Program 1

#include <utility>

#include <vector>

std::vector< std::pair<int, int> > factor\_table;

void fill\_sieve( int n )

{

factor\_table.resize(n+1);

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

factor\_table[i] = std::pair<int, int>(i, 1);

for( int j = 2, j2 = 4; j2 <= n; (j2 += j), (j2 += ++j) ) {

if (factor\_table[j].second == 1) {

int i = j;

int ij = j2;

while (ij <= n) {

factor\_table[ij] = std::pair<int, int>(j, i);

++i;

ij += j;

}

}

}

}

std::vector<unsigned> powers;

template<int dir>

void factor( int num )

{

while (num != 1) {

powers[factor\_table[num].first] += dir;

num = factor\_table[num].second;

}

}

template<unsigned N>

void calc\_combinations(unsigned (&bin\_sizes)[N])

{

using std::swap;

powers.resize(0);

if (N < 2) return;

unsigned& largest = bin\_sizes[0];

size\_t sum = largest;

for( int bin = 1; bin < N; ++bin ) {

unsigned& this\_bin = bin\_sizes[bin];

sum += this\_bin;

if (this\_bin > largest) swap(this\_bin, largest);

}

fill\_sieve(sum);

powers.resize(sum+1);

for( unsigned i = largest + 1; i <= sum; ++i ) factor<+1>(i);

for( unsigned bin = 1; bin < N; ++bin )

for( unsigned j = 2; j <= bin\_sizes[bin]; ++j ) factor<-1>(j);

}

#include <iostream>

#include <cmath>

int main(void)

{

unsigned bin\_sizes[] = { 8, 1, 18, 19, 10, 10, 7, 18, 7, 2, 16, 8, 5, 8, 2, 3, 19, 19, 12, 1, 5, 7, 16, 0, 1, 3, 13, 15, 13, 9, 11, 6, 15, 4, 14, 4, 7, 13, 16, 2, 19, 16, 10, 9, 9, 6, 10, 10, 16, 16 };

calc\_combinations(bin\_sizes);

char\* sep = "";

for( unsigned i = 0; i < powers.size(); ++i ) {

if (powers[i]) {

std::cout << sep << i;

sep = " \* ";

if (powers[i] > 1)

std::cout << "\*\*" << powers[i];

}

}

std::cout << "\n\n";

}

Program 2

#include <iostream>

using namespace std;

int fact(int n) {

if (n == 0 || n == 1)

return 1;

else

return n \* fact(n - 1);

}

int main() {

int n, r, comb, per;

cout<<"Enter n : ";

cin>>n;

cout<<"\nEnter r : ";

cin>>r;

comb = fact(n) / (fact(r) \* fact(n-r));

cout << "\nCombination : " << comb;

per = fact(n) / fact(n-r);

cout << "\nPermutation : " << per;

return 0;

}