**ETHNUS ASSIGNMENT - 2**

Q1

Given the value of the number of elements ‘n’ and the value of ‘r’ which are the number of elements, we can choose from (n>=r). Find out the number of permutations and combinations of these elements.

The input format is as follows,

The first line is the value of n and the second line is the value of r.

Both n and r are integers

The output consists of two lines, the first line with the number of permutations, the second line would be the number of combinations

Test Case #1

Input

6

2

Output

30

15

Test Case #2

Input

9

5

Output

30

15

nPr = n!/r!

nCr = n!/(n-r)!r!

*// Given the value of the number of elements ‘n’ and the value of ‘r’ which are the number of elements, we can choose from (n>=r). Find out the number of permutations and combinations of these elements.*

*// The input format is as follows,*

*// The first line is the value of n and the second line is the value of r.*

*// Both n and r are integers*

*// The output consists of two lines, the first line with the number of permutations, the second line would be the number of combinations*

*// Test Case #1*

*// Input*

*// 6*

*// 2*

*// Output*

*// 30*

*// 15*

*/\*!*

*\* Copyright (c) 2020 Abhishek Srivastava*

*\*/*

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

int fact(int n) {

int f=1;

for(int i=2; i<=n; i++)

f = f\*i;

return f;

}

*// int fact(int n) {*

*// return n == 1 ? 1 : n \* fact(n - 1);*

*// }*

*// nPr = n!/r!*

*// nCr = n!/(n-r)!r!*

int permutations(int n, int r) {

return fact(n)/fact(n-r);

}

int combinations(int n, int r) {

return fact(n)/(fact(n-r)\*fact(r));

}

void solve() {

int n, r;

cin >> n >> r;

cout << fact(n) << "-> " << fact(r) << "\n\n";

cout << permutations(n, r) << "\n";

cout << combinations(n, r);

}

int main(){

ios :: sync\_with\_stdio(false);

cin.tie(0);

int t; cin >> t;

while(t--)

solve();

return 0;

}

Q2

You have been given two arrays. You have to rotate the first array by ‘x’ to the right and rotate the second array by ‘x’ to the left. Output the resultant arrays

The input array will be space-separated integers. Arrays will be represented in the first two lines. The third line contains the value for ‘x’

Test Case #1

Input

4 5 6 7 8

10 11 12 13

2

Output

7 8 4 5 6

12 13 10 11

Test Case #2

Input

23 42 53 36 53 999640

52 53

7

Output

23 42 53 36 53 999 640

53 52

*// You have been given two arrays. You have to rotate the first array by ‘x’ to the right and rotate the second array by ‘x’ to the left. Output the resultant arrays*

*// The input array will be space-separated integers. Arrays will be represented in the first two lines. The third line contains the value for ‘x’*

*// Test Case #1*

*// Input*

*// 4 5 6 7 8*

*// 10 11 12 13*

*// 2*

*// Output*

*// 7 8 4 5 6*

*// 12 13 10 11*

*/\*!*

*\* Copyright (c) 2020 Abhishek Srivastava*

*\*/*

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

void rightRotate(vector<int>& v) {

int temp = v[size(v)-1];

for(int i=size(v)-1; i>=0; i--) {

v[i] = v[i-1];

}

v[0] = temp;

}

void leftRotate(vector<int>& v){

int temp = v[0];

for(int i=1; i<v.size(); i++) {

v[i-1] = v[i];

}

v[v.size()-1] = temp;

}

void solve() {

vector<int> a = {4, 5, 6, 7, 8};

vector<int> b = {10, 11, 12, 13};

int r; cin >> r;

int l=r;

*// right rotate A*

while(r--) rightRotate(a);

while(l--) leftRotate(b);

for(int it: a) cout << it << " ";

cout << "\n";

for(int it: b) cout << it << " ";

}

int main(){

ios :: sync\_with\_stdio(false);

cin.tie(0);

int t; cin >> t;

while(t--)

solve();

return 0;

}

Q3

A permutation is one of the ways in which a number of things can be ordered or arranged. Given a string of length ‘s’, provide all the possible permutations of the string ‘s’ and print each and every one of it.

Note that the string does not contain repetitive characters.

Input is a string with unique characters. Output is each line with a combination of the input.

Test Case #1

Input

abc

Output

abc

acb

bac

bca

cab

cba

Test Case #2

Input

jump

Output

jupm

jmup

jmpu

jpum

jpmu

ujmp

ujpm

umjp

umpj

upjm

upmj

mjup

mjpu

mujp

mupj

mpju

mpuj

pjum

pjmu

pujm

pumj

pmju

pmuj

*// A permutation is one of the ways in which a number of things can be ordered or arranged. Given a string of length ‘s’, provide all the possible permutations of the string ‘s’ and print each and every one of it.*

*// Note that the string does not contain repetitive characters.*

*// Input is a string with unique characters. Output is each line with a combination of the input.*

*// Test Case #1*

*// Input*

*// abc*

*// Output*

*// abc*

*// acb*

*// bac*

*// bca*

*// cab*

*// cba*

*/\*!*

*\* Copyright (c) 2020 Abhishek Srivastava*

*\*/*

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

void permute(string a, int l, int r) {

if (l == r) cout<<a<<endl;

else {

for (int i = l; i <= r; i++) {

swap(a[l], a[i]);

permute(a, l+1, r);

swap(a[l], a[i]);

}

}

}

void solve() {

string str; cin >> str;

int len = str.size();

permute(str, 0, len-1);

}

int main(){

ios :: sync\_with\_stdio(false);

cin.tie(0);

solve();

return 0;

}

Q4

You are given an array of integers. Write a program to print all the elements which are greater than the elements on the right of it.

The input will be a space-separated integer which will be the contents of the array. The output should be a space-separated list of leader array elements.

Note that the solution must not be O(N\*N) at time complexity, but must be better than quadratic time complexity.

Test Case #1

Input

7 6 4 5 0 1

Output

7 6 5 1

Test Case #2

Input

14 12 70 15 99 65 21 190

Output

99 190

*// You are given an array of integers. Write a program to print all the elements which are greater than the elements on the right of it.*

*// The input will be a space-separated integer which will be the contents of the array. The output should be a space-separated list of leader array elements.*

*// Note that the solution must not be O(N\*N) at time complexity, but must be better than quadratic time complexity.*

*// Test Case #1*

*// Input*

*// 7 6 4 5 0 1*

*// Output*

*// 7 6 5 1*

*/\*!*

*\* Copyright (c) 2020 Abhishek Srivastava*

*\*/*

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

void solve() {

int n, temp; cin >> n;

vector<int> v, suff(n);

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

cin >> temp;

v.push\_back(temp);

}

suff[n-1] = v[n-1];

for(int i=n-2; i>=0; i--) {

suff[i] = max(v[i+1], v[i]);

}

temp = -1;

for(int a: suff) {

if(a != temp) {

cout << a << " ";

temp = a;

}

}

}

int main(){

ios :: sync\_with\_stdio(false);

cin.tie(0);

solve();

return 0;

}

Q5

Given an array of integers, print the position of the integer where the sum of its elements on the left is equal to the sum of its elements on the right. The element itself is not counted during the process

The time complexity of the solution should be linear

*// Given an array of integers, print the position of the integer where the sum of its elements on the left is equal to the sum of its elements on the right. The element itself is not counted during the process*

*// The time complexity of the solution should be linear*

*/\*!*

*\* Copyright (c) 2020 Abhishek Srivastava*

*\*/*

#include <bits/stdc++.h>

using namespace std;

#define ll long long int

void solve() {

int n; cin >> n;

vector<int> v;

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

int temp; cin >> temp;

v.push\_back(temp);

}

int lS=0, rS=0;

int l=0, r=n-1;

while(l<r) {

if(lS < rS) {

lS += v[l];

l++;

} else {

rS += v[r];

r--;

}

}

cout << l << "\n";

}

int main(){

ios :: sync\_with\_stdio(false);

cin.tie(0);

solve();

return 0;

}