

```

1: /*-----Add 1 to binary-----*/
2:
3: #include<iostream>
4: #include<string>
5: using namespace std;
6:
7: int binAdd(string &n) /* If we don't use &
8: the string will be considered as a copy and
9: the actual string will not get effected by
10: this function */
11: {
12:     int len = n.size();
13:     int carry = 1;
14:     int i = len - 1;
15:     int temp;
16:
17:     while(i >= 0)
18:     {
19:         temp = n[i] - '0'; /* We can't
20:         directly sum n[i] with carry so
21:         we're first converting n[i] to
22:         int type data */
23:
24:         temp += carry; /* Now we're
25:         adding the int type value of
26:         n[i] to carry */
27:
28:         if(temp == 2)
29:         {
30:             n[i] = '0';
31:         }
32:         else if(temp == 1)
33:         {
34:             n[i] = '1';
35:             carry = 0;
36:         }
37:         else
38:         {
39:             n[i] = '0';
40:             carry = 0;
41:         }
42:         i--;
43:     }
44:
45:     if(carry == 1)
46:         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: }
```

```

1: /*-----Binary to Decimal-----*/
2:
3: #include<iostream>
4: #include<cmath>
5: using namespace std;
6:
7: binToDeci(int n)
8: {
9:     int digit;
10:    int decimal = 0;
11:    int i = 0;
12:
13:    while(n > 0)
14:    {
15:        digit = n % 10;
16:        decimal += (pow(2, i) * digit);
17:        n /= 10;
18:        i++;
19:    }
20:
21:    cout << decimal;
22: }
23:
24: int main()
25: {
26:     int n;
27:     cout << "Enter your number: \n";
28:     cin >> n;
29:
30:     cout << "Decimal representation of " << n << " is ";
31:     binToDeci(n);
32:
33:     return 0;
34: }

```

```

1:  /*-----Decimal to Binary-----*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  void binary(int n)
7:  {
8:      /* First we'll find the required dimension
9:      of our array to store the remainders */
10:     int size = 0;
11:     int temp = n;
12:     while(temp > 0)
13:     {
14:         temp /= 2;
15:         size += 1;
16:     }
17:
18:     int arr[size];
19:     int index = size - 1;
20:     while(n > 0)
21:     {
22:         arr[index] = n % 2;
23:         n /= 2;
24:         index--;
25:     }
26:
27:     for(int j = 0; j < size; j++)
28:     {
29:         cout << arr[j];
30:     }
31: }
32:
33: int main()
34: {
35:     int n;
36:     cout << "Enter your number: \n";
37:     cin >> n;
38:
39:     cout << "Binary representation of " << n << " is:\n";
40:     binary(n);
41:
42:     return 0;
43: }

```

```

1:  /*-----Decimal to Octal-----*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  void hexaDecimal(int n)
7:  {
8:      /* First we'll find the required dimension
9:      of our array to store the remainders */
10:     int size = 0;
11:     int temp = n;
12:     while(temp > 0)
13:     {
14:         temp /= 16;
15:         size += 1;
16:     }
17:
18:     char arr[size]; // Type is char to take char val
19:     int index = 0;
20:     while(n > 0)
21:     {
22:         int rem = 0;
23:         rem = n % 16;
24:
25:         if(rem < 10)
26:         {
27:             arr[index] = rem + 48; /* Since ASCII
28:             val of 48 is 0 so val assigned is temp*/
29:         }
30:         else
31:         {
32:             arr[index] = rem + 55; /* Since ASCII val
33:             of 65 is A, 66 is B & so on, so when remainder
34:             is 10 then arr will be A, when 11 array will
35:             be B & vice versa */
36:         }
37:
38:         index++;
39:         n /= 16;
40:     }
41:
42:     for(int j = index - 1; j >= 0; j--)
43:     {
44:         cout << arr[j];
45:     }
46: }

```

```
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: }
```

```

1:  /*-----Decimal to Octal-----*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  void octal(int n)
7:  {
8:      /* First we'll find the required dimension
9:      of our array to store the remainders */
10:     int size = 0;
11:     int temp = n;
12:     while(temp > 0)
13:     {
14:         temp /= 8;
15:         size += 1;
16:     }
17:
18:     int arr[size];
19:     int index = 0;
20:     while(n > 0)
21:     {
22:         arr[index] = n % 8;
23:         n /= 8;
24:         index++;
25:     }
26:
27:     for(int j = index - 1; j >= 0; j--)
28:     {
29:         cout << arr[j];
30:     }
31: }
32:
33: int main()
34: {
35:     int n;
36:     cout << "Enter your number: \n";
37:     cin >> n;
38:
39:     cout << "Octal representation of " << n << " is:\n";
40:     octal(n);
41:
42:     return 0;
43: }

```

```

1: #include<iostream>
2: #include<cmath>
3: using namespace std;
4:
5: void swap(int arr[], int n, int i)
6: {
7:     int temp = arr[n];
8:     arr[n] = arr[i];
9:     arr[i] = temp;
10: }
11:
12: void heapify(int arr[], int n, int i)
13: {
14:     int l_c = 2 * i + 1;
15:     int r_c = 2 * i + 2;
16:     int large = i;
17:
18:     if(l_c < n && arr[l_c] > arr[large])
19:     {
20:         large = l_c;
21:     }
22:     if(r_c < n && arr[r_c] > arr[large])
23:     {
24:         large = r_c;
25:     }
26:
27:     if(large != i)
28:     {
29:         swap(arr, i, large);
30:         heapify(arr, n, large);
31:     }
32: }
33:
34: void build_heap(int arr[], int n)
35: {
36:     for(int i = (n - 1) / 2; i >= 0; i--)
37:     {
38:         heapify(arr, n, i);
39:     }
40: }
41:
42: int height(int arr[], int n)
43: {
44:     int i = 0;
45:     int height = 0;
46:

```



```

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: }
```

```

1:  /*-----GDC using Euclid's alg-----*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  int euclidGCD(int a, int b)
7:  {
8:      int gcd;
9:
10:     if(a == 0)
11:     {
12:         gcd = b;
13:     }
14:     else if(b == 0)
15:     {
16:         gcd = a;
17:     }
18:     else
19:     {
20:         if(a > b)
21:         {
22:             gcd = euclidGCD(b, a % b);
23:         }
24:         else
25:         {
26:             gcd = euclidGCD(a, b % a);
27:         }
28:     }
29:
30:     return gcd;
31: }
32:
33: int main()
34: {
35:     int n;
36:     int a, b;
37:     cout << "Enter two numbers: \n";
38:     cin >> a >> b;
39:
40:     int gcd = euclidGCD(a, b);
41:
42:     cout << "GCD of " << a << " & " << b << " is " << gcd;
43:     return 0;
44: }

```

```
1: /*---Factorial of n----*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: long long fact(int n)
7: {
8:     long long factorial = n;
9:     for(int i = 1; i < n; i++)
10:         factorial *= i;
11:     return factorial;
12: }
13:
14: int main()
15: {
16:     int n;
17:     cout << "Enter your number:\n";
18:     cin >> n;
19:
20:     cout << "Factorial of " << n;
21:     cout << " is" << fact(n);
22:
23:     return 0;
24: }
```

```

1:  /*-----GCD & LCM of two given nums-----*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  int GCD(int a, int b)
7:  {
8:      int gcd = 0;
9:      int x = min(a, b);
10:     int y = max(a, b);
11:
12:     if(x == y)
13:     {
14:         gcd = x;
15:     }
16:     else
17:     {
18:         for(int i = 1; i <= x; i++)
19:         {
20:             if(x % i == 0 && y % i == 0)
21:             {
22:                 gcd = i;
23:             }
24:         }
25:     }
26:     return gcd;
27: }
28:
29: /*
30: int LCM(int a, int b)
31: {
32:     return (a * b)/GCD(a, b);
33: }
34: */
35:
36: int LCM(int a, int b)
37: {
38:     int lcm = 0;
39:     int x = min(a, b);
40:     int y = max(a, b);
41:
42:     if(x == y)
43:         lcm = x;
44:     else
45:     {
46:         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 << " & " << 1
70:
71:     return 0;
72: }

```

```

1: /*-----Heap Tree Print-----*/
2:
3: #include<iostream>
4: #include<cmath>
5: using namespace std;
6:
7: void swap(int arr[], int n, int i)
8: {
9:     int temp = arr[n];
10:    arr[n] = arr[i];
11:    arr[i] = temp;
12: }
13:
14: void heapify(int arr[], int n, int i)
15: {
16:     int l_c = 2 * i + 1;
17:     int r_c = 2 * i + 2;
18:     int large = i;
19:
20:     if(l_c < n && arr[l_c] > arr[large])
21:     {
22:         large = l_c;
23:     }
24:     if(r_c < n && arr[r_c] > arr[large])
25:     {
26:         large = r_c;
27:     }
28:
29:     if(large != i)
30:     {
31:         swap(arr, i, large);
32:         heapify(arr, n, large);
33:     }
34: }
35:
36: void build_heap(int arr[], int n)
37: {
38:     for(int i = (n - 1) / 2; i >= 0; i--)
39:     {
40:         heapify(arr, n, i);
41:     }
42: }
43:
44: int height(int arr[], int n)
45: {
46:     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: }
```

```

1:  /*-----Highly Composite-----*/
2:
3:  /*A num is highly composite if the num of divisors
4:  of num is greater than the num of divisors of all
5:  -----nums less than the given num-----*/
6:
7:  #include<iostream>
8:  using namespace std;
9:
10: int numDivisors(int n)
11: {
12:     int count = 0;
13:     for(int i = 1; i <= n; i++)
14:     {
15:         if(n % i == 0)
16:         {
17:             count += 1;
18:         }
19:     }
20:     return count;
21: }
22:
23: int highlyCom(int n)
24: {
25:     int result = 1;
26:     for(int i = 1; i < n; i++)
27:     {
28:         if(numDivisors(i) >= numDivisors(n))
29:             result = 0;
30:     }
31:     return result;
32: }
33:
34: int main()
35: {
36:     int n;
37:     cout << "Enter your number: \n";
38:     cin >> n;
39:
40:     if(highlyCom(n) == 1)
41:         cout << n << " is a highly composite number. \n";
42:     else
43:         cout << n << " is NOT a highly composite number. \n";
44:
45:     return 0;
46: }

```

```

1:  /*---Matrix Addition---*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  int main()
7:  {
8:      int n;
9:      cout << "Enter the dimension of your matrix:\n";
10:     cin >> n;
11:
12:     int add[n][n], M1[n][n], M2[n][n];
13:     cout << "Enter first matrix:\n";
14:     for(int i = 0; i < n; i++)
15:     {
16:         for(int j = 0; j < n; j++)
17:         {
18:             cin >> M1[i][j];
19:         }
20:     }
21:
22:     cout << "Enter second matrix:\n";
23:     for(int i = 0; i < n; i++)
24:     {
25:         for(int j = 0; j < n; j++)
26:         {
27:             cin >> M2[i][j];
28:         }
29:     }
30:
31:     for(int i = 0; i < n; i++)
32:     {
33:         for(int j = 0; j < n; j++)
34:         {
35:             add[i][j] = M1[i][j] + M2[i][j];
36:         }
37:     }
38:
39:     cout << "Addition is: \n";
40:     for(int i = 0; i < n; i++)
41:     {
42:         for(int j = 0; j < n; j++)
43:         {
44:             cout << add[i][j] << " ";
45:         }
46:         cout << endl;

```

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

```

1: /*---Matrix Multiplication---*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int main()
7: {
8:     int n;
9:     cout << "Enter the dimension of your matrix:\n";
10:    cin >> n;
11:
12:    int Multi[n][n], M1[n][n], M2[n][n];
13:    cout << "Enter first matrix:\n";
14:    for(int i = 0; i < n; i++)
15:    {
16:        for(int j = 0; j < n; j++)
17:        {
18:            cin >> M1[i][j];
19:        }
20:    }
21:
22:    cout << "Enter second matrix:\n";
23:    for(int i = 0; i < n; i++)
24:    {
25:        for(int j = 0; j < n; j++)
26:        {
27:            cin >> M2[i][j];
28:        }
29:    }
30:
31:    for(int i = 0; i < n; i++)
32:    {
33:        for(int j = 0; j < n; j++)
34:        {
35:            Multi[i][j] = 0;
36:            for(int k = 0; k < n; k++)
37:            {
38:                Multi[i][j] += M1[i][k] * M2[k][j];
39:            }
40:        }
41:    }
42:
43:    cout << "Multiplication is: \n";
44:    for(int i = 0; i < n; i++)
45:    {
46:        for(int j = 0; j < n; j++)

```

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

```
1: /*---nth term of Fibbo----*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int Fibo(int n)
7: {
8:     int arr[n];
9:     arr[0] = 1;
10:    arr[1] = 1;
11:
12:    for(int i = 2; i < n; i++)
13:        arr[i] = arr[i - 1] + arr[i - 2];
14:
15:    return arr[n - 1];
16: }
17:
18: int main()
19: {
20:     int n;
21:     cout << "Enter your number:\n";
22:     cin >> n;
23:
24:     cout << n << "th term the Fibonacci ";
25:     cout << "sequence is " << Fibo(n);
26:
27:     return 0;
28: }
```



```

1: /*---num of prime till n---*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int numFact(int n)
7: {
8:     int count = 0;
9:     for(int i = 1; i <= n; i++)
10:    {
11:        if(n % i == 0)
12:            count += 1;
13:    }
14:    return count;
15: }
16:
17: void numPrimes(int n)
18: {
19:     for(int i = 2; i <= n; i++)
20:     {
21:         if(numFact(i) <= 2)
22:             cout << i << " ";
23:     }
24: }
25:
26: int main()
27: {
28:     int n;
29:     cout << "Enter your number:\n";
30:     cin >> n;
31:
32:     cout << "Prime numbers till " << n;
33:     cout << " are:\n";
34:     numPrimes(n);
35:     return 0;
36: }

```

```
1: /*---num of divisors of n---*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int numFact(int n)
7: {
8:     int count = 0;
9:     for(int i = 1; i <= n; i++)
10:    {
11:        if(n % i == 0)
12:            count += 1;
13:    }
14:    return count;
15: }
16:
17: int main()
18: {
19:     int n;
20:     cout << "Enter your number:\n";
21:     cin >> n;
22:
23:     int nums = numFact(n);
24:     cout << "Number of divisors of ";
25:     cout << n << " is " << nums;
26:
27:     return 0;
28: }
```

```

1: /*-----Octal to Decimal-----*/
2:
3: #include<iostream>
4: #include<cmath>
5: using namespace std;
6:
7: octToDeci(int n)
8: {
9:     int digit;
10:    int decimal = 0;
11:    int i = 0;
12:
13:    while(n > 0)
14:    {
15:        digit = n % 10;
16:        decimal += (pow(8, i) * digit);
17:        n /= 10;
18:        i++;
19:    }
20:
21:    cout << decimal;
22: }
23:
24: int main()
25: {
26:     int n;
27:     cout << "Enter your number: \n";
28:     cin >> n;
29:
30:     cout << "Decimal representation of " << n << " is ";
31:     octToDeci(n);
32:
33:     return 0;
34: }

```

```

1:  /*-----Check n is perfect or not-----*/
2:
3:  /* Perfect number: Whose sum of factors, excluding
4:  the num itself is equal to the num.-----*/
5:
6:  #include<iostream>
7:  using namespace std;
8:
9:  void perfectNum(int n)
10: {
11:     int i = 1;
12:     int sum = 0;
13:     while(i < n)
14:     {
15:         if(n % i == 0)
16:         {
17:             sum += i;
18:         }
19:         i++;
20:     }
21:
22:     if(sum == n)
23:         cout << n << " is a perfect number. \n";
24:     else
25:         cout << n << " is not a prefect number. \n";
26: }
27:
28: int main()
29: {
30:     int n;
31:     cout << "Enter your number: \n";
32:     cin >> n;
33:
34:     perfectNum(n);
35:
36:     return 0;
37: }

```

```
1: /*---n is prime or not---*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int numFact(int n)
7: {
8:     int count = 0;
9:     for(int i = 1; i <= n; i++)
10:    {
11:        if(n % i == 0)
12:            count += 1;
13:    }
14:    return count;
15: }
16:
17: void checkPrime(int n)
18: {
19:
20:     if(numFact(n) == 2)
21:         cout << n << " is a Prime";
22:     else
23:         cout << n << " is NOT a Prime";
24:
25: }
26:
27: int main()
28: {
29:     int n;
30:     cout << "Enter your number:\n";
31:     cin >> n;
32:
33:     checkPrime(n);
34:     return 0;
35: }
```

```

1:  /*-----Pythagorean Triplets less than n-----*/
2:
3:  #include<iostream>
4:  #include<cmath>
5:  using namespace std;
6:
7:  void checkPtriple(int n)
8:  {
9:      int i = 1;
10:     while(i <= n)
11:     {
12:         int power = pow(i, 2);
13:         for(int j = 1; j <= power; j++)
14:         {
15:             for(int k = j; k <= power; k++)
16:             {
17:                 if(power == pow(j, 2) + pow(k, 2))
18:                 {
19:                     cout << "(" << i << "," << j << "," << k << ")" << endl;
20:                 }
21:             }
22:         }
23:         i++;
24:     }
25: }
26:
27: int main()
28: {
29:     int n;
30:     cout << "Enter your number: \n";
31:     cin >> n;
32:
33:     cout << "Possible Pythagorean triplets less than " << n << " are: \n";
34:     checkPtriple(n);
35:
36:     return 0;
37: }

```

```

1:  /*---nums relatively prime to n---*/
2:
3:  #include<iostream>
4:  using namespace std;
5:
6:  int GCD(int a, int b)
7:  {
8:      int gcd = 0;
9:      int x = min(a, b);
10:     int y = max(a, b);
11:
12:     if(x == y)
13:     {
14:         gcd = x;
15:     }
16:     else
17:     {
18:         for(int i = 1; i <= x; i++)
19:         {
20:             if(x % i == 0 && y % i == 0)
21:             {
22:                 gcd = i;
23:             }
24:         }
25:     }
26:     return gcd;
27: }
28:
29: void relPrimes(int n)
30: {
31:     for(int i = 0; i < n; i++)
32:     {
33:         if(GCD(i, n) == 1)
34:             cout << i << " ";
35:     }
36: }
37:
38: int main()
39: {
40:     int n;
41:     cout << "Enter your number:\n";
42:     cin >> n;
43:
44:     cout << "Numbers less than " << n << " & relatively prime";
45:     cout << " to" << n << " are:\n";
46:     relPrimes(n);

```

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



```

1: #include<iostream>
2: using namespace std;
3:
4: int main()
5: {
6:     int n;
7:     cout << "Enter your number: \n";
8:     cin >> n;
9:
10:    for(int i = 0; i < n; i++)
11:    {
12:        for(int j = i; j < n; j++)
13:        {
14:            for(int k = j; k < n; k++)
15:            {
16:                for(int l = k; l < n; l++)
17:                {
18:                    if(i * i + j * j + k * k + l * l == n)
19:                    {
20:                        cout << n << " = " << i << "^2 + " ;
21:                        cout << j << "^2 + " << k << "^2 + ";
22:                        cout << l << "^2";
23:                        goto label; /* We might get multiple
24:                        expression of a num, so as soon as we
25:                        are getting one we're exiting out of
26:                        all loops */
27:                    }
28:                }
29:            }
30:        }
31:    }
32:    label:
33:
34:    return 0;
35: }

```

```
1: /*---Sum of first n nums----*/
2:
3: #include<iostream>
4: using namespace std;
5:
6: int sumTill(int n)
7: {
8:     int sum = 0;
9:     for(int i = 1; i <= n; i++)
10:         sum += i;
11:     return sum;
12: }
13:
14: int main()
15: {
16:     int n;
17:     cout << "Enter your number:\n";
18:     cin >> n;
19:
20:     cout << "Sum of first " << n;
21:     cout << " number is " << sumTill(n);
22:
23:     return 0;
24: }
```