```cpp
/*-------Add 1 to binary-------*/

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

int binAdd(string &n)   /* If we don't use &
the string will be considered as a copy and
the actual string will not get effected by
this function */
{
    int len = n.size();
    int carry = 1;
    int i = len - 1;
    int temp;

    while(i >= 0)
    {
        temp = n[i] - '0';  /* We can't
        directly sum n[i] with carry so
        we're first converting n[i] to
        int type data */

        temp += carry; /* Now we're
        adding the int type value of
        n[i] to carry */

        if(temp == 2)
        {
            n[i] = '0';
        }
        else if(temp == 1)
        {
            n[i] = '1';
            carry = 0;
        }
        else
        {
            n[i] = '0';
            carry = 0;
        }
        i--;
    }

    if(carry == 1)
        n = '1' + n;
```

```
47: }
48:
49: int main()
50: {
51:     string n;
52:     cout << "Enter your binary number:\n";
53:     cin >> n;
54:
55:     binAdd(n);
56:     cout << "Adding 1 we get " << n;
57:
58:     return 0;
59: }
```

```cpp
/*--------Binary to Decimal--------*/

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

binToDeci(int n)
{
    int digit;
    int decimal = 0;
    int i = 0;

    while(n > 0)
    {
        digit = n % 10;
        decimal += (pow(2, i) * digit);
        n /= 10;
        i++;
    }

    cout << decimal;
}

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

    cout << "Decimal representation of " << n << " is ";
    binToDeci(n);

    return 0;
}
```

```cpp
/*--------Decimal to Binary--------*/

#include<iostream>
using namespace std;

void binary(int n)
{
    /* First we'll find the required dimension
    of our array to store the remainders */
    int size = 0;
    int temp = n;
    while(temp > 0)
    {
        temp /= 2;
        size += 1;
    }

    int arr[size];
    int index = size - 1;
    while(n > 0)
    {
        arr[index] = n % 2;
        n /= 2;
        index--;
    }

    for(int j = 0; j < size; j++)
    {
        cout << arr[j];
    }
}

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

    cout << "Binary representation of " << n << " is:\n";
    binary(n);

    return 0;
}
```

```cpp
/*--------Decimal to Octal--------*/

#include<iostream>
using namespace std;

void hexaDecimal(int n)
{
    /* First we'll find the required dimension
    of our array to store the remainders */
    int size = 0;
    int temp = n;
    while(temp > 0)
    {
        temp /= 16;
        size += 1;
    }

    char arr[size]; // Type is char to take char val
    int index = 0;
    while(n > 0)
    {
        int rem = 0;
        rem = n % 16;

        if(rem < 10)
        {
            arr[index] = rem + 48;  /* Since ASCII
            val of 48 is 0 so val asigned is temp*/
        }
        else
        {
            arr[index] = rem + 55;  /* Since ASCII val
            of 65 is A, 66 is B & so on, so when remainder
            is 10 then arr will be A, when 11 array will
            be B & vice versa */
        }

        index++;
        n /= 16;
    }

    for(int j = index - 1; j >= 0; j--)
    {
        cout << arr[j];
    }
}
```

```
47:
48: int main()
49: {
50:     int n;
51:     cout << "Enter your number: \n";
52:     cin >> n;
53:
54:     cout << "Hexa Decimal representation of " << n << " is:\n";
55:     hexaDecimal(n);
56:
57:     return 0;
58: }
```

```cpp
/*--------Decimal to Octal--------*/

#include<iostream>
using namespace std;

void octal(int n)
{
    /* First we'll find the required dimension
    of our array to store the remainders */
    int size = 0;
    int temp = n;
    while(temp > 0)
    {
        temp /= 8;
        size += 1;
    }

    int arr[size];
    int index = 0;
    while(n > 0)
    {
        arr[index] = n % 8;
        n /= 8;
        index++;
    }

    for(int j = index - 1; j >= 0; j--)
    {
        cout << arr[j];
    }
}

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

    cout << "Octal representation of " << n << " is:\n";
    octal(n);

    return 0;
}
```

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

void swap(int arr[], int n, int i)
{
    int temp = arr[n];
    arr[n] = arr[i];
    arr[i] = temp;
}

void heapify(int arr[], int n, int i)
{
    int l_c = 2 * i + 1;
    int r_c = 2 * i + 2;
    int large = i;

    if(l_c < n && arr[l_c] > arr[large])
    {
        large = l_c;
    }
    if(r_c < n && arr[r_c] > arr[large])
    {
        large = r_c;
    }

    if(large != i)
    {
        swap(arr, i, large);
        heapify(arr, n, large);
    }
}

void build_heap(int arr[], int n)
{
    for(int i = (n - 1) / 2; i >= 0; i--)
    {
        heapify(arr, n, i);
    }
}

int height(int arr[], int n)
{
    int i = 0;
    int height = 0;

```

```
47:     while(i < n)
48:     {
49:         height++;
50:         i = 2 * i + 1;
51:     }
52:     return height;
53: }
54:
55: void heapTree(int arr[], int n)
56: {
57:     int h = height(arr, n);
58:     int blocks = pow(2, h);
59:     int temp_blocks = blocks;
60:     int index = 0;
61:     int printed_index;
62:     int k = 3;
63:
64:     for(int i = 0; i < h; i++)
65:     {
66:         for(int j = 0; j < blocks; j++)
67:         {
68:             if(j == temp_blocks / 2 && index < n)
69:             {
70:                 cout << arr[index];
71:                 index++;
72:                 printed_index = j;
73:             }
74:             else
75:             {
76:                 if(j == (temp_blocks * k) / 2 && index < n)
77:                 {
78:                     cout << arr[index];
79:                     index++;
80:                     k += 2;
81:                 }
82:                 else
83:                 {
84:                     cout << " ";
85:                 }
86:             }
87:         }
88:         cout << "\n";
89:         printed_index = 0;
90:         temp_blocks = temp_blocks / 2;
91:         k = 3;
92:     }
```

```
 93: }
 94:
 95: void deleteElt(int arr[], int& n, int i)
 96: {
 97:     int last_elt = arr[n - 1];
 98:
 99:     arr[i] = last_elt;
100:
101:     n -= 1;
102:
103:     build_heap(arr, n);
104: }
105:
106: void insertElt(int arr[], int& n, int i)
107: {
108:     n += 1;
109:
110:     arr[n - 1] = i;
111:
112:     build_heap(arr, n);
113: }
114:
115: int main()
116: {
117:     int n;
118:     cout << "Enter the length:\n";
119:     cin >> n;
120:
121:     int arr[n];
122:     cout << "Enter elements: \n";
123:     for(int i = 0; i < n; i++)
124:     {
125:         cin >> arr[i];
126:     }
127:
128:     build_heap(arr, n);
129:     heapTree(arr, n);
130:
131:     //Deletion
132:     int i;
133:
134:     cout << "By looking at the tree enter ";
135:     cout << "the index of your element you";
136:     cout << " want to delete: \n";
137:
138:     cin >> i;
```

```
139:     cout << "After deleting '" << arr[i];
140:     cout << "' New heap tree is: \n";
141:     deleteElt(arr, n, i);
142:     heapTree(arr, n);
143:
144:
145:     //Insertion
146:     int j;
147:
148:     cout << "Enter the element you want to insert:\n";
149:     cin >> j;
150:
151:     insertElt(arr, n, j);
152:     cout << "New heap tree is: \n";
153:     heapTree(arr, n);
154:
155:
156:     return 0;
157: }
```

```cpp
/*---------GDC using Euclid's alg---------*/

#include<iostream>
using namespace std;

int euclidGCD(int a, int b)
{
    int gcd;

    if(a == 0)
    {
        gcd = b;
    }
    else if(b == 0)
    {
        gcd = a;
    }
    else
    {
        if(a > b)
        {
            gcd = euclidGCD(b, a % b);
        }
        else
        {
            gcd = euclidGCD(a, b % a);
        }
    }

    return gcd;
}

int main()
{
    int n;
    int a, b;
    cout << "Enter two numbers: \n";
    cin >> a >> b;

    int gcd = euclidGCD(a, b);

    cout << "GCD of " << a << " & " << b << " is " << gcd;
    return 0;
}
```

```cpp
/*---Factorial of n----*/

#include<iostream>
using namespace std;

long long fact(int n)
{
    long long factorial = n;
    for(int i = 1; i < n; i++)
        factorial *= i;
    return factorial;
}

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

    cout << "Factorial of " << n;
    cout << " is" << fact(n);

    return 0;
}
```

```cpp
/*-------GCD & LCM of two given nums--------*/

#include<iostream>
using namespace std;

int GCD(int a, int b)
{
    int gcd = 0;
    int x = min(a, b);
    int y = max(a, b);

    if(x == y)
    {
        gcd = x;
    }
    else
    {
        for(int i = 1; i <= x; i++)
        {
            if(x % i == 0 && y % i == 0)
            {
                gcd = i;
            }
        }
    }
    return gcd;
}

/*
int LCM(int a, int b)
{
    return (a * b)/GCD(a, b);
}
*/

int LCM(int a, int b)
{
    int lcm = 0;
    int x = min(a, b);
    int y = max(a, b);

    if(x == y)
        lcm = x;
    else
    {
        int i = y;
```

```
47:         while(i <= x * y)
48:         {
49:             if(i % x == 0 && i % y == 0)
50:             {
51:                 lcm = i;
52:                 break;
53:             }
54:             i++;
55:         }
56:     }
57:     return lcm;
58: }
59:
60: int main()
61: {
62:     int a, b;
63:     cout << "Enter your nums: \n";
64:     cin >> a >> b;
65:
66:     int gcd = GCD(a, b);
67:     int lcm = LCM(a, b);
68:
69:     cout << "GCD & LCM of " << a << " & " << b << " are " << gcd << " & " << l
70:
71:     return 0;
72: }
```

```cpp
/*----------Heap Tree Print--------*/

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

void swap(int arr[], int n, int i)
{
    int temp = arr[n];
    arr[n] = arr[i];
    arr[i] = temp;
}

void heapify(int arr[], int n, int i)
{
    int l_c = 2 * i + 1;
    int r_c = 2 * i + 2;
    int large = i;

    if(l_c < n && arr[l_c] > arr[large])
    {
        large = l_c;
    }
    if(r_c < n && arr[r_c] > arr[large])
    {
        large = r_c;
    }

    if(large != i)
    {
        swap(arr, i, large);
        heapify(arr, n, large);
    }
}

void build_heap(int arr[], int n)
{
    for(int i = (n - 1) / 2; i >= 0; i--)
    {
        heapify(arr, n, i);
    }
}

int height(int arr[], int n)
{
    int i = 0;
```

```
47:     int height = 0;
48:
49:     while(i < n)
50:     {
51:         height++;
52:         i = 2 * i + 1;
53:     }
54:     return height;
55: }
56:
57: void heapTree(int arr[], int n)
58: {
59:     int h = height(arr, n);
60:     int blocks = pow(2, h);
61:     int temp_blocks = blocks;
62:     int index = 0;
63:     int printed_index;
64:     int k = 3;
65:
66:     for(int i = 0; i < h; i++)
67:     {
68:         for(int j = 0; j < blocks; j++)
69:         {
70:             if(j == temp_blocks / 2 && index < n)
71:             {
72:                 cout << arr[index];
73:                 index++;
74:                 printed_index = j;
75:             }
76:             else
77:             {
78:                 if(j == (temp_blocks * k) / 2 && index < n)
79:                 {
80:                     cout << arr[index];
81:                     index++;
82:                     k += 2;
83:                 }
84:                 else
85:                 {
86:                     cout << " ";
87:                 }
88:             }
89:         }
90:         cout << "\n";
91:         printed_index = 0;
92:         temp_blocks = temp_blocks / 2;
```

```
 93:         k = 3;
 94:     }
 95: }
 96:
 97: int main()
 98: {
 99:     int n;
100:     cout << "Enter the length:\n";
101:     cin >> n;
102:
103:     int arr[n];
104:     cout << "Enter elements: \n";
105:     for(int i = 0; i < n; i++)
106:     {
107:         cin >> arr[i];
108:     }
109:
110:     build_heap(arr, n);
111:     heapTree(arr, n);
112:
113:     return 0;
114: }
```

```cpp
/*--------Highly Composite-------*/

/*A num is highly composite if the num of divisors
of num is greater than the num of divisors of all
--------nums less than the given num-----------*/

#include<iostream>
using namespace std;

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

int highlyCom(int n)
{
    int result = 1;
    for(int i = 1; i < n; i++)
    {
        if(numDivisors(i) >= numDivisors(n))
            result = 0;
    }
    return result;
}

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

    if(highlyCom(n) == 1)
        cout << n << " is a highly composite number. \n";
    else
        cout << n << " is NOT a highly composite number. \n";

    return 0;
}
```

```cpp
/*---Matrix Addition----*/

#include<iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the dimension of your matrix:\n";
    cin >> n;

    int add[n][n], M1[n][n], M2[n][n];
    cout << "Enter first matrix:\n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cin >> M1[i][j];
        }
    }

    cout << "Enter second matrix:\n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cin >> M2[i][j];
        }
    }

    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            add[i][j] = M1[i][j] + M2[i][j];
        }
    }

    cout << "Addition is: \n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cout << add[i][j] << " ";
        }
        cout << endl;
```

```
47:
48:     }
49:
50:     return 0;
51: }
```

```cpp
/*---Matrix Multiplication----*/

#include<iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the dimension of your matrix:\n";
    cin >> n;

    int Multi[n][n], M1[n][n], M2[n][n];
    cout << "Enter first matrix:\n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cin >> M1[i][j];
        }
    }

    cout << "Enter second matrix:\n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            cin >> M2[i][j];
        }
    }

    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
        {
            Multi[i][j] = 0;
            for(int k = 0; k < n; k++)
            {
                Multi[i][j] += M1[i][k] * M2[k][j];
            }
        }
    }

    cout << "Multiplication is: \n";
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
```

```
47:         {
48:             cout << Multi[i][j] << " ";
49:         }
50:         cout << endl;
51:
52:     }
53:
54:     return 0;
55: }
```

```cpp
/*---nth term of Fibbo----*/

#include<iostream>
using namespace std;

int Fibo(int n)
{
    int arr[n];
    arr[0] = 1;
    arr[1] = 1;

    for(int i = 2; i < n; i++)
        arr[i] = arr[i - 1] + arr[i - 2];

    return arr[n - 1];
}

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

    cout << n << "th term the Fibonacci ";
    cout << "sequence is " << Fibo(n);

    return 0;
}
```

```cpp
/*---num of prime till n----*/

#include<iostream>
using namespace std;

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

void numPrimes(int n)
{
    for(int i = 2; i <= n; i++)
    {
        if(numFact(i) <= 2)
            cout << i << " ";
    }
}

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

    cout << "Prime numbers till " << n;
    cout << " are:\n";
    numPrimes(n);
    return 0;
}
```

```cpp
/*---num of divisors of n----*/

#include<iostream>
using namespace std;

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

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

    int nums = numFact(n);
    cout << "Number of divisors of ";
    cout << n << " is " << nums;

    return 0;
}
```

```cpp
/*--------Octal to Decimal--------*/

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

octToDeci(int n)
{
    int digit;
    int decimal = 0;
    int i = 0;

    while(n > 0)
    {
        digit = n % 10;
        decimal += (pow(8, i) * digit);
        n /= 10;
        i++;
    }

    cout << decimal;
}

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

    cout << "Decimal representation of " << n << " is ";
    octToDeci(n);

    return 0;
}
```

```cpp
/*-----Check n is perfect or not-----*/

/* Perfect number: Whose sum of factors, excluding
the num itself is equal to the num.-------------*/

#include<iostream>
using namespace std;

void perfectNum(int n)
{
    int i = 1;
    int sum = 0;
    while(i < n)
    {
        if(n % i == 0)
        {
            sum += i;
        }
        i++;
    }

    if(sum == n)
        cout << n << " is a perfect number. \n";
    else
        cout << n << " is not a prefect number. \n";
}

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

    perfectNum(n);

    return 0;
}
```

```cpp
/*---n is prime or not----*/

#include<iostream>
using namespace std;

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

void checkPrime(int n)
{

    if(numFact(n) == 2)
        cout << n << " is a Prime";
    else
        cout << n << " is NOT a Prime";

}

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

    checkPrime(n);
    return 0;
}
```

```cpp
/*-------Pythagorean Triplets less than n-------*/

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

void checkPtriple(int n)
{
    int i = 1;
    while(i <= n)
    {
        int power = pow(i, 2);
        for(int j = 1; j <= power; j++)
        {
            for(int k = j; k <= power; k++)
            {
                if(power == pow(j, 2) + pow(k , 2))
                {
                    cout << "(" << i << "," << j << "," << k << ")" << endl;
                }
            }
        }
        i++;
    }
}

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

    cout << "Possible Pythagorean triplets less than " << n << " are: \n";
    checkPtriple(n);

    return 0;
}
```

```cpp
/*---nums relatively prime to n----*/

#include<iostream>
using namespace std;

int GCD(int a, int b)
{
    int gcd = 0;
    int x = min(a, b);
    int y = max(a, b);

    if(x == y)
    {
        gcd = x;
    }
    else
    {
        for(int i = 1; i <= x; i++)
        {
            if(x % i == 0 && y % i == 0)
            {
                gcd = i;
            }
        }
    }
    return gcd;
}

void relPrimes(int n)
{
    for(int i = 0; i < n; i++)
    {
        if(GCD(i, n) == 1)
            cout << i << " ";
    }
}

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

    cout << "Numbers less than " << n << " & relatively prime";
    cout << " to" << n << " are:\n";
    relPrimes(n);
```

```
47:     return 0;
48: }
```

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

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

    for(int i = 0; i < n; i++)
    {
        for(int j = i; j < n; j++)
        {
            for(int k = j; k < n; k++)
            {
                for(int l = k; l < n; l++)
                {
                    if(i * i + j * j + k * k + l * l == n)
                    {
                        cout << n << " = " << i << "^2 + " ;
                        cout << j << "^2 + " << k << "^2 + ";
                        cout << l << "^2";
                        goto label; /* We might get multiple
                        expression of a num, so as soon as we
                        are getting one we're exiting out of
                        all loops */
                    }
                }
            }
        }
    }
    label:

    return 0;
}
```

```cpp
/*---Sum of first n nums----*/

#include<iostream>
using namespace std;

int sumTill(int n)
{
    int sum = 0;
    for(int i = 1; i <= n; i++)
        sum += i;
    return sum;
}

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

    cout << "Sum of first " << n;
    cout << " number is " << sumTill(n);

    return 0;
}
```