*Recursion and Backtracking Questions:*

*1• Reverse a string using recursion*

*[Practice here:* [*https://www.geeksforgeeks.org/reverse-a-string-using-recursion/*](https://www.geeksforgeeks.org/reverse-a-string-using-recursion/) *]*

*#include<bits/stdc++.h>*

**using** **namespace** std**;**

string reverse**(**string**,int);**

**int** main**()**

**{**

string s**;**

cin**>>**s**;**

**int** x**=**s**.**length**();**

string s1**=**s**;**

cout**<<**reverse**(**s**,**x**-1);**

**return** **0;**

**}**

string reverse**(**string s**,int** x**)**

**{**

**if(**x**<0)**

**return** ""**;**

**return** s**[**x**]+**reverse**(**s**,**x**-1);}**

*• 2Reverse a Stack using recursion*

*[Practice here:* [*https://www.geeksforgeeks.org/reverse-a-stack-using-recursion/*](https://www.geeksforgeeks.org/reverse-a-stack-using-recursion/) *]*

*#include<bits/stdc++.h>*

**using** **namespace** std**;**

**void** reversestack**();**

**void** putinstack**(int);**

stack**<int>** st**;**

**int** main**()**

**{**

**int** n**,**x**,**i**;**

cin**>>**n**;**

stack**<int>** st**;**

**while(**n**--)**

**{** cin**>>**x**;**st**.**push**(**x**);}**

reversestack**();**

**while(!**st**.**empty**())**

**{**

cout**<<**st**.**top**();**

st**.**pop**();**

**}**

**return** **0;**

**}**

**void** reversestack**()**

**{**

**if(**st**.**empty**())**

**{return;}**

**int** e**=**st**.**top**();**

st**.**pop**();**

reversestack**();**

putinstack**(**e**);}**

**void** putinstack**(int** e**)**

**{**

**if(**st**.**empty**())**

**{**st**.**push**(**e**);}**

**else** **{**

**int** x**=**st**.**top**();**

st**.**pop**();**

putinstack**(**e**);**

st**.**push**(**x**);**

**}**

**}**

*•3 Check if a number is Palindrome or not*

*[Practice here:* [*https://practice.geeksforgeeks.org/problems/palindrome/0*](https://practice.geeksforgeeks.org/problems/palindrome/0) *]*

*#include<bits/stdc++.h>*

using namespace std;

string reverse(string,int);

int main()

{

string s;

cin>>s;

int x=s.length();

string s1=s;

string s2=reverse(s,x-1);

if(s1.compare(s2)==0)

cout<<"Palindrome";

else

cout<<"not Palindrome";

return 0;

}

string reverse(string s,int x)

{

if(x<0)

return "";

return s[x]+reverse(s,x-1);

}

*• 4Generate all binary strings without consecutive 1’s*

*[Practice here:* [*https://www.geeksforgeeks.org/generate-binary-strings-without-*](https://www.geeksforgeeks.org/generate-binary-strings-without-)

[*consecutive-1s/*](https://www.geeksforgeeks.org/generate-binary-strings-without-) *]*

*#include<bits/stdc++.h>*

using namespace std;

void getnum(int [],int,int);

void getbinary(int [],int,int);

int main()

{

int n;

cin>>n;*//n=4*

int a[n];

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

a[i]=0;

n=pow(2,n);*//n=16*

getnum(a,0,n);

return 0;

}

void getnum(int a[],int n,int m)

{*//n=0;m=16*

if(n>=m)

return;

getbinary(a,n,m);

getnum(a,n+1,m);

}

void getbinary(int a[],int n,int m)

{int i=0;

bool x=false;*//x=0;n=0;m=16*

while(n!=0)

{

a[i]=n%2;

n=n/2;

i++;

if(a[i-1]==1 && a[i]==1){

x=true;

break;

}

}

m=sqrt(m);

if(!x){

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

cout<<a[m-i-1];

cout<<" ";

}

}

*• 5• Program for length of a string using recursion*

*[Practice here:* [*https://www.geeksforgeeks.org/program-for-length-of-a-string-using-*](https://www.geeksforgeeks.org/program-for-length-of-a-string-using-)

[*recursion/*](https://www.geeksforgeeks.org/program-for-length-of-a-string-using-) *]*

*#include <bits/stdc++.h>*

**using** **namespace** std**;**

**int** recLen**(char\*** str**)**

**{**

**if** **(\***str **==** '\0'**)**

**return** **0;**

**else**

**return** **1** **+** recLen**(**str **+** **1);**

**}**

**int** main**()**

**{**

**char** str**[]** **=** "abc"**;**

cout **<<** recLen**(**str**);**

**return** **0;**

**}**

*• 6• Combinations in a String of Digits*

*[Practice here:* [*https://www.geeksforgeeks.org/combinations-string-digits/*](https://www.geeksforgeeks.org/combinations-string-digits/) *]*

#include<iostream>

**using** **namespace** std;

void findnumbers(char\* input,int inputindex,char\* output,int outputindex )

{

**if**(input[inputindex]=='\0')

{

cout<<output<<endl;

**return** ;

}

output[outputindex]=input[inputindex];

output[outputindex+1]=' ';

findnumbers(input,inputindex+1,output,outputindex+2);

**if**(input[inputindex+1]!='\0')

{

output[outputindex+1]=input[inputindex+1];

findnumbers(input,inputindex+1,output,outputindex+1);

}}

int main()

{

char input[]="1234";

char\* output=**new** char[100];

output[0]='\0';

findnumbers(input,0,output,0);

**return** 0;

}

• 7Count consonants in a string (Iterative and recursive methods)

[Practice here: <https://www.geeksforgeeks.org/count-consonants-string-iterative->

[recursive-methods/](https://www.geeksforgeeks.org/count-consonants-string-iterative-) ]

#include<iostream>

**using** **namespace** std;

int findconsonant(string s,int i){

**if**(s[i]=='\0')

**return** 0;

**if**(s[i]!='a'&& s[i]!='e' && s[i]!='i' && s[i]!='o' && s[i]!='u')

**return** 1+findconsonant(s,i+1);

**else** **return** findconsonant(s,i+1);}

int main()

{string s;

cin>>s;

cout<<findconsonant(s,0);

**return** 0;}

*•8 Program for length of a string using recursionFirst uppercase letter in a*

*string (Iterative and Recursive)*

*[Practice here:* [*https://www.geeksforgeeks.org/first-uppercase-letter-in-a-string-*](https://www.geeksforgeeks.org/first-uppercase-letter-in-a-string-)

[*iterative-and-recursive/*](https://www.geeksforgeeks.org/first-uppercase-letter-in-a-string-) *]*

#include<iostream>

**using** **namespace** std;

char finduper(string s,int i)

{

**if**(s[i]=='\0')

**return** '0';

**if**(s[i]>=65&&s[i]<=90)

**return** s[i];

**else** **return** finduper(s,i+1);

}

int main()

{

string s;

cin>>s;

cout<<finduper(s,0);

**return** 0;

}

*•9 Print all possible combinations of r elements in a given array of size n*

*[Practice here:* [*https://www.geeksforgeeks.org/print-all-possible-combinations-of-r-*](https://www.geeksforgeeks.org/print-all-possible-combinations-of-r-)

[*elements-in-a-given-array-of-size-n/*](https://www.geeksforgeeks.org/print-all-possible-combinations-of-r-) *]*

#include<iostream>

**using** **namespace** std;

void printarray(int data[],int a[],int index,int start,int r,int n)

{

**if**(index==r)

{

**for**(int count=0;count<r;count++)

cout<<data[count];

cout<<endl;

**return**;

}

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

data[index]=a[i];

printarray(data,a,index+1,i+1,r,n);}}

int main()

{

int n,r;

cin>>n;

int a[n];

int data[n];

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

cin>>a[i];

cin>>r;

printarray(data,a,0,0,r,n-1);

**return** 0;

}

Alternative solution:

Like the above method, We create a temporary array data[]. The idea here is similar to [Subset Sum Problem](https://www.geeksforgeeks.org/dynamic-programming-subset-sum-problem/). We one by one consider every element of input array, and recur for two cases:

1) The element is included in current combination (We put the element in data[] and increment next available index in data[])

2) The element is excluded in current combination (We do not put the element and do not change index)

When number of elements in data[] become equal to r (size of a combination), we print it.

This method is mainly based on [Pascal’s Identity](http://en.wikipedia.org/wiki/Pascal's_rule), i.e. **ncr = n-1cr + n-1cr-1**

void combinationUtil(int arr[], int n, int r,

int index, int data[], int i)

{

**if** (index == r)

{

**for** (int j = 0; j < r; j++)

cout << data[j] << " ";

cout << endl;

**return**;

}

**if** (i >= n)

**return**;

*// current is included, put next at next location*

data[index] = arr[i];

combinationUtil(arr, n, r, index + 1, data, i + 1);

*// current is excluded, replace it with next (Note that*

*// i+1 is passed, but index is not changed)*

combinationUtil(arr, n, r, index, data, i+1);

}

*•10 • Print all increasing sequences of length k from first n natural numbers*

*[Practice here:* [*https://www.geeksforgeeks.org/print-increasing-sequences-length-k-*](https://www.geeksforgeeks.org/print-increasing-sequences-length-k-)

[*first-n-natural-numbers/*](https://www.geeksforgeeks.org/print-increasing-sequences-length-k-) *]*

#include<iostream>

**using** **namespace** std;

void printseq(int data[],int n,int start,int index,int k)

{

**if**(index==k)

{

**for**(int count=0;count<k;count++)

cout<<data[count];

cout<<endl;

**return** ;

}

**for**(int i=start;i<=n;i++)

{

data[index]=i;

printseq(data,n,i+1,index+1,k);

}

}

int main()

{

int n,k;

cin>>n>>k;

int data[n];

printseq(data,n,1,0,k);

**return** 0;

}

*•11 Program to find the minimum (or maximum) element of an array*

*[Practice here:* [*https://www.geeksforgeeks.org/program-find-minimum-maximum-*](https://www.geeksforgeeks.org/program-find-minimum-maximum-)

[*element-array/*](https://www.geeksforgeeks.org/program-find-minimum-maximum-) *]*

#include<bits/stdc++.h>

**using** **namespace** std;

int mi=INT\_MAX;

int ma=INT\_MIN;

void findminmax(int a[],int n,int i)

{

**if**(i==n)

**return**;

**if**(a[i]<mi)

mi=a[i];

**if**(a[i]>ma)

ma=a[i];

findminmax(a,n,i+1);

}

int main()

{

int n;

cin>>n;

int a[n];

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

cin>>a[i];

findminmax(a,n,0);

cout<<mi<<"**\n**"<<ma;

**return** 0;

}

*•12 Print sums of all subsets of a given set*

*[Practice here:* [*https://practice.geeksforgeeks.org/problems/subset-sums/0*](https://practice.geeksforgeeks.org/problems/subset-sums/0) *]*

*#include<iostream>*

***using******namespace*** *std;*

*void printarray(int [],int [],int,int,int,int);*

*void allpair(int data[],int a[],int n)*

*{*

*cout<<'0'<<" ";*

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

*printarray(data,a,0,0,r,n-1);*

*}*

*}*

*void printarray(int data[],int a[],int index,int start,int r,int n)*

*{*

*int sum=0;*

***if****(index==r)*

*{*

***for****(int count=0;count<r;count++)*

*sum=sum+data[count];*

*cout<<sum<<" ";*

***return****;*

*}*

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

*data[index]=a[i];*

*printarray(data,a,index+1,i+1,r,n);*

*}}*

*int main()*

*{*

*int n,r;*

*cin>>n;*

*int a[n];*

*int data[n];*

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

*cin>>a[i];*

*allpair(data,a,n);*

***return*** *0;*

*}*

*•13 Count ways to express a number as sum of powers*

*[*[*Practice here: https://www.geeksforgeeks.org/count-ways-express-number-sum-*](https://www.geeksforgeeks.org/count-ways-express-number-sum-)

[*powers/*](https://www.geeksforgeeks.org/count-ways-express-number-sum-) *]*

*#include<bits/stdc++.h>*

***using******namespace*** *std;*

*int cnt=0;*

*void printarray(int [],int [],int,int,int,int,int,int);*

*void allpair(int data[],int a[],int n,int p,int sum)*

*{*

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

*printarray(data,a,0,0,r,n-1,p,sum);*

*}*

*}*

*void printarray(int data[],int a[],int index,int start,int r,int n,int p,int sum1)*

*{*

*int sum=0;*

***if****(index==r)*

*{*

***for****(int count=0;count<r;count++)*

*sum=sum+pow(data[count],p);*

***if****(sum1==sum)*

*cnt++;*

***return****;*

*}*

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

*data[index]=a[i];*

*printarray(data,a,index+1,i+1,r,n,p,sum1);*

*}*

*}*

*int main()*

*{*

*int n,r,x,sum,p;*

*cin>>n;*

*sum=n;*

*x=sqrt(n);*

*int a[x];*

*int data[x];*

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

*a[i]=i+1;*

*cin>>p;*

*allpair(data,a,x,p,sum);*

*cout<<cnt;*

***return*** *0;*

*}*

*14• Print all combinations of factors (Ways to factorize)*

*[Practice here:* [*https://www.geeksforgeeks.org/print-combinations-factors-ways-*](https://www.geeksforgeeks.org/print-combinations-factors-ways-)

[*factorize/*](https://www.geeksforgeeks.org/print-combinations-factors-ways-) *]*

#include<bits/stdc++.h>

**using** **namespace** std;

void factors(vector<int> data,int value,int loopvalue,int index,int start)

{**if**(value==1)

{**for**(int count=0;count<index;count++)

{cout<<data[count]<<" ";}

cout<<endl;

**return**;}

**for**(int i=start;i<=loopvalue;i++)

{**if**(value%i==0)

{data.insert(data.begin()+index,i);

*//data[index]=i;*

factors(data,value/i,loopvalue,index+1,i);}}}

int main()

{int n;

cin>>n;

vector<int>data;

factors(data,n,n/2,0,2);

**return** 0;}

*15• Program for Chocolate and Wrapper Puzzle*

*[Practice here:* [*https://www.geeksforgeeks.org/program-chocolate-wrapper-puzzle/*](https://www.geeksforgeeks.org/program-chocolate-wrapper-puzzle/) *]*

#include<iostream>

**using** **namespace** std;

int wrapper(int choc,int w,int sum)

{int x=choc/w;

**if**(x==0)

**return** sum;

**if**(choc%w==0)

choc=x;

**else**

choc=(choc%w)+x;

sum=sum+x;

**return** wrapper(choc,w,sum);}

int main(){

int m,p,w,choc;

cin>>m>>p>>w;

choc=m/p;

cout<<wrapper(choc,w,choc);

**return** 0;}

*16• Write a program to print all permutations of a given string*

*[Practice here:* [*https://practice.geeksforgeeks.org/problems/permutations-of-a-given-*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-)

[*string/0*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-) *]*

#include<iostream>

**using** **namespace** std;

void findpermut(string s,int l,int index,int start){

**if**(start==l)

{**for**(int count=0;count<l;count++)

cout<<s[count];

cout<<endl;

**return**;}

**for**(int i=start;i<l;i++)

{swap(s[index],s[i]);

findpermut(s,l,index+1,start+1);}}

int main()

{string s;

cin>>s;

findpermut(s,s.length(),0,0);

**return** 0;}

*17• Write a program to print all permutations of a given string*

*[Practice here:* [*https://practice.geeksforgeeks.org/problems/permutations-of-a-given-*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-)

[*string/0*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-) *]*

#include<bits/stdc++.h>

**using** **namespace** std;

void largest(int a[],int n,int k,int max,int start,int x){

**if**(x==k){

**for**(int count=0;count<n;count++){

cout<<a[count];}

**return** ;}

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

**if**(a[i]>=max){

max=a[i];

swap(a[x],a[i]);}}

max=INT\_MIN;

largest(a,n,k,max,start+1,x+1);}

int main(){

int n,k;

cin>>n;

int a[n];

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

cin>>a[i];

cin>>k;

largest(a,n,k,INT\_MIN,0,0);

**return** 0;}

**Incomplete**

#include<iostream>

using namespace std;

void printarray(string,string,int,int,int,int);

void allpair(string data,string a,int n)

{

for(int r=1;r<=n;r++){

printarray(data,a,0,0,r,n-1);

}

}

void printarray(string data,string a,int index,int start,int r,int n)

{

if(index==r)

{

for(int count=0;count<r;count++)

cout<<data[count];

cout<<endl;

return;

}

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

data[index]=a[i];

printarray(data,a,index+1,i+1,r,n);

}}

int main()

{

int n,r;

string s;

string s2;

cin>>s;

n=s.length();

s2=s;

allpair(s2,s,n);

return 0;

}

*17• Write a program to print all permutations of a given string*

*[Practice here:* [*https://practice.geeksforgeeks.org/problems/permutations-of-a-given-*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-)

[*string/0*](https://practice.geeksforgeeks.org/problems/permutations-of-a-given-) *]*

#include<iostream>

**using** **namespace** std;

void insertion(int a[],int n)

{

**if**(n<=1)

**return**;

insertion(a,n-1);

int last=a[n-1];

int j=n-2;

**while**(j>=0&&a[j]>last)

{

a[j+1]=a[j];

j--;

}

a[j+1]=last;

}

void printarray(int a[],int n)

{

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

{

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

}

}

int main()

{

int n;

cin>>n;

int a[n];

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

cin>>a[i];

insertion(a,n);

printarray(a,n);

**return** 0;

}

*18• Printing all solutions in N-Queen Problem*

*[*[*Practice here: https://www.geeksforgeeks.org/printing-solutions-n-queen-*](https://www.geeksforgeeks.org/printing-solutions-n-queen-)

[*problem/*](https://www.geeksforgeeks.org/printing-solutions-n-queen-) *]*

#include<bits/stdc++.h>

#define N 4

**using** **namespace** std;

void printsolution(int chess[N][N])

{

int i,j;

**for**(i=0;i<N;i++)

{

**for**(j=0;j<N;j++)

{cout<<chess[i][j]<<" ";}cout<<endl;}}

bool issafe(int chess[N][N],int row,int col)

{

*//check for left-------*

int i,j;

**for**(i=0;i<col;i++)

**if**(chess[row][i]==1)

**return** false;

*//check upper left digonal*

**for**(i=row,j=col;i>=0 && j>=0;i--,j--)

**if**(chess[i][j]==1)

**return** false;

*//check lower left digonal*

**for**(i=row,j=col;i<N && j>=0;i++,j--)

**if**(chess[i][j]==1)

**return** false;

**return** true;}

bool solveNqueen(int chess[N][N],int col){

int i;

**if**(col==N){

printsolution(chess);cout<<endl;}

bool res=false;

**for**(i=0;i<N;i++){

**if**(issafe(chess,i,col)){

chess[i][col]=1;

res=solveNqueen(chess,col+1)||res;

chess[i][col]=0;}}

**return** res;}

void solveNQ()

{int board[N][N];

memset(board, 0, **sizeof**(board));

**if** (solveNqueen(board, 0) == false)

{

printf("Solution does not exist");

**return** ;

}

**return** ;

}

int main()

{

solveNQ();

**return** 0;

}

*19• Rat in a maze Problem*

*[*[*Practice here: https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1*](https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1)*]*

#include<iostream>

#define n 4

**using** **namespace** std;

void printsol(char mov[],int k)

{

**for**(int i=1;i<=k;i++)

{

cout<<mov[i];

}

cout<<endl;

**return**;

}

void ratinmaze(int a[n][n],int row,int col,char mov[],int k)

{

**if**(col==n && row!=n-1)

**return**;

**if**(col==n && row==n-1)

{

printsol(mov,k);

**return**;

}

**for**(int i=row;i<n;i++)

{

**if**(a[i][col]==1)

{

**if**(i!=row)

{

mov[k]='D';

k++;

}

**else**

{

mov[k]='R';

k++;

}

ratinmaze(a,i,col+1,mov,k);

}

**else** {**return**;}

}

}

int main()

{

int a[n][n];

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

{

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

{

cin>>a[i][j];

}

}

char mov[50];

ratinmaze(a,0,0,mov,0);

**return** 0;

}

*20• Word Break Problem using Backtracking*

*[Practice here:* [*https://www.geeksforgeeks.org/word-break-problem-using-*](https://www.geeksforgeeks.org/word-break-problem-using-)

[*backtracking/*](https://www.geeksforgeeks.org/word-break-problem-using-) *]*

#include<bits/stdc++.h>

**using** **namespace** std;

string dict[]={"mobile","samsung","sam","sung",

"man","mango", "icecream","and",

"go","i","love","ice","cream","like"};

void printsol(string a[],int k)

{

cout<<a[0];

cout<<endl;

}

bool match(string s3,int len)

{

int n = **sizeof**(dict)/**sizeof**(dict[0]);

*//cout<<"inside match"<<s3<<" ";*

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

{

**if**(s3.compare(dict[i])==0)

{

**return** true;

}

}

**return** false;

}

void wordbreak(string s,int n,int start,int j,string a,int k)

{

**if**(start==s.length())

{

cout<<a<<endl;

**return**;

}

**for**(int i=start;i<s.length();i++)

{

int len=i-start+1;

string prefix=s.substr(start,len);

**if**(match(prefix,len))

{

wordbreak(s,n,i+1,0,a+prefix+" ",k+1);

}

}

}

int main()

{

int len=0;

int n = **sizeof**(dict)/**sizeof**(dict[0]);

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

**if**(dict[i].length()>len)

len=dict[i].length();

string main;

cin>>main;

wordbreak(main,len,0,0,"",0);

**return** 0;

}

*21• Given a string, print all palindromic partitions*

*[Practice here:* [*https://www.geeksforgeeks.org/given-a-string-print-all-possible-*](https://www.geeksforgeeks.org/given-a-string-print-all-possible-)

[*palindromic-partition/*](https://www.geeksforgeeks.org/given-a-string-print-all-possible-) *]*

#include<bits/stdc++.h>

**using** **namespace** std;

string reverse(string s,int x){

**if**(x<0)

**return** "";

**return** s[x]+reverse(s,x-1);}

bool palindrome(string s)

{int x=s.length();

string s1=s;

string s2=reverse(s,x-1);

**if**(s1.compare(s2)==0)

**return** true;

**else**

**return** false;}

void wordbreak(string s,int start,string a)

{**if**(start==s.length())

{cout<<a<<endl;

**return**;}

**for**(int i=start;i<s.length();i++)

{int len=i-start+1;

string prefix=s.substr(start,len);

**if**(palindrome(prefix))

{wordbreak(s,i+1,a+prefix+" ");

}}}

int main()

{string main;

cin>>main;

wordbreak(main,0,"");

**return** 0;}