**// (TLE) check number of exact operations if you believe that it will accept**

**// loops I , j , k ---------> j = I , k = j**

**// Don't use float**

**// 1/6 --> 0.1(6) / 1/2 ---> 0.5(0)**

**void fileInput() {**

**freopen("in.txt", "r", stdin);**

**}**

**void fileOutput() {**

**freopen("out.txt", "w", stdout);**

**}**

**void fastIO() {**

**ios\_base::sync\_with\_stdio(0);**

**cin.tie(NULL);**

**cout.tie(NULL);**

**}**

**void fileIO() {**

**fileInput();**

**fileOutput();**

**}**

**Compare double**

**if(x == 4.7) XXXXXX fails!**

**// return 0 for a == b, 1 for a > b , -1 for a < b**

**int comp\_double(double a,double b)**

**{**

**// if very small difference , then equal**

**if(fabs(a-b) <= 1e-10)**

**return 0;**

**return a > b ? -1 : 1;**

**}**

* **round -- > away from 0 تقريب عادي**
* **trunc --> remove fraction**

**example**

**value round floor ceil trunc**

**2.3 2.0 2.0 3.0 2.0**

**5.5 6.0 5.0 6.0 5.0**

**-2.3 -2.0 -3.0 -2.0 -2.0**

**-3.8 -4.0 -4.0 -3.0 -3.0**

**round(x) == x < 0 ? ceil(x-0.5) : floor(x+0.5)**

**round to multiple of specified amount 🡪 round(x,m) = round(x/m) \* m**

**ex) round (48.2 sec,15) = 45 sec**

**Exponential**

**e --> euler's number 2.7**

**2\*\*(-x) == 0.5\*\*x / 2\*\*(x) == 0.5\*\*(-x)**

**Log**

// **Log b > 1 increment / 0 < b < 1 decrement**

**---> log10(x) / log2(x) / log(x)**

* **y = b\*\*x ---> log(b)y = x**
* **log10(1000) --> 3**

**#### Use it to know number of digits**

**digits = 1 + floor(log10(x))**

**ex) log10(1430) = 3.15 ---> 4 digits / log10(10000) = 4 ---> 5 digits**

**Factors**

**vector<int> factors(int n) {**

**vector<int> f;**

**for (int x = 2; x \* x <= n; x++) {**

**while (n % x == 0) {**

**f.push\_back(x);**

**n /= x;**

**} }**

**if (n > 1) f.push\_back(n);**

**return f;}**

**Count permutations** // examples number of ways to set down 7 students boys / girls repetition is allowed

**1)**

**vector<int> permutations;**

**int n\_perm = 4, perm\_cnt = 0;**

**bool isVisited[4];**

**void get\_permutation(int i = 0) {**

**if (i == n\_perm) {**

**++perm\_cnt; // finally will be 4! = 24**

**return; // you can print permutation here**

**}**

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

**if(isVisited[j])**

**continue;**

**permutations.push\_back(j);**

**isVisited[j] = 1;**

**get\_permutation(i+1);**

**isVisited[j] = 0;**

**permutations.pop\_back();**

**}**

**}**

**2)**

**vector<int> p = {0, 1, 2, 3};**

**int perm\_cnt = 0;**

**do {**

**++perm\_cnt;**

**} while (next\_permutation(p.begin(), p.end()));**

**cout << perm\_cnt;**

**Combinations**

**count combinations**

**vector<int> combinations;**

**int n = 20, m = 4, cnt = 0;**

**void get\_combination(int i = 0, int last\_val = 0) {**

**if (i == m) {**

**++cnt; // will be 4845 c(20,4)**

**return; // you can print combinations here**

**}**

**for (int j = last\_val + 1; j <= n; ++j) {**

**combinations.push\_back(j);**

**// think dynamically create one more loop**

**get\_combination(i + 1, j); // backtracking**

**combinations.pop\_back();**

**}**

**}**

-------------------------------------------------------------------------

**Atmost(k) = SUM Exact(i) --> i = [0-k]**

**Exact(k) = Atmost(k) - Atmost(k-1)**

**// Range (start , end)**

**--> loop / Atmost(end) - Atmost(start-1) / Atmost(start) - Atmost(end+1)**

**// 7 persons .. 9 men / 8 women**

**-> no condition c(17,7)**

**-> exactly 5 women c(8,5) \*c(9,2)**

**-> at least 5 women w >= 5 (loop) w(5)+w(6)+w(7)**

**-> at least 5 women c(17,7) useless**

**-> at least 5 women c(17,7)- c(9,2)**

**Combinations with repetitions**

**C(r+n-1,r) = C(r+n-1,n-1)**

**ex -> A+B+C+D = 100, n=4,r=100 / C(103,100) = C(103,3)**

**ex -> A+B+C+D = 100, a >= -3, b >= 5,c >= -2,d >= -1**

**--> (A-3)+(B+5)+(C-2)+(D-1) = 100 / A+B+C+D = 101**

**Partitions**

**(\*) given no.of groups / size**

**--> Order matters n!/(r1!\*r2!..rk!) / Doesn't matter (Unordered) n!/((r1!\*r2!..rk!))\*(k!)**

**Ex 🡪 18 friends into 5 groups each with size {5,5,5,2,1} friend**

**Ordered 🡪 18! / (5!\*5!\*5!\*2!\*1!) / Unordered -🡪 18! / (5!\*5!\*5!\*2!\*1!) \* 3! // repeated 3**

**(\*) given no.of groups / without size**

**--> recurences F(n,k) = k \* F(n-1,k) + F(n-1,k)**

**Bases --> (n==k,1) and (k==1,1)**

**stirling second**

**long long stirling2(long long n, long long k) {**

**if (n == k || k == 1)**

**return 1;**

**return k \* stirling2(n - 1, k) + stirling2(n - 1, k - 1);**

**}**

**Pascal's Triangle**

**int binomialCoeff(int n, int k);**

**void printPascal(int n)**

**{**

**for (int line = 0; line < n; line++)**

**{**

**for (int i = 0; i <= line; i++)**

**cout <<" "<< binomialCoeff(line, i);**

**cout <<"\n";**

**}**

**}**

**int binomialCoeff(int n, int k){**

**int res = 1;**

**if (k > n - k)**

**k = n - k;**

**for (int i = 0; i < k; ++i) {**

**res \*= (n - i);**

**res /= (i + 1);**

**}**

**return res;**

**}**

**lexicographical compare** 🡪 **string a,b; / if(a > b)**

**reverse string** 🡪 **reverse(s1.begin(),s1.end());**

**Decimal to Binary**

**int binary = 0, remainder,decimal , product = 1;**

**while (decimal != 0) {**

**remainder = decimal % 2;**

**binary = binary + (remainder \* product);**

**decimal = decimal / 2;**

**product \*= 10;**

**}**

**Prime check**

**// i <= sqrt(n) ---> float / i \* i <= n -----> int //better**

**bool isPrime(int n) {**

**if (n == 2) return true;**

**if (n < 2 || n % 2 == 0) return false;**

**for (int i = 3; i <= sqrt(n); i += 2) {**

**if (n % i == 0)**

**return false;**

**}**

**return true;**

**}**

**Count primes (remove duplicates)**

**int countPrimes(int n) {**

**vector<bool> isPrime(n + 1, true);**

**isPrime[0] = isPrime[1] = 0;**

**for (ll i = 2; i \* i <= n; ++i) {**

**if (isPrime[i]) {**

**for (int j = i \* 2; j <= n; j += i)**

**isPrime[j] = 0;**

**}**

**}**

**int count = 0;**

**for (int i = 0; i < (int) isPrime.size(); ++i)**

**if (isPrime[i])**

**count++;**

**return count**

**}**

**print first 6 float digits 🡪** **printf("%.6f\n", num);**

**array to set 🡪** **set<int> set5(arr, arr+n);**

**Shift right**

**for (int i = 0; i < n - 1; i++) {**

**arr[i] = arr[n - 1] + arr[i];**

**arr[n - 1] = arr[i] - arr[n - 1];**

**arr[i] = arr[i] - arr[n - 1];**

**}**

**OR**

**int i=n-1;**

**int j=i-1;**

**int temp=arr[n-1];**

**while(j>=0 && i>=0)**

**{**

**arr[i]=arr[j];**

**j--;**

**i--;**

**}**

**arr[i]=temp;**

**String stream**

**stringstream s(str);**

**string word;**

**int count = 0;**

**while (s >> word)**

**count++;**

**cout << count;**

**Prefix sum**

**prefix\_array1[0] = arr[0];**

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

**prefix\_array1[i] = arr[i] + prefix\_array1[i - 1];**

**}**

**Print all combination**

**void Combination(int a[], int reqLen, int start, int currLen, bool check[], int len)**

**{**

**if(currLen > reqLen)**

**return;**

**else if (currLen == reqLen) {**

**cout<<"\t";**

**for (int i = 0; i < len; i++){**

**if (check[i] == true) {**

**cout<<a[i]<<" ";**

**}**

**}**

**cout<<"\n";**

**return;**

**}**

**if (start == len) {**

**return;**

**}**

**check[start] = true;**

**Combination(a, reqLen, start + 1, currLen + 1, check, len);**

**check[start] = false;**

**Combination(a, reqLen, start + 1, currLen, check, len);**

**}**

**Combination(arr, 2, 0, 0, check, n); // 2 --> number of pairs**

**DFS**

**void dfs(int n, vector<vector<int>> &connections, int target) {**

**if (visited[n]) return;**

**visited[n] = true;**

**bara[n] = true;**

**for (auto i: connections[n]) {**

**if (visited[i]) continue;**

**if (!bara[i]) {**

**myanss = make\_pair(n, i);**

**found = true;**

**return;**

**}**

**dfs(i, connections, target);**

**if (found) return;**

**}**

**visited[n] = false;**

**return;**

**}**

**int maxCountSameSUM(int arr[], int n) {**

**unordered\_map<int, int> M;**

**for (int i = 0; i < n - 1; i++)**

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

**M[(arr[i] + arr[j])]++;**

**int max\_count = 0;**

**for (auto ele: M)**

**if (max\_count < ele.second)**

**max\_count = ele.second;**

**return max\_count;}**

**interactive**

**int l = 0,r = 1e6,mid;**

**while(l < r){**

**mid = (l + r + 1)/2;**

**cout << mid << '\n';**

**cout << flush;**

**string s;**

**cin >> s;**

**if(s == ">="){**

**l = mid;**

**}else{**

**r = mid - 1;**

**}**

**}**

**cout << "!" << " " << l << endl;**

**cout << flush;**

-------------------------------------------------------------------------

**int maxCountSameSUM(int arr[], int n) {**

**unordered\_map<int, int> M;**

**for (int i = 0; i < n - 1; i++)**

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

**M[(arr[i] + arr[j])]++;**

**int max\_count = 0;**

**for (auto ele: M)**

**if (max\_count < ele.second)**

**max\_count = ele.second;**

**return max\_count;**

**}**

**Count 3 at time and return the min sum**

**int sum = 0, min;**

**int index = k - 1;**

**for (int i = 0; i < k; i++) {**

**sum = arr[i] + sum;**

**}**

**min = sum;**

**for (int j = k; j < n; j++) {**

**min = min + arr[j] - arr[j - k];**

**if (min < sum) {**

**min = sum; index = j;**

**}}**

**cout << (index - k + 2);**

**FIB**

**long long dp[100];**

**void fib();**

**int main(){**

**fib();**

**int i, j, k, made, num, x, y, z;**

**cin>>num;**

**if(num<=2){**

**if(num==0) cout<<"0 0 0"<<endl;**

**else if(num==1) cout<<"0 0 1"<<endl;**

**else cout<<"0 1 1"<<endl;**

**}else{**

**bool ck=false;**

**for(i=0; dp[i]<num; i++){**

**for(j=i; dp[j]<num; j++){**

**for(k=j; dp[k]<num; k++){**

**made = dp[i]+dp[j]+dp[k];**

**if(made==num){**

**x = dp[i]; y = dp[j]; z = dp[k]; ck=true; break;**

**}else if(made>num){**

**ck=false;**

**break;**

**}**

**}if(ck==true) break;**

**}if(ck==true) break;**

**}if(made!=num) printf("I'm too stupid to solve this problem\n");**

**else printf("%d %d %d\n", x, y, z);**

**}return 0; }**

**void fib(){long long a=0, b=1, c;**

**for(int i=0; i<=50; i++){**

**c = a+b; a=b; b=c; dp[i]=a; }}**