**MEDIUM LEVEL**

**PROBLEM 1:**

The factorial of the integer N , written N!, is defined as:

Calculate and print the factorial of a given integer.

Eg : Calculate factorial of 30 = 30\*29\*28\*..................\*3\*2\*1

SOLUTION:

import java.io.\*;

import java.math.\*;

import java.security.\*;

import java.text.\*;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.regex.\*;

import java.math.BigInteger.\*;

public class Solution {

// Complete the extraLongFactorials function below.

static void extraLongFactorials(int n) {

BigInteger fact=BigInteger.valueOf(1);

//BigInteger n = BigInteger.valueOf(m)

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

{

fact=fact.multiply(BigInteger.valueOf(i));

}

System.out.println(fact);

}

private static final Scanner scanner = new Scanner(System.in);

public static void main(String[] args) {

int n = scanner.nextInt();

scanner.skip("(\r\n|[\n\r\u2028\u2029\u0085])?");

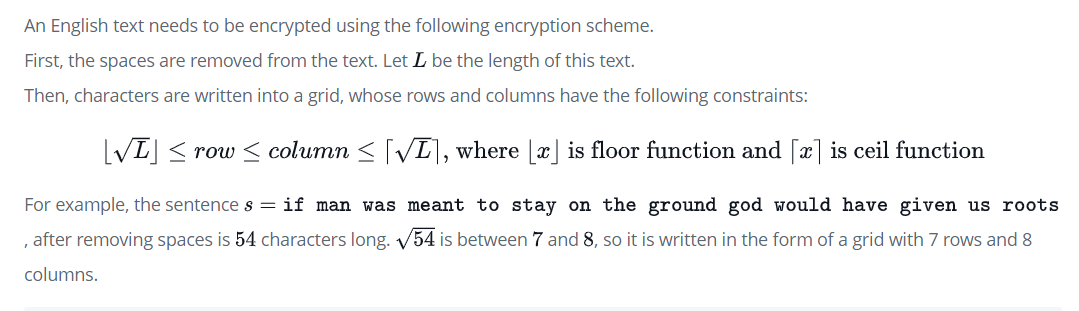
extraLongFactorials(n);

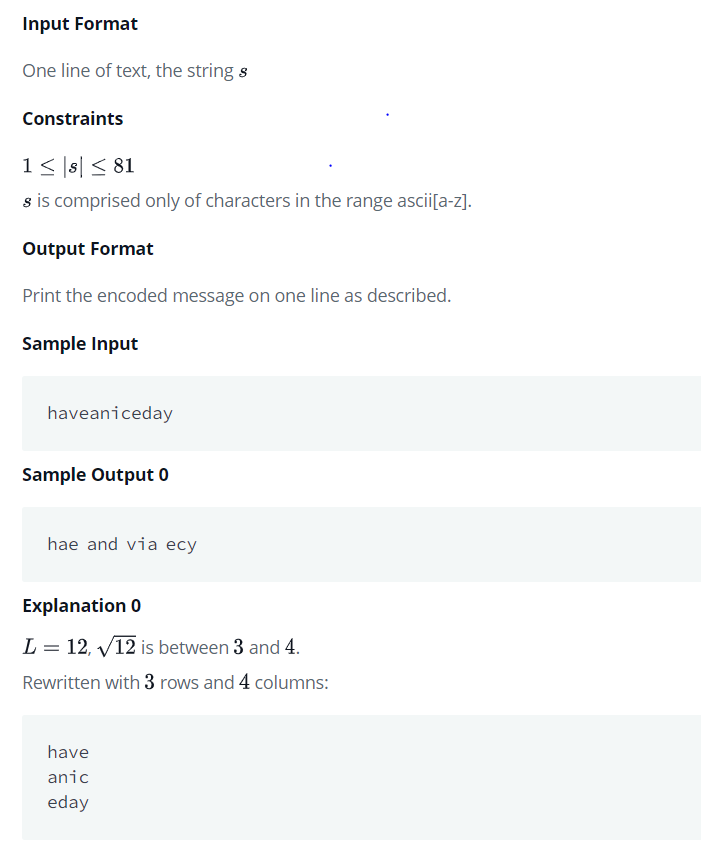
scanner.close();

}

}

**2. PROBLEM 2**





SOLUTION:

#include <algorithm>  
#include <cmath>  
#include <fstream>  
#include <iostream>  
#include <ostream>  
#include <string>  
using namespace std;string encryption(string& input) {   
 input.erase(std::remove(input.begin(), input.end(), ' '), input.end()); int size = input.length();  
 int lowerbound = (int) floor(sqrt(size));  
 int upperbound = (int) ceil(sqrt(size));  
 int total = upperbound \* lowerbound;  
   
 while (total < size) {  
 if (lowerbound < upperbound) {  
 ++lowerbound;  
 } else {  
 ++upperbound;  
 }  
 total = upperbound \* lowerbound;  
 }  
   
 char grid[lowerbound][upperbound] = {0};  
 int index = 0;  
   
 for (int row = 0; row < lowerbound; ++row) {  
 for (int col = 0; col < upperbound; ++col) {  
 if (index <= input.length() - 1) {  
 grid[row][col] = input.at(index);  
 ++index;  
 }  
 }  
 }  
   
 string encrypted = "";  
   
 for (int col = 0; col < upperbound; ++col) {  
 for (int row = 0; row < lowerbound; ++row) {  
 if (grid[row][col] != 0) {  
 encrypted += grid[row][col];  
 }  
 }  
   
 if (col != upperbound - 1) {  
 encrypted += " ";  
 }  
 }  
   
 return encrypted;  
}

int main() {  
 ofstream fout(getenv("OUTPUT\_PATH"));  
 string s;  
   
 getline(cin, s);

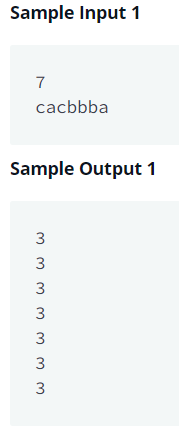
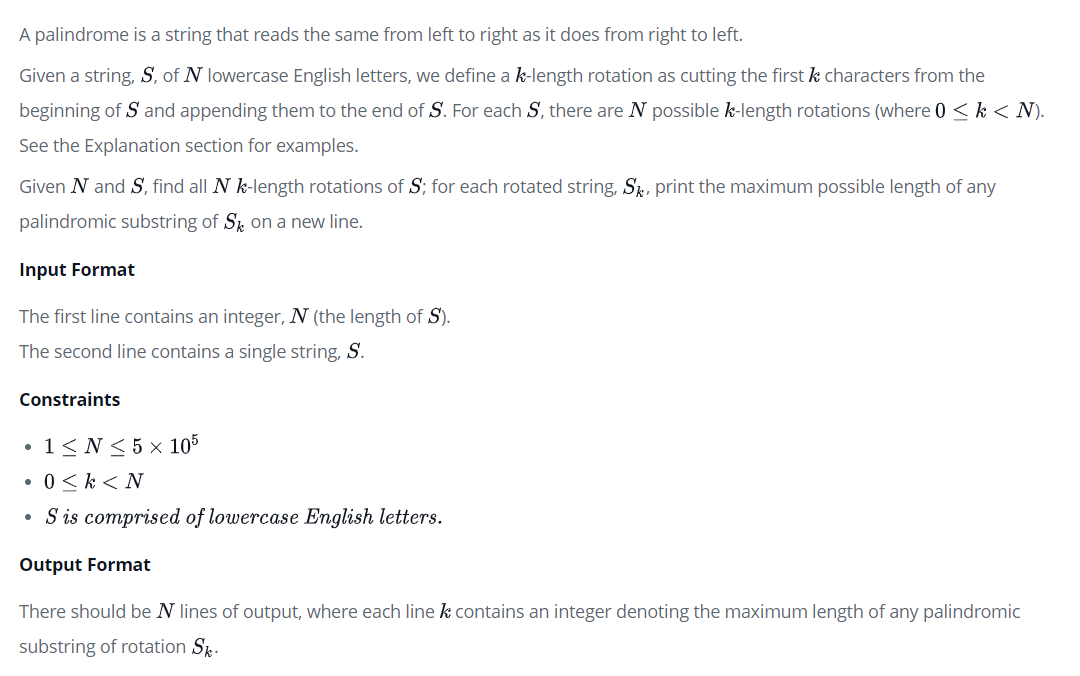
string result = encryption(s);

fout << result << "\n";

fout.close();  
 return 0;  
}

**COMPLEX LEVEL**

**PROBLEM 3:**



**SOLUTION**

import java.io.ByteArrayInputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.PrintWriter;

import java.util.Arrays;

import java.util.InputMismatchException;

public class E2 {

InputStream is;

PrintWriter out;

String INPUT = "";

void solve()

{

int n = ni();

char[] s = ns(n);

char[] s2 = new char[2\*n];

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

s2[i] = s2[i+n] = s[i];

}

int[] pal = palindrome(s2);

long[] es = new long[16\*n];

int p = 0;

for(int i = 0;i < 4\*n;i+=2){

pal[i] = Math.min(pal[i], n-((n&1)^1));

es[p++] = (long)(i/2)<<32|i;

es[p++] = (long)(i/2+pal[i]/2)<<32|i;

es[p++] = (long)(i/2+n-pal[i]/2-1)<<32|i;

es[p++] = (long)(i/2+n)<<32|i;

}

for(int i = 1;i < 4\*n;i+=2){

pal[i] = Math.min(pal[i], n-((n&1)));

es[p++] = (long)(i/2)<<32|i;

es[p++] = (long)(i/2+pal[i]/2)<<32|i;

es[p++] = (long)(i/2+n-pal[i]/2)<<32|i;

es[p++] = (long)(i/2+n)<<32|i;

}

Arrays.sort(es, 0, p);

MaxHeap inc = new MaxHeap(4\*n+1);

MaxHeap dec = new MaxHeap(4\*n+1);

MaxHeap flat = new MaxHeap(4\*n+1);

int[] st = new int[4\*n];

int q = 0;

for(int i = 0;i < 2\*n-1;i++){

while(q < p && es[q]>>>32 <= i){

int ind = (int)es[q];

if(st[ind] == 0){

inc.add(ind, (pal[ind]&1)-2\*i);

}else if(st[ind] == 1){

inc.remove(ind);

flat.add(ind, pal[ind]);

}else if(st[ind] == 2){

flat.remove(ind);

dec.add(ind, pal[ind]+2\*i);

}else if(st[ind] == 3){

dec.remove(ind);

}

st[ind]++;

q++;

}

if(i >= n-1){

int max = 0;

if(inc.size() > 0)max = Math.max(inc.max()+2\*i, max);

if(dec.size() > 0)max = Math.max(dec.max()-2\*i, max);

max = Math.max(flat.max(), max);

out.println(max);

}

}

}

public static class MaxHeap {

public int[] a;

public int[] map;

public int[] imap;

public int n;

public int pos;

public static int INF = Integer.MIN\_VALUE;

public MaxHeap(int m)

{

n = m+2;

a = new int[n];

map = new int[n];

imap = new int[n];

Arrays.fill(a, INF);

Arrays.fill(map, -1);

Arrays.fill(imap, -1);

pos = 1;

}

public int add(int ind, int x)

{

int ret = imap[ind];

if(imap[ind] < 0){

a[pos] = x; map[pos] = ind; imap[ind] = pos;

pos++;

up(pos-1);

}

return ret != -1 ? a[ret] : x;

}

public int update(int ind, int x)

{

int ret = imap[ind];

if(imap[ind] < 0){

a[pos] = x; map[pos] = ind; imap[ind] = pos;

pos++;

up(pos-1);

}else{

int o = a[ret];

a[ret] = x;

up(ret);

down(ret);

}

return x;

}

public int remove(int ind)

{

if(pos == 1)return INF;

if(imap[ind] == -1)return INF;

pos--;

int rem = imap[ind];

int ret = a[rem];

map[rem] = map[pos];

imap[map[pos]] = rem;

imap[ind] = -1;

a[rem] = a[pos];

a[pos] = INF;

map[pos] = -1;

up(rem);

down(rem);

return ret;

}

public int max() { return a[1]; }

public int argmax() { return map[1]; }

public int size() { return pos-1; }

private void up(int cur)

{

for(int c = cur, p = c>>>1;p >= 1 && a[p] < a[c];c>>>=1, p>>>=1){

int d = a[p]; a[p] = a[c]; a[c] = d;

int e = imap[map[p]]; imap[map[p]] = imap[map[c]]; imap[map[c]] = e;

e = map[p]; map[p] = map[c]; map[c] = e;

}

}

private void down(int cur)

{

for(int c = cur;2\*c < pos;){

int b = a[2\*c] > a[2\*c+1] ? 2\*c : 2\*c+1;

if(a[b] > a[c]){

int d = a[c]; a[c] = a[b]; a[b] = d;

int e = imap[map[c]]; imap[map[c]] = imap[map[b]]; imap[map[b]] = e;

e = map[c]; map[c] = map[b]; map[b] = e;

c = b;

}

else{

break;

}

}

}

}

public static int[] palindrome(char[] str)

{

int n = str.length;

int[] r = new int[2\*n];

int k = 0;

for(int i = 0, j = 0;i < 2\*n;i += k, j = Math.max(j-k, 0)){

// normally

while(i-j >= 0 && i+j+1 < 2\*n && str[(i-j)/2] == str[(i+j+1)/2])j++;

r[i] = j;

// skip based on the theorem

for(k = 1;i-k >= 0 && r[i]-k >= 0 && r[i-k] != r[i]-k;k++){

r[i+k] = Math.min(r[i-k], r[i]-k);

}

}

return r;

}

void run() throws Exception

{

is = INPUT.isEmpty() ? System.in : new ByteArrayInputStream(INPUT.getBytes());

out = new PrintWriter(System.out);

long s = System.currentTimeMillis();

solve();

out.flush();

if(!INPUT.isEmpty())tr(System.currentTimeMillis()-s+"ms");

}

public static void main(String[] args) throws Exception { new E2().run(); }

private byte[] inbuf = new byte[1024];

private int lenbuf = 0, ptrbuf = 0;

private int readByte()

{

if(lenbuf == -1)throw new InputMismatchException();

if(ptrbuf >= lenbuf){

ptrbuf = 0;

try { lenbuf = is.read(inbuf); } catch (IOException e) { throw new InputMismatchException(); }

if(lenbuf <= 0)return -1;

}

return inbuf[ptrbuf++];

}

boolean isSpaceChar(int c) { return !(c >= 33 && c <= 126); }

int skip() { int b; while((b = readByte()) != -1 && isSpaceChar(b)); return b; }

double nd() { return Double.parseDouble(ns()); }

char nc() { return (char)skip(); }

String ns()

{

int b = skip();

StringBuilder sb = new StringBuilder();

while(!(isSpaceChar(b))){

sb.appendCodePoint(b);

b = readByte();

}

return sb.toString();

}

char[] ns(int n)

{

char[] buf = new char[n];

int b = skip(), p = 0;

while(p < n && !(isSpaceChar(b))){

buf[p++] = (char)b;

b = readByte();

}

return n == p ? buf : Arrays.copyOf(buf, p);

}

private char[][] nm(int n, int m)

{

char[][] map = new char[n][];

for(int i = 0;i < n;i++)map[i] = ns(m);

return map;

}

private int[] na(int n)

{

int[] a = new int[n];

for(int i = 0;i < n;i++)a[i] = ni();

return a;

}

private int ni()

{

int num = 0, b;

boolean minus = false;

while((b = readByte()) != -1 && !((b >= '0' && b <= '9') || b == '-'));

if(b == '-'){

minus = true;

b = readByte();

}

while(true){

if(b >= '0' && b <= '9'){

num = num \* 10 + (b - '0');

}else{

return minus ? -num : num;

}

b = readByte();

}

}

private long nl()

{

long num = 0;

int b;

boolean minus = false;

while((b = readByte()) != -1 && !((b >= '0' && b <= '9') || b == '-'));

if(b == '-'){

minus = true;

b = readByte();

}

while(true){

if(b >= '0' && b <= '9'){

num = num \* 10 + (b - '0');

}else{

return minus ? -num : num;

}

b = readByte();

}

}

static void tr(Object... o) { System.out.println(Arrays.deepToString(o)); }

}