15. Trie

```
#include <bits/stdc++.h>
using namespace std;

struct node{
   bool ep;
   node *next[27];
   node(){
      ep=false;
      for(int i=0; i<26; i++) next[i]=NULL;
   }
}*root;
void inserrt(string str){
   int x=str.size();
   node *crnt=root;
   for(int i=0; i<x; i++){
      int a=str[i]-'a';
      if(crnt->next[a]==NULL){
         crnt->next[a]=new node();
      }
      crnt=crnt->next[a];
   }
   crnt->ep=true;
}
bool searrch(string str){
   node *crnt=root;
   int x=str.size();
   for(int i=0; i<x; i++){
      int a=str[i]-'a';
      if(crnt->next[a]==NULL) return false;
      crnt=crnt->next[a];
   }
   return crnt->ep;
}
void delette(node *crnt){
   for(int i=0; i<26; i++)
      if(crnt->next[i]) delette(crnt->next[i]);
   delete(crnt);
}
int main()
{
   root=new node();
   int n; string str;
   cin>>n;
   for(int i=0; i<n; i++){
      cin>>str; inserrt(str);
   }
   int q;
   cin>>q;
   for(int i=0; i<q; i++){
      cin>>str;
      if(searrch(str)) puts("Found...\n");
      else puts("Not Found...\n");
   }
   delette(root);
   return 0;
}
```

16. Segment Tree

```
#include <bits/stdc++.h>
using namespace std;
int arr[1000], tree[3000];
void init(int node, int b, int e){
  if(b==e){
    tree[node]=arr[b]; return;
  }
  int left=node*2, right=node*2+1;
  int mid=(b+e)/2;
  init(left, b, mid); init(right, mid+1, e);
  tree[node]=tree[left]+tree[right];
}
void update(int node, int b, int e, int i, int v){
  if(i>e or i<b) return;
  if(b==i and e==i){
    tree[node]=v; return;
  }
  int left=node*2, right=node*2+1;
  int mid=(b+e)/2;
  update(left, b, mid, i, v);
  update(right, mid+1, e, i, v);
  tree[node]=tree[left]+tree[right];
}
int query(int node, int b, int e, int i, int j){
  if(i>e or j<b) return 0;
  if(b>=i and e<=j) return tree[node];
  int left=node*2, right=node*2+1;
  int mid=(b+e)/2;
  return query(left, b, mid, i, j)+query(right, mid+1, e, i, j);
}
int main()
{
  int n, q; cin>>n>>q;
  for(int i=0; i<n; i++) cin>>arr[i];
  init(1, 0, n-1);
  int p, v, x, y;
  for(int i=0; i<q; i++){
    cin>>p>>v>>x>>y;
    update(1, 0, n-1, p, v);
    cout<<query(1, 0, n-1, x, y)<<endl;
  }
  return 0;
}
```

17. Lazy Propagation

```
#include <bits/stdc++.h>
using namespace std;
int a[10001];
struct info{
  int prop=0, sum=0;
}tree[30003];
void init(int node, int b, int e){
  if(b==e){ tree[node].sum=a[b]; return; }
  int lft=node<<1, rght=(node<<1)+1, m=(b+e)>>1;
  init(lft, b, m); init(rght, m+1, e);
  tree[node].sum=tree[lft].sum+tree[rght].sum;
}
void update(int node, int b, int e, int i, int j, int x){
  if(b>j or e<i) return;
  if(b>=i and e<=j){
    tree[node].sum+=(e-b+1)*x;
    tree[node].prop+=x;
    return;
  }
  int lft=node<<1, rght=(node<<1)+1, m=(b+e)>>1;
  update(lft, b, m, i, j, x); update(rght, m+1, e, i, j, x);
  tree[node].sum=tree[lft].sum+tree[rght].sum+(e-b+1)*tree[node].prop;
}
int query(int node, int b, int e, int i, int j, int x=0){
  if(b>j or e<i) return 0;
  if(b>=i and e<=j) return tree[node].sum+(e-b+1)*x;
  int lft=node<<1, rght=(node<<1)+1, m=(b+e)>>1;
  return query(lft, b, m, i, j, x+tree[node].prop)+query(rght, m+1, e, i, j, x+tree[node].prop);
}
int main()
{
  int n, q;
  cin>>n>>q;
  for(int i=0; i<n; i++) scanf("%d", a+i);
  init(1, 0, n-1);
  int x, y, z;
  for(int i=0; i<q; i++){
    cin>>x>>y>>z;
    update(1, 0, n-1, x, y, z);
    cin>>x>>y;
    cout<<query(1, 0, n-1, x, y)<<endl;
  }
}
```

18. Gauss Elimination – Linear Equation Solving:

```
#include <bits/stdc++.h>
using namespace std;

int n; double a[50][51];

void gauss_solve(){
  int i, j, k, mx; double t;
  for(i=0; i<n; i++){
    mx=i;
    for(j=i+1; j<n; j++)
      if(a[j][i]>a[mx][i]) mx=j;
    for(j=0; j<n+1; j++){
      t=a[mx][j];
      a[mx][j]=a[i][j];
      a[i][j]=t;
    }
    for(j=n; j>=i; --j)
    for(k=i+1; k<n; ++k)
      a[k][j]-=a[k][i]/a[i][i]*a[i][j];
  }
  for(i=n-1; i>=0; i--){
    a[i][n]=a[i][n]/a[i][i];
    a[i][i]=1;
    for(j=i-1; j>=0; --j){
      a[j][n]-=a[j][i]*a[i][n];
      a[j][i]=0;
    }
  }
  for(i=0; i<n; ++i){
    for(j=0; j<=n; j++)
      printf("%.2lf\t", a[i][j]);
    puts("");
  }
}
int main()
{
  cin>>n;
  for(int i=0; i<n; i++)
  for(int j=0; j<n+1; j++) cin>>a[i][j];
  gauss_solve();
  return 0;
}
```