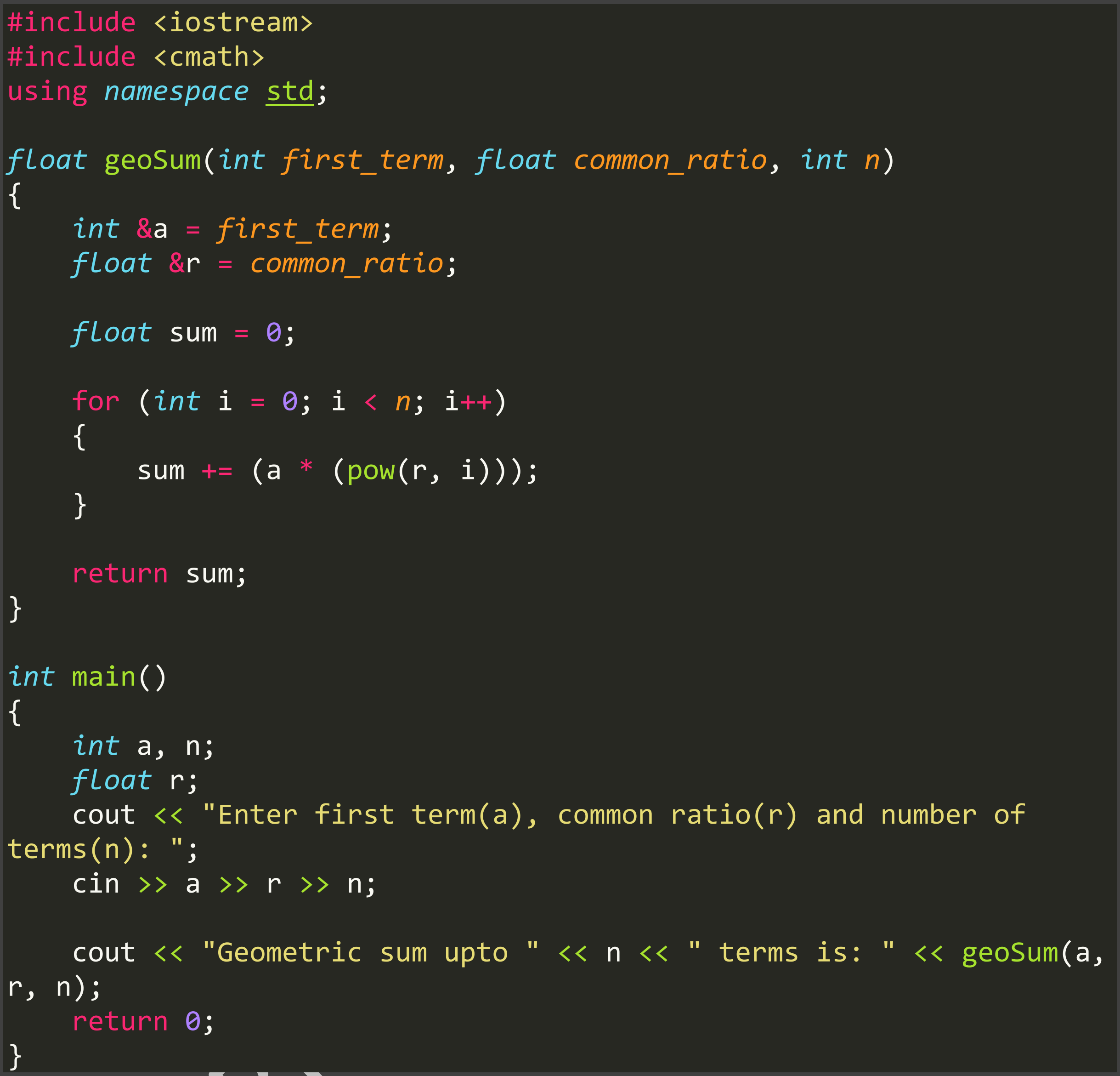
iv) To find decimal equivalents for given binary, hexadecimal & octal numbers.

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

int bin_to_dec(int n)
{
    int dec = 0;
    int pow = 1;
    while (n != 0)
    {
        dec += (n % 10) * pow;
        pow *= 2;
        n /= 10;
    }

    return dec;
}

int octal_to_dec(int n)
{
    int dec = 0;
    int pow = 1;
    while (n != 0)
    {
        dec += (n % 10) * pow;
        pow *= 8;
        n /= 10;
    }

    return dec;
}

int hex_to_dec(string str)
{
    int l = str.length();

    int dec = 0;
    int pow = 1;
    int n;

    for (int i = (l - 1); i >= 0; i--)
    {
        if ((str[i] >= 'A') && (str[i] <= 'F'))
        {
            n = 10 + (str[i] - 'A');
        }
        else if ((str[i] >= '0') && (str[i] <= '9'))
        {
            // Converting char to int!
```



```
            n = (str[i] - '0');
        }
        else
        {
            cout << "Invalid hex!\n";
            return -1;
        }

        dec += (n * pow);
        pow *= 16;
    }
    return dec;
}

int main()
{
    int bin = 1111100;
    cout << "Decimal equivalent for binary" << bin << " is: " <<
bin_to_dec(bin) << endl;

    int oct = 777;
    cout << "Decimal equivalent for octal " << oct << " is: " <<
octal_to_dec(oct) << endl;

    string hex = "1A29";
    cout << "Decimal equivalent for hex " << hex << " is: " <<
hex_to_dec(hex) << endl;

    return 0;
}
```

v) To calculate geometric sum up to 'n' terms.



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

float geoSum(int first_term, float common_ratio, int n)
{
    int &a = first_term;
    float &r = common_ratio;

    float sum = 0;

    for (int i = 0; i < n; i++)
    {
        sum += (a * (pow(r, i)));
    }

    return sum;
}

int main()
{
    int a, n;
    float r;
    cout << "Enter first term(a), common ratio(r) and number of 
terms(n): ";
    cin >> a >> r >> n;

    cout << "Geometric sum upto " << n << " terms is: " << geoSum(a, 
r, n);
    return 0;
}
```

Q.17 Using recursion, write C++ program for:

i) Print binary number for a decimal number.

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

int bin(int n)
{
    if (n == 0)
    {
        return 0;
    }
    else
    {
        return (n % 2) + 10 * bin(n / 2);
    }
}

int main()
{
    int n = 40;
    cout << "binary equivalent of " << n << " is: " << bin(n) <<
endl;

    return 0;
}
```

ii) Print octal number for a decimal number.

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

int octal(int n)
{
    if (n == 0)
    {
        return 0;
    }
    else
    {
        return (n % 8) + 10 * octal(n / 8);
    }
}

int main()
{
    int n = 40;
    cout << "octal equivalent of " << n << " is: " << octal(n) <<
endl;

    return 0;
}
```

iii) Print factorials for a given range.

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

long long fact(int n)
{
    if (n == 0)
    {
        return 1;
    }
    return n * fact(n - 1);
}

int main()
{
    int n = 20;
    for (int i = 1; i <= n; i++)
    {
        cout << fact(i) << " ";
    }

    return 0;
}
```

iv) Print first 'n' terms of Fibonacci series.

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

int fib(int n)
{
    if (n == 1 || n == 2)
    {
        return (n - 1);
    }
    return fib(n - 1) + fib(n - 2);
}

int main()
{
    int no_of_terms = 20;
    cout << "Fib series for " << no_of_terms << " terms is: ";

    for (int i = 1; i <= no_of_terms; i++)
    {
        cout << fib(i) << " ";
    }

    return 0;
}
```

Q.18 WAP to calculate average of all elements of 1D array.

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

int main()
{
    int size, sum = 0;
    cout << "Enter size of array: ";
    cin >> size;

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

    float avg = (float(sum) / size);

    cout << "Average of all elements of array is: " << avg << endl;

    return 0;
}
```

Q.19 WAP to find out minimum & maximum value of a 1D numeric array.

```
#include <iostream>

using namespace std;

int main()
{
    int size;
    cout << "Enter size of array: ";
    cin >> size;

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

    int max, min;
    max = min = arr[0];

    for (int i = 0; i < size; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
        if (arr[i] < min)
        {
            min = arr[i];
        }
    }
    cout << "Maximum element: " << max << endl;
    cout << "Minimum element: " << min << endl;

    return 0;
}
```

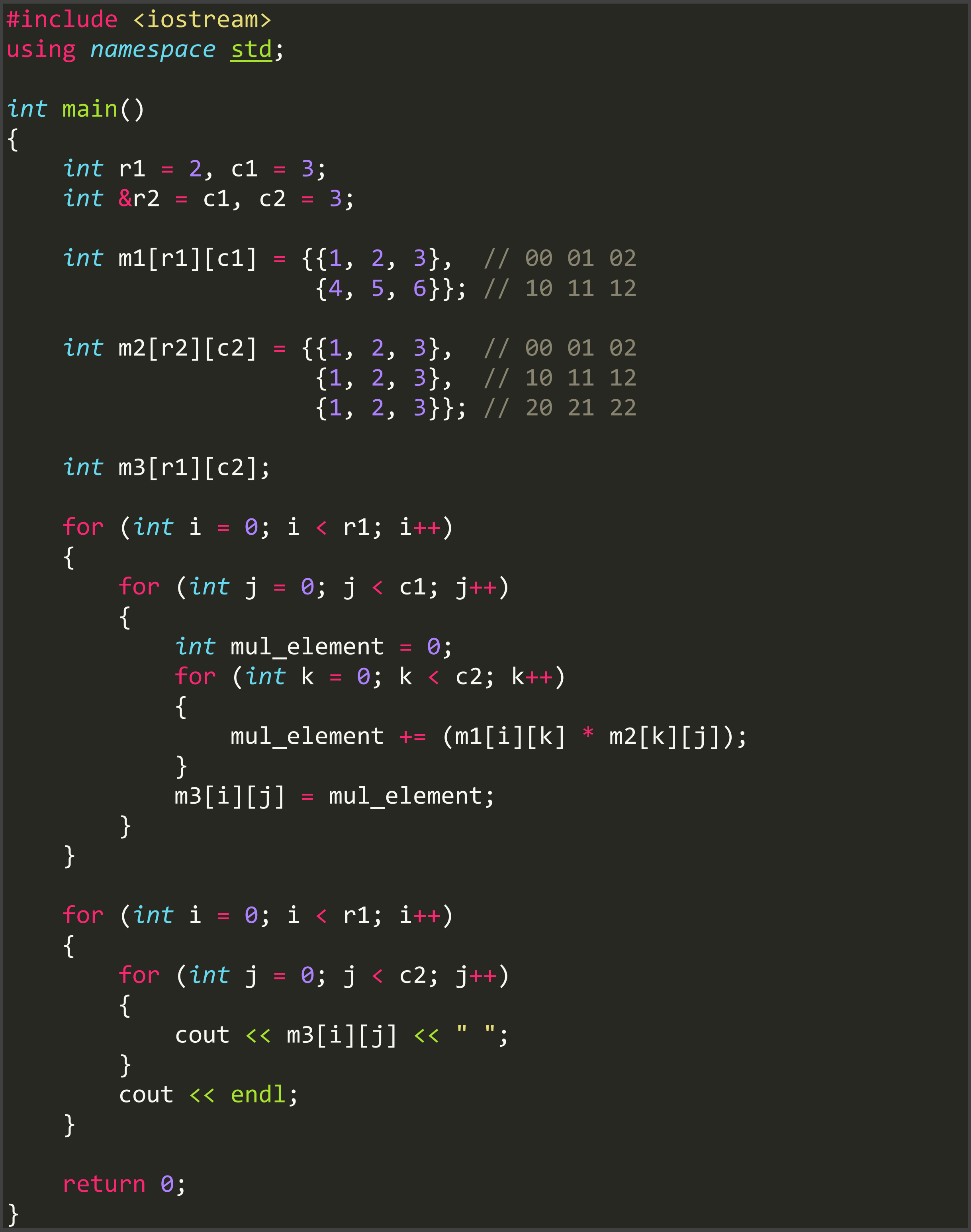
Q,20 WAP to find transpose of a 2D matrix.

```
#include <iostream>

using namespace std;

int main()
{
    int rows = 2, cols = 3;
    int mat[rows][cols] = {{1, 2, 3},
                           {4, 5, 6}};

    cout << "Transpose is: " << endl;
    for (int i = 0; i < cols; i++)
    {
        for (int j = 0; j < rows; j++)
        {
            cout << mat[j][i] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Q.21 WAP to add 2D matrices.

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

int main()
{
    int row, col;
    cout << "Enter number of rows and columns of two matrices: ";
    cin >> row >> col;

    int m1[row][col], m2[row][col], sum[row][col] = {0};

    cout << "Enter " << (row * col) << " elements for matrix-1(m1):
";
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            cin >> m1[i][j];
        }
    }

    cout << "Enter " << (row * col) << " elements for matrix-2(m2):
";
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            cin >> m2[i][j];
        }
    }

    cout << "Sum of both the matrices is: " << endl;
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++)
        {
            sum[i][j] = m1[i][j] + m2[i][j];
            cout << sum[i][j] << " ";
        }
        cout << endl;
   }

    return 0;
}
```

Q.22 WAP to multiply 2D matrices.



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

int main()
{
    int r1 = 2, c1 = 3;
    int &r2 = c1, c2 = 3;

    int m1[r1][c1] = {{1, 2, 3},  // 00 01 02
                      {4, 5, 6}}; // 10 11 12

    int m2[r2][c2] = {{1, 2, 3},  // 00 01 02
                      {1, 2, 3},  // 10 11 12
                      {1, 2, 3}}; // 20 21 22

    int m3[r1][c2];

    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c1; j++)
        {
            int mul_element = 0;
            for (int k = 0; k < c2; k++)
            {
                mul_element += (m1[i][k] * m2[k][j]);
            }
            m3[i][j] = mul_element;
        }
    }

    for (int i = 0; i < r1; i++)
    {
        for (int j = 0; j < c2; j++)
        {
            cout << m3[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Q.23 WAP to sort an array in ascending order.

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

int main()
{
    int arr[10] = {-1, -22, 3, 4, -90, 89, 10, 20, 30,
40};

    for (int i = 0; i < 10; i++)
    {
        for (int j = (i + 1); j < 10; j++)
        {
            if (arr[j] < arr[i])
            {
                int temp;
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }

    cout << "Sorted Array: ";
    for (int i = 0; i < 10; i++)
    {
        cout << arr[i] << " ";
    }

    return 0;
}
```

Q.24 WAP to reverse a given string.

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

int main()
{
    string str;
    cout << "Enter the string: ";
    cin >> str;

    string str_rev = str;

    int l = 0;
    while (str_rev[l] != '\0')
    {
        l++;
    }

    for (int i = 0; i < (l / 2); i++)
    {
        char temp;
        temp = str_rev[i];
        str_rev[i] = str_rev[l - i - 1];
        str_rev[(l - i - 1)] = temp;
    }

    cout << "Reverse of " << str << " is: " <<
str_rev<<endl;

    return 0;
}
```

Q.25 WAP to count all vowels in a given string.

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

int main()
{
    string str;
    cout << "Enter a string: ";
    getline(cin, str);

    int i = 0, no_of_vowels = 0;
    while (str[i] != '\0')
    {
        if ((str[i] == 'A') || (str[i] == 'E') || (str[i] == 'I') || (str[i] == 'O') || (str[i] == 'U') || (str[i] == 'a') || (str[i] == 'e') || (str[i] == 'i') || (str[i] == 'o') || (str[i] == 'u'))
        {
            no_of_vowels++;
        }
        i++;
    }

    cout << "Number of vowels in " << str << " is: " << no_of_vowels << endl;

    return 0;
}
```

Q.26 WAP to check if a given string is palindrome or not.

```
#include <iostream>

#include <cstring>
using namespace std;

int main()
{
    string str;
    cout << "Enter the string: ";
    cin >> str;

    int l = str.length();

    string str_rev;
    str_rev.resize(l);

    for (int i = 0; i < l; i++)
    {
        str_rev[i] = str[(l - i - 1)];
    }

    cout << "Reverse of " << str << " is: " << str_rev <<
endl;

    if (str_rev == str)
    {
        cout << "Palindrome\n";
    }
    else
    {
        cout << "Not Palindrome\n";
    }

    return 0;
}
```

Q.27 WAP to check if a given string is anagram or not.

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

bool anagram(string str1, string str2)
{

    sort(str1.begin(), str1.end());
    sort(str2.begin(), str2.end());

    return (str1 == str2);
}

int main()
{
    string str1;
    string str2;

    cout << "Enter two strings: ";
    getline(cin, str1);
    getline(cin, str2);

    if (anagram(str1, str2))
    {
        cout << "Anagram\n";
    }
    else
    {
        cout << "Not Anagram\n";
    }

    return 0;
}
```