1. -Frequency of Characters

vector<int> Solution::solve(string A) {

    vector<int> v(26,0);

    for(int i=0;i<A.length();i++){

        v[A[i]-'a']++;

    }

    return v;

}

1. Maximum substring

int Solution::solve(string A, int B) {

int res=INT\_MIN,count=0,i;

for(i=0;A.length()-i>=B;i=i+B){

count = 0;

for(int j=i;j<i+B;j++){

if(A[j]=='a'){

count++;

}

}

res = max(res,count);

}

count=0;

for(int j=i;j<A.length();j++){

if(A[j]=='a'){

count++;

}

}

res = max(res,count);

return res;

}

1. String Inversion

string Solution::solve(string A) {

    for(int i=0;i<A.length();i++){

        if(A[i]>='a' && A[i]<='z') A[i]=A[i]-32;

        else A[i]=A[i]+32;

    } return A;

}

1. Valid Password

int Solution::solve(string A) {

    int a=0,b=0,c=0,d=0,e=0;

    if(A.length()>7 && A.length()<16) b=1;

    for(int i=0;i<A.length();i++) {

        if(A[i]>='0' && A[i]<='9') a=1;

        if(A[i]>='a' && A[i]<='z') c=1;

        if(A[i]>='A' && A[i]<='Z') d=1;

        if(A[i]=='@' || A[i]=='#' || A[i]=='%' || A[i]=='&' || A[i]=='!' || A[i]=='$' || A[i]=='\*') e=1;

    }

    if(a && b && c && d && e) return 1;

    return 0;

}

1. Palindromic Words

bool check (string s){

    for(int i=0;i<s.length()/2;i++) {

        if(s[i]!=s[s.length()-i-1]) return 0;

    }

    return 1;

}

int Solution::solve(string A) {

    string s=""; int c=0;

    for(int i=0;i<A.length();i++){

        if(A[i]==' '){

            if(check(s)) {c++;}

            s="";

        }

        else s=s+A[i];

    }

    if(check(s)) {c++;}

    return c;

}

1. Palindrome String

int Solution::isPalindrome(string A) {

int flag = true;

transform(A.begin(), A.end(), A.begin(), ::tolower);

string s1 = "";

for(int i = 0; i<A.size(); i++){

if(isalnum(A[i])) s1.push\_back(A[i]);

}

int N = s1.length()-1;

for(int j = 0; j<=N/2; j++){

if(s1[j]!=s1[N-j]){

flag = false;

break;

}

}

return flag;

}

1. Longest Common Prefix //gg

string Solution::longestCommonPrefix(vector<string> &A) {

    string s="";

    sort(A.begin(),A.end());

    for(int i=0;i<min(A[0].length(),A[(A.size()-1)].length());i++){

        char c1=A[0][i],c2=A[A.size()-1][i];

        if(c1==c2) s=s+c1;

    }

    return s;

}

1. Length of last word

int Solution::lengthOfLastWord(const string A) {

    string lw=""; int l=0;

    for(int i=0;i<A.length();i++) {

        if(A[i]==' ' && lw!="") {

            l=lw.length();lw="";

        }

        else if(A[i]!=' ') lw=lw+A[i];

    } if(lw!="") return lw.length();

    else return l; //l bcoz of case “hello “

}

1. Count and Say

string giveString(string str){

    string res="";

    char ch=str[0];

    int cnt=1;

    for(int i=1;i<str.size();i++){

        if(str[i]!=ch){

            res+=to\_string(cnt);

            res+=ch;

            cnt=1;

            ch=str[i];

        }

        else{

            cnt++;

        }

    }

    res+=to\_string(cnt);

    res+=ch;

    return res;

}

string Solution::countAndSay(int n) {

    vector<string>dp(n+1);

    dp[1]="1";

    if(n==1)return dp[1];

    dp[2]="11";

    for(int i=3;i<=n;i++){

        string temp=giveString(dp[i-1]);

        dp[i]=temp;

    }

    return dp[n];

}

1. Reverse the string

string Solution::solve(string A) {

    string ans="";

    vector<string> s;

    for(int i=0;i<A.length();i++){

        char ch=A[i];

        if(A[i]==' ') {continue;}

        string k="";

        while(A[i]!=' ' && i<A.length()) {k=k+A[i];i++;}

        s.push\_back(k);

    }

    for(int i=s.size()-1;i>0;i--){

        ans=ans + s[i]+' ';

    }

    ans=ans + s[0];

    return ans;

}

1. Add Binary strings

string Solution::addBinary(string A, string B) {

while(A.size() < B.size()) A = '0'+A;

while(A.size() > B.size()) B = '0'+B;

string ret = "";

int c = 0;

int n = A.size();

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

int s = A[i]+B[i]-2\*'0';

ret = (char)((s+c)%2 + '0') + ret; //The sum bit will be 1 if, either all of the 3 bits are set or one of them is set.  
So we can add all the bits and then take remainder with 2 to get the current bit in the answer.

c = (s+c)/2; //Carry will be 1 if any of the two bits is set. So we can find carry by **adding the bits and dividing the result by 2**.

}

if(c) ret = '1'+ret;

return ret;

}

1. Compare version numbers

int Solution::compareVersion(string A, string B) {

// Do not write main() function.

// Do not read input, instead use the arguments to the function.

// Do not print the output, instead return values as specified

// Still have a doubt. Checkout www.interviewbit.com/pages/sample\_codes/ for more details

int j, i;

for( i=0, j=0 ; i<A.size() || j<B.size() ; i++, j++){

unsigned long long num1 = 0, num2 = 0;

while(i < A.size() && A[i] != '.'){

num1 \*= 10;

num1 += A[i] - '0';

i++;

}

while(j < B.size() && B[j] != '.'){

num2 \*= 10;

num2 += B[j] - '0';

j++;

}

if(num1 > num2) return 1;

if(num1 < num2) return -1;

}

return 0;

}

1. Amazing Subarrays

int Solution::solve(string A) {

    int n=A.length();

    int ans=0;

    for(int i=0;i<n;i++){

        if(A[i]>='a' && A[i]<='z') A[i]=A[i]-32;

        if(A[i]=='A' || A[i]=='E' || A[i]=='O' || A[i]=='I' || A[i]=='U'){

            ans+=n-i;

        }

    } return ans%10003;

}

1. Convert to palindrome

//naïve approach gave TLE

int Solution::solve(string A) {

int count = 0, n = A.size();

int i = 0, j = n-1;

while(i<j){

if (A[i] == A[j]){

i++;

j--;

}

else{

if (A[i+1] == A[j]){

i++;

count++;

}

else if (A[i] == A[j-1]){

j--;

count++;

}

else{

return 0;

}

}

}

return (count <= 1);

}

1. Vowel and Consonants Substring

int Solution::solve(string A) {

long int l = A.length();

long int mod = 1000000007;

int vowel =0, cons =0;

long int sum=0;

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

A[i] = tolower(A[i]);

if(A[i]=='a' || A[i]=='e' || A[i]=='i'|| A[i]=='o'|| A[i]=='u'){

sum = ((sum+ cons%mod)%mod);

vowel++;

}

else{

sum = ((sum+ vowel%mod)%mod);

cons++;

}

}

return sum;

}

1. Minimum Parentheses

int Solution::solve(string A) {

    stack<char> s; int ans=0;

    for(int i=0;i<A.length();i++){

        if(A[i]=='(') s.push('(');

        else {if(!s.empty()) s.pop(); else ans++;}

    }

    return ans+s.size();

}

1. Character Frequencies //without map

vector<int> Solution::solve(string A) {

vector<int>v;

int l=A.length();

int c[26]={0};

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

c[A[i]-'a']++;

}

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

if(c[A[i]-'a']!=0){

v.push\_back(c[A[i]-'a']);

c[A[i]-'a']=0;

}

}

return v;

}

1. Remove consecutive characters

string Solution::solve(string A, int B) {

string ans="";

int c=1;

char ch=A[0];

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

{

if(A[i]==ch)

{

c++;

}

else

{

if(c!=B)

{

for(int j=0;j<c;j++)

ans+=ch;

}

ch=A[i];

c=1;

}

}

if(c!=B)

{

for(int j=0;j<c;j++)

ans+=ch;

}

return ans;

}

1. Implement strStr

int Solution::strStr(const string A, const string B) {

//A is the haystack and the B is the needle

//we have to find the first index of needle in the haystack and return it

//first we need to find whether needle exists in haystack or not

// we can create a window that checks whether the string in the window is the correct or not

int size\_of\_window=B.size();

int n=A.size();

int i=0,j=0;

int index=0;

int flag=0;

while(i<n)

{

if(A[i]!=B[j])

{

j=0;

// int temp=i;

i=index+1;

index=i;

continue;

}

else

j++;

if(j==B.size())

{ flag=1;break;}

i++;

}

if(index>=n || flag==0)

return -1;

return index;

}

1. Integer to Roman

string Solution::intToRoman(int A) {

string ones[] = {"","I","II","III","IV","V","VI","VII","VIII","IX"};

string tens[] = {"","X","XX","XXX","XL","L","LX","LXX","LXXX","XC"};

string hrns[] = {"","C","CC","CCC","CD","D","DC","DCC","DCCC","CM"};

string thns[] = {"","M","MM","MMM"};

return thns[A/1000] + hrns[(A%1000)/100] + tens[(A%100)/10] + ones[A%10];

}

1. Roman to Integer

int Solution::romanToInt(string A) {

map<char,int> m;

m['I'] = 1;

m['V'] = 5;

m['X'] = 10;

m['L'] = 50;

m['C'] = 100;

m['D'] = 500;

m['M'] = 1000;

int ans = 0, i = 0, N = A.size();

while(i < N-1){

if(m[A[i]]< m[A[i+1]]) ans-=m[A[i]];

else ans+=m[A[i]];

i++;

}

ans+=m[A[i]];

return ans;

}

1. Multilpy strings //can’t multiply nos. & convert to string, coz product will be vvlarge no.

string Solution::multiply(string A, string B) {  
int n = A.length(),m = B.length();  
string res(n+m,‘0’);

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

for(int j=m-1;j>=0;j--){

int num = (A[i] - '0') \* (B[j] - '0') + res[i+j+1] - '0';

res[i+j+1] = num%10 + '0';

res[i+j] += num/10;

}

}

for(int i=0;i<res.length();i++) if(res[i] != '0') return res.substr(i);

return "0";

}

1. Zigzag String

string Solution::convert(string A, int B) {

int n = A.size();

/\*if (n <= B) {

return A;

}\*/

if (B <= 1) {

return A;

}

vector <int> place(n,0);

string ret[B];

for (int i=0;i<B;i++)

ret[i]="";

bool down = true;

int count = 0;

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

{

if (down)

{

if (count < B)

{

ret[count] +=A[i];

count++;

i++;

}

else

{

count -= 2;

down = false;

}

}

else

{

if (count >=0)

{

ret[count] +=A[i];

count--;

i++;

}

else

{

count += 2;

down = true;

}

}

}

string ans = "";

for (int i=0;i<B;i++)

ans+=ret[i];

//cout<<ans<<endl;

return ans;

// Do not write main() function.

// Do not read input, instead use the arguments to the function.

// Do not print the output, instead return values as specified

// Still have a doubt. Checkout www.interviewbit.com/pages/sample\_codes/ for more details

}

1. Atoi

int Solution::atoi(const string &A) {

long long x;

int mul = 1;

int i = 0;

while(A[i]==' ') i++;

if(A[i]=='-')

{

mul = -1;

i++;

}

if(A[i]=='+') i++;

x = 0;

while(i<A.size())

{

if(isdigit(A[i]))

x = x\*10 + (A[i]-'0');

else

break;

if((mul==1 && x>1LL\*INT\_MAX) || (mul==-1 && x>abs(1LL\*INT\_MIN)))

break;

i++;

}

x\*=mul;

x = max(1LL\*INT\_MIN,x);

x = min(x,1LL\*INT\_MAX);

return x;

// Do not write main() function.

// Do not read input, instead use the arguments to the function.

// Do not print the output, instead return values as specified

// Still have a doubt. Checkout www.interviewbit.com/pages/sample\_codes/ for more details

}

1. Valid ip address

bool valid(string &s) {

if(!s.size()) return false;

if(s[0] == '0' && s.size() > 1) return false;

stringstream ss(s);

int num;

ss >> num;

return num < 256;

}

vector<string> Solution::restoreIpAddresses(string s) {

vector<string> res;

for(int i = 0; i < s.size(); ++ i) {

string a = s.substr(0, i + 1); if(!valid(a)) break;

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

string b = s.substr(i + 1, j - i); if(!valid(b)) break;

for(int k = j + 1; k < s.size(); ++ k) {

string c = s.substr(j + 1, k - j); if(!valid(c)) break;

string d = s.substr(k + 1); if(!valid(d)) continue;

res.push\_back(a + "." + b + "." + c + "." + d);

}

}

}

return res;

}

1. Longest Palindromic substring

//LCS on A,reverse(A) doesn’t work here, only gives LC Subsequence

string Solution::longestPalindrome(string A) {

int n=A.size(),i,j,k,max=1,max\_i=0,max\_j=0;

for(k=0;k<n-1;k++){

i=k;j=k+1;

while(i >= 0 && i < n && j >= 0 && j < n && A[i]==A[j]){

if(max<j-i+1){

max=j-i+1;

max\_i=i;max\_j=j;

}

i--;j++;

}

}

//cout<<max\_i<<" "<<max\_j<<endl;

for(k=1;k<n-1;k++){

i=k-1;j=k+1;

while(i >= 0 && i < n && j >= 0 && j < n && A[i]==A[j]){

if(max<j-i+1){

max=j-i+1;

max\_i=i;max\_j=j;

}

i--;j++;

}

}

//cout<<max\_i<<" "<<max\_j<<endl;

//return A.substr(max\_i, max\_j-max\_i+1);

// cout<<max\_i<<" "<<max\_j<<endl;

string res="";//(A.begin()+max\_i,A.begin()+max\_j+1);

for(i=max\_i;i<=max\_j;i++)res+=A[i];

return res;

}

1. Serialize

string Solution::serialize(vector<string> &A) {

    string ans="";

    for(int i=0;i<A.size();i++){

        string x=A[i];

        ans+=A[i];

        ans=ans+to\_string(A[i].length())+"~";

    } return ans;

}

1. Self permutation

int Solution::permuteStrings(string A, string B) {

    int a[26],b[26];

    for(int i=0;i<26;i++) a[i]=0,b[i]=0;

    if(A.length()!=B.length()) return 0;

    for(int i=0;i<A.length();i++){

        a[A[i]-'a']++;

    }

    for(int i=0;i<B.length();i++){

        b[B[i]-'a']++;

    }

    for(int i=0;i<26;i++){

        if(a[i]!=b[i]) return 0;

    } return 1;

}

1. Salutes //ez but tricky

long Solution::countSalutes(string A) {

    long ans=0;

    long count1=0;

    for(auto i:A){

        if(i=='<')ans+=count1;

        if(i=='>')count1++;

    }

    return ans;

}

1. String and its frequency

string Solution::solve(string A) {

    int ans[26]; string s="";

    for(int i=0;i<26;i++) ans[i]=0;

    for(int i=0;i<A.length();i++) ans[A[i]-'a']++;

    for(int i=0;i<A.length();i++){

        if(ans[A[i]-'a']) {s+=A[i]+to\_string(ans[A[i]-'a']);

        ans[A[i]-'a']=0;}

    } return s;

}

1. Deserialize

vector<string> Solution::deserialize(string A) {

    vector<string> v;

    string word="";

    for(int i=0;i<A.length();i++){

        if(A[i]>='a' && A[i]<='z') word+=A[i];

        else { if(word!=""){

            v.push\_back(word); word="";}

        }

    } return v;

}

1. Bulls and Cows

string Solution::solve(string secret, string guess) {

int bull=0;

int arr[10]={0};

for(int i=0;i<secret.size();i++) {

if(secret[i]==guess[i]) arr[guess[i]-'0']--, bull++;

arr[guess[i]-'0']++;

}

int cow=0;

int j=0;

for(auto i: secret) {

if(i!=guess[j++] && arr[i-'0']>0) cow++, arr[i-'0']--;

}

string s=to\_string(bull) + "A" + to\_string(cow) + "B";

return s;

}

1. Convert the amount in number to words //useless

// strings at index 0 is not used

// to make indexing simple

//Please note we will add space after every word

string one[] = { "", "one-", "two-", "three-", "four-",

"five-", "six-", "seven-", "eight-",

"nine-", "ten-", "eleven-", "twelve-",

"thirteen-", "fourteen-", "fifteen-",

"sixteen-", "seventeen-", "eighteen-",

"nineteen-"

};

// strings at index 0 and 1 are not used

//to make array indexing simple

string ten[] = { "", "", "twenty-", "thirty-", "forty-",

"fifty-", "sixty-", "seventy-", "eighty-",

"ninety-"

};

// n is 1- or 2-digit number

string numToWords(int n, string s)

{

string str = "";

// if n is more than 19, divide it

if (n > 19)

str += ten[n / 10] + one[n % 10];

else

str += one[n];

// if n is non-zero

if (n)

str += s;

return str;

}

// Function to print a given number in words

string convertToWords(int n)

{

// stores word representation of given number n

string out;

// handles digits at ten crore and crore places (if any)

out += numToWords((n / 10000000), "crore-");

// handles digits at ten lakh and lakh places (if any)

out += numToWords(((n / 100000) % 100), "lakh-");

// handles digits at thousands and tens thousands

// places (if any)

out += numToWords(((n / 1000) % 100), "thousand-");

// handles digit at hundreds places (if any)

out += numToWords(((n / 100) % 10), "hundred-");

// we need to add "and" if the number is more than hundred and contains digit at ten's or one's place

if (n > 100 && n % 100)

out += "and-";

// handles digits at ones and tens places (if any)

out += numToWords((n % 100), "");

out = out.substr(0, out.size()-1); // to remove the last trailing "-"

return out;

}

int Solution::solve(const string A, const string B) {

int n = stoi(A);

string y = convertToWords(n);

if(y.compare(B)==0)

return 1;

return 0;

}

1. Minimum appends for palindrome

int palindrome(string s)

{

for(int i=0;i<s.size()/2;i++)

{

if(s[i]!=s[s.size()-i-1])

return 0;

}

return 1;

}

int Solution::solve(string a) {

int l=a.size();

if(palindrome(a)==1)

return 0;

int c=0;

while(l--)

{

string s=a.substr(++c);

if(palindrome(s)==1)

{

break;

}

}

return c;

}

1. Pretty Json

vector<string> Solution::prettyJSON(string A) {

int n=A.length(),i,j,tb=0;

string tp="";

vector<string> ans;

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

{

if(A[i]==' ')

continue;

else if(A[i]=='{' || A[i]=='[')

{

if(tp!="")

ans.push\_back(tp);

tp="";

for(int k=0;k<tb;k++)

tp+="\t";

tp+=A[i];

ans.push\_back(tp);

tp="";

tb++;

}

else if(A[i]==']'|| A[i]=='}')

{

if(tp!="")

ans.push\_back(tp);

tp="";

for(int k=0;k<tb-1;k++)

tp+="\t";

tp+=A[i];

if(i<A.length() && A[i+1]==',')

{

tp+=A[++i];

}

ans.push\_back(tp);

tp="";

tb--;

}

else

{

if(tp=="")

{

for(int k=0;k<tb;k++)

tp+="\t";

}

tp+=A[i];

if(A[i]==',')

{

ans.push\_back(tp);

tp="";

}

}

}

return ans;

}

1. Power of 2 //op solution

int Solution::power(string A) {

    long double  t=stold(A);

    if(t==1)return 0;

    return floor(log2(t))==ceil(log2(t));

}

1. Min characters required to make a string palindromic

int Solution::solve(string A) {

int n= A.length();

int palen;

int count=0;

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

palen=0;

for(int j=0;j<=(n-i)/2;j++){

if(A[j]==A[n-i-j-1])

continue;

else

{palen=1;

break;}

}

if(palen==1)

count++;

else

break;

}

return count;

}

1. Justified text //didn’t get it
2. Stringholics //don’t know