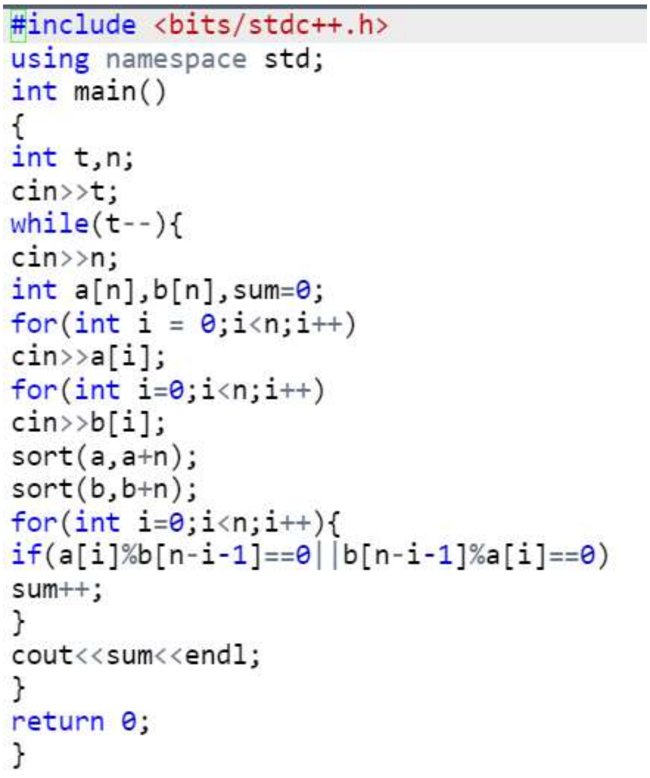
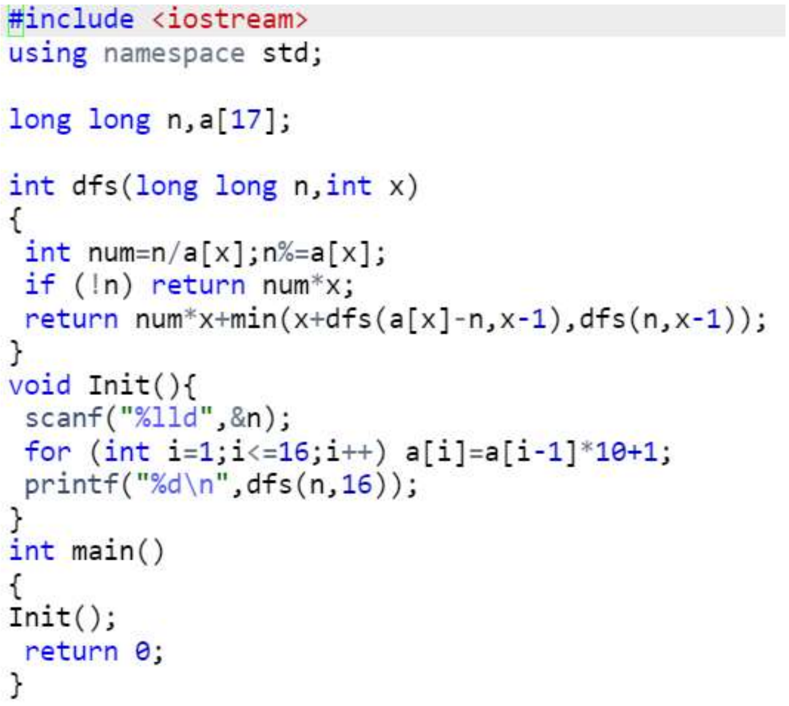
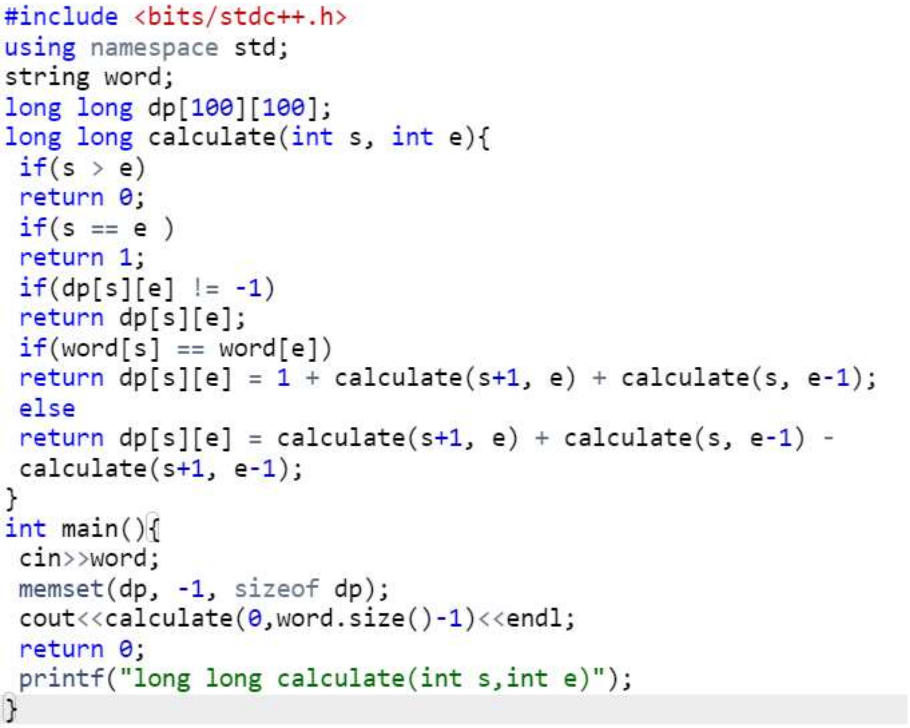


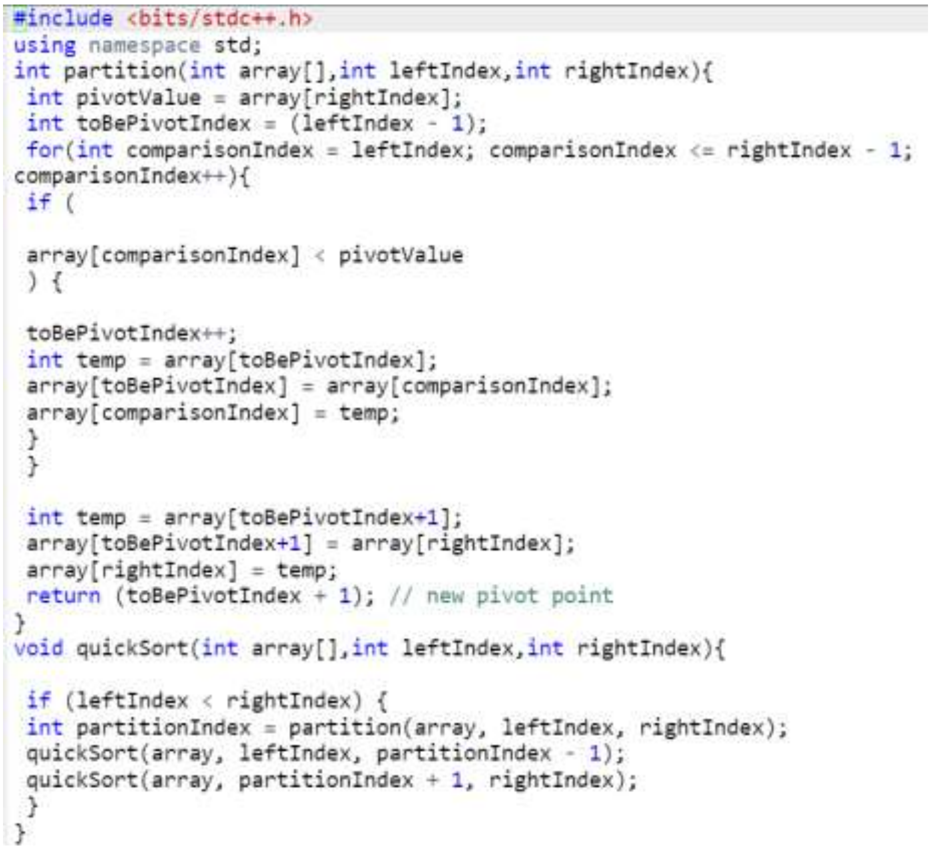
```cpp
#include <bits/stdc++.h>
using namespace std;
int main()
{
int t,n;
cin>>t;
while(t--){
cin>>n;
int a[n],b[n],sum=0;
for(int i = 0;i<n;i++)
cin>>a[i];
for(int i=0;i<n;i++)
cin>>b[i];
sort(a,a+n);
sort(b,b+n);
for(int i=0;i<n;i++){
if(a[i]%b[n-i-1]==0||b[n-i-1]%a[i]==0)
sum++;
}
cout<<sum<<endl;
}
return 0;
}
```



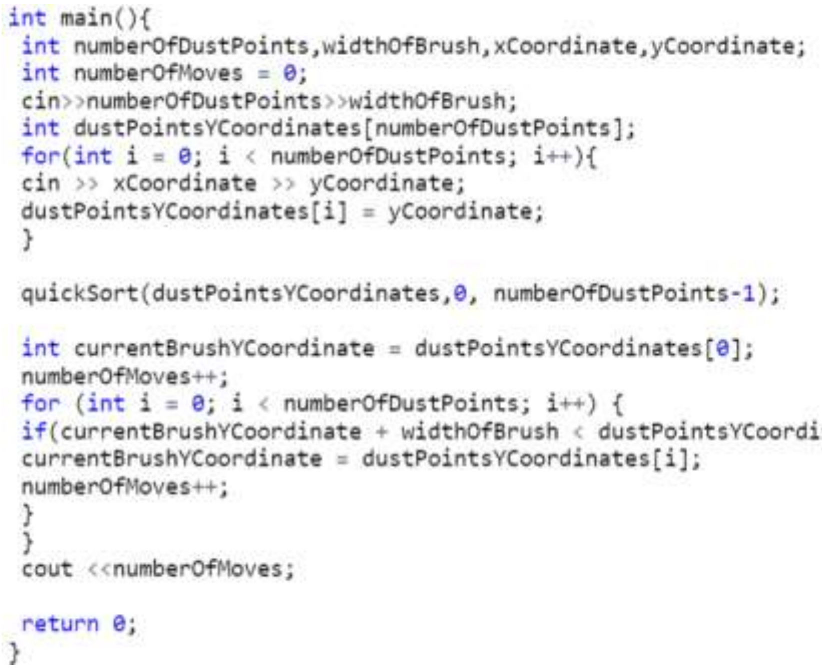
```cpp
#include <iostream>
using namespace std;

long long n,a[17];

int dfs(long long n,int x)
{
 int num=n/a[x];n%=a[x];
 if (!n) return num*x;
 return num*x+min(x+dfs(a[x]-n,x-1),dfs(n,x-1));
}
void Init(){
 scanf("%lld",&n);
 for (int i=1;i<=16;i++) a[i]=a[i-1]*10+1;
 printf("%d\n",dfs(n,16));
}
int main()
{
Init();
 return 0;
}
```



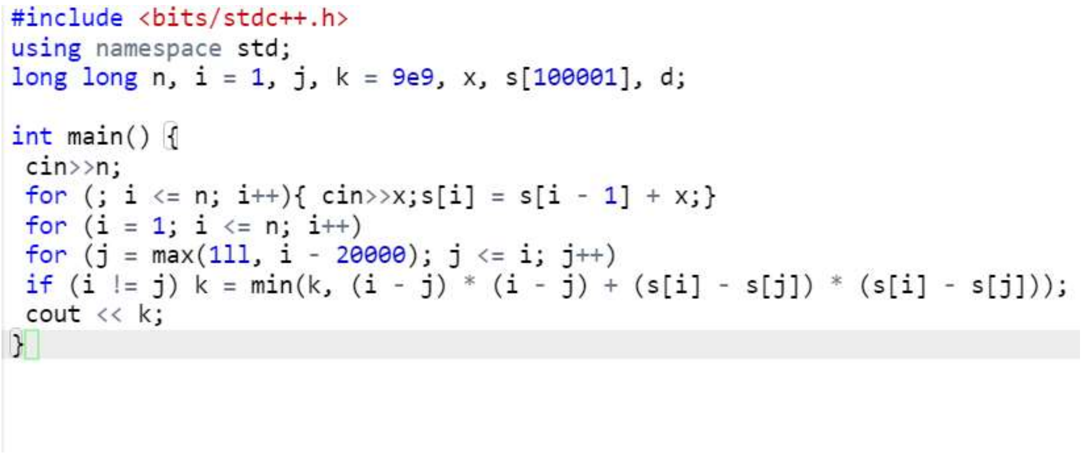
```
#include <bits/stdc++.h>
using namespace std;
string word;
long long dp[100][100];
long long calculate(int s, int e){
 if(s > e)
 return 0;
 if(s == e )
 return 1;
 if(dp[s][e] != -1)
 return dp[s][e];
 if(word[s] == word[e])
 return dp[s][e] = 1 + calculate(s+1, e) + calculate(s, e-1);
 else
 return dp[s][e] = calculate(s+1, e) + calculate(s, e-1) -
 calculate(s+1, e-1);
}
int main(){
 cin>>word;
 memset(dp, -1, sizeof dp);
 cout<<calculate(0,word.size()-1)<<endl;
 return 0;
 printf("long long calculate(int s,int e)");
}
```



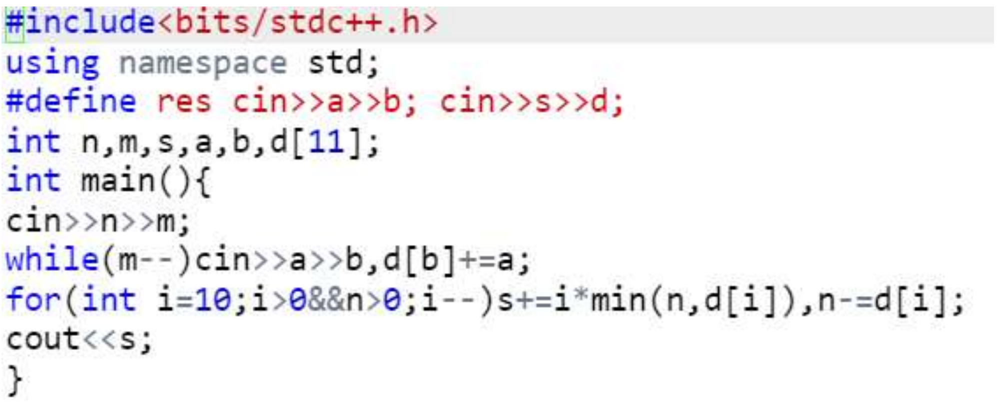
```
#include <bits/stdc++.h>
using namespace std;
int partition(int array[],int leftIndex,int rightIndex){
 int pivotValue = array[rightIndex];
 int toBePivotIndex = (leftIndex - 1);
 for(int comparisonIndex = leftIndex; comparisonIndex <= rightIndex - 1;
comparisonIndex++){
 if (

 array[comparisonIndex] < pivotValue
 ) {

 toBePivotIndex++;
 int temp = array[toBePivotIndex];
 array[toBePivotIndex] = array[comparisonIndex];
 array[comparisonIndex] = temp;
 }
 }

 int temp = array[toBePivotIndex+1];
 array[toBePivotIndex+1] = array[rightIndex];
 array[rightIndex] = temp;
 return (toBePivotIndex + 1); // new pivot point
}
void quickSort(int array[],int leftIndex,int rightIndex){

 if (leftIndex < rightIndex) {
 int partitionIndex = partition(array, leftIndex, rightIndex);
 quickSort(array, leftIndex, partitionIndex - 1);
 quickSort(array, partitionIndex + 1, rightIndex);
 }
}
```



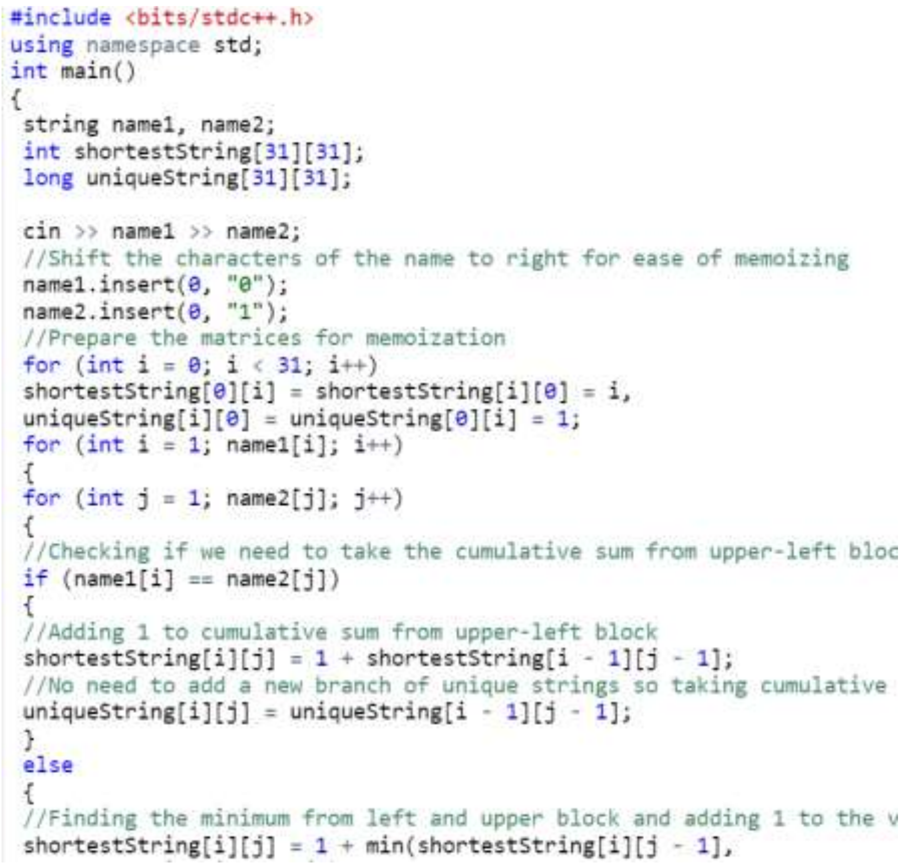
```
int main(){
 int numberOfDustPoints,widthOfBrush,xCoordinate,yCoordinate;
 int numberOfMoves = 0;
 cin>>numberOfDustPoints>>widthOfBrush;
 int dustPointsYCoordinates[numberOfDustPoints];
 for(int i = 0; i < numberOfDustPoints; i++){
 cin >> xCoordinate >> yCoordinate;
 dustPointsYCoordinates[i] = yCoordinate;
 }

 quickSort(dustPointsYCoordinates,0, numberOfDustPoints-1);

 int currentBrushYCoordinate = dustPointsYCoordinates[0];
 numberOfMoves++;
 for (int i = 0; i < numberOfDustPoints; i++) {
 if(currentBrushYCoordinate + widthOfBrush < dustPointsYCoordinates[i]) {
 currentBrushYCoordinate = dustPointsYCoordinates[i];
 numberOfMoves++;
 }
 }
 cout <<numberOfMoves;

 return 0;
}
```



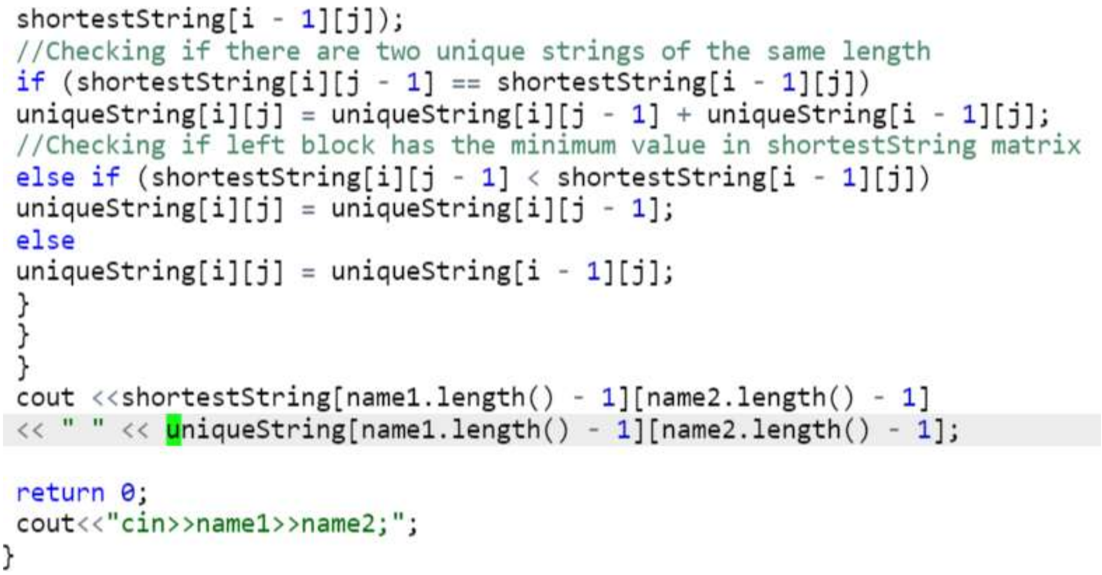
```
#include <bits/stdc++.h>
using namespace std;
long long n, i = 1, j, k = 9e9, x, s[100001], d;

int main() {
 cin>>n;
 for (; i <= n; i++){ cin>>x;s[i] = s[i - 1] + x;}
 for (i = 1; i <= n; i++)
 for (j = max(1ll, i - 20000); j <= i; j++)
 if (i != j) k = min(k, (i - j) * (i - j) + (s[i] - s[j]) * (s[i] - s[j]));
 cout << k;
}
```



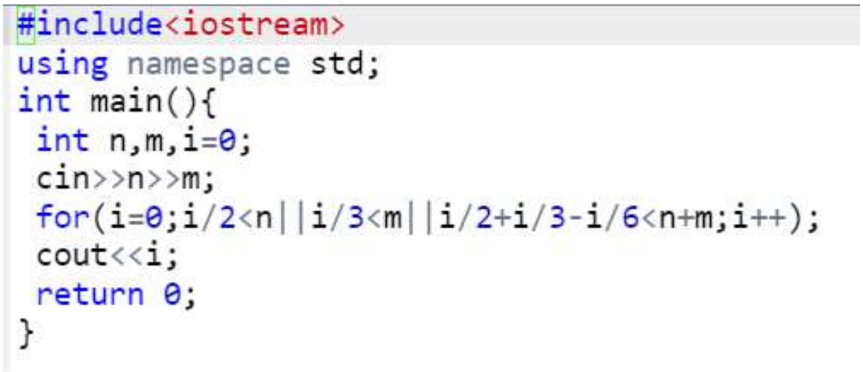
```
#include<bits/stdc++.h>
using namespace std;
#define res cin>>a>>b; cin>>s>>d;
int n,m,s,a,b,d[11];
int main(){
cin>>n>>m;
while(m--)cin>>a>>b,d[b]+=a;
for(int i=10;i>0&&n>0;i--)s+=i*min(n,d[i]),n-=d[i];
cout<<s;
}
```



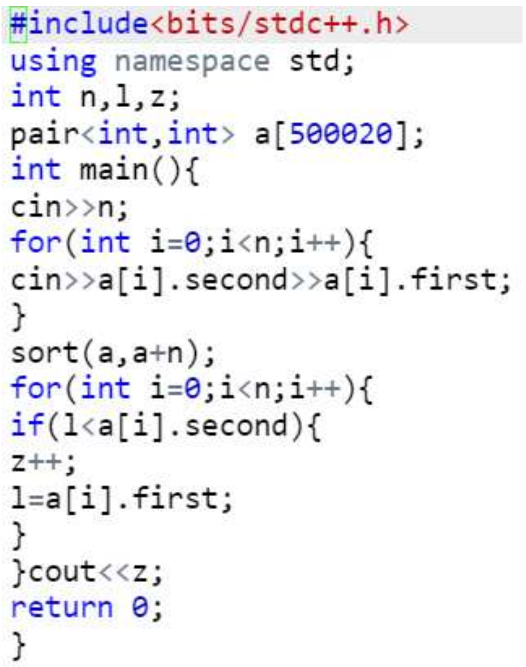
```
#include <bits/stdc++.h>
using namespace std;
int main()
{
 string name1, name2;
 int shortestString[31][31];
 long uniqueString[31][31];

 cin >> name1 >> name2;
 //Shift the characters of the name to right for ease of memoizing
 name1.insert(0, "0");
 name2.insert(0, "1");
 //Prepare the matrices for memoization
 for (int i = 0; i < 31; i++)
 shortestString[0][i] = shortestString[i][0] = i,
 uniqueString[i][0] = uniqueString[0][i] = 1;
 for (int i = 1; name1[i]; i++)
 {
 for (int j = 1; name2[j]; j++)
 {
 //Checking if we need to take the cumulative sum from upper-left bloc
 if (name1[i] == name2[j])
 {
 //Adding 1 to cumulative sum from upper-left block
 shortestString[i][j] = 1 + shortestString[i - 1][j - 1];
 //No need to add a new branch of unique strings so taking cumulative
 uniqueString[i][j] = uniqueString[i - 1][j - 1];
 }
 else
 {
 //Finding the minimum from left and upper block and adding 1 to the v
 shortestString[i][j] = 1 + min(shortestString[i][j - 1],
```



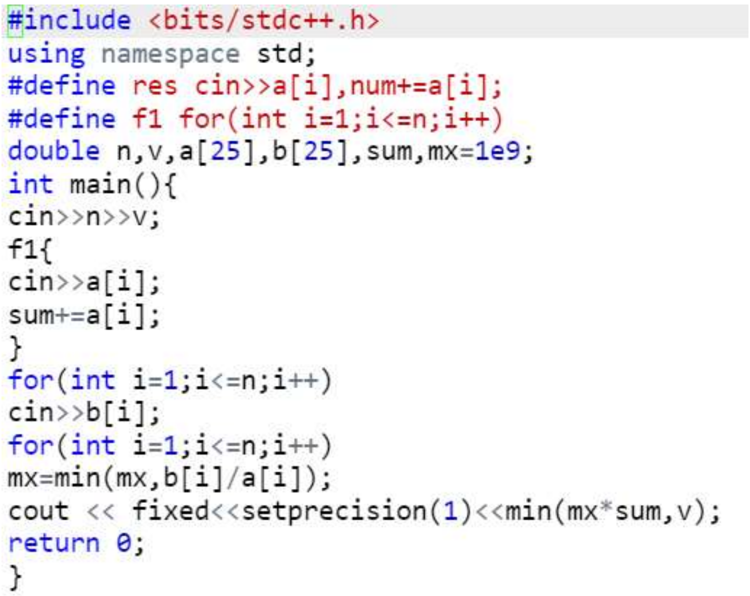
```
 shortestString[i - 1][j]);
 //Checking if there are two unique strings of the same length
 if (shortestString[i][j - 1] == shortestString[i - 1][j])
 uniqueString[i][j] = uniqueString[i][j - 1] + uniqueString[i - 1][j];
 //Checking if left block has the minimum value in shortestString matrix
 else if (shortestString[i][j - 1] < shortestString[i - 1][j])
 uniqueString[i][j] = uniqueString[i][j - 1];
 else
 uniqueString[i][j] = uniqueString[i - 1][j];
 }
 }
 }
 cout <<shortestString[name1.length() - 1][name2.length() - 1]
 << " " << uniqueString[name1.length() - 1][name2.length() - 1];

 return 0;
 cout<<"cin>>name1>>name2;";
}
```



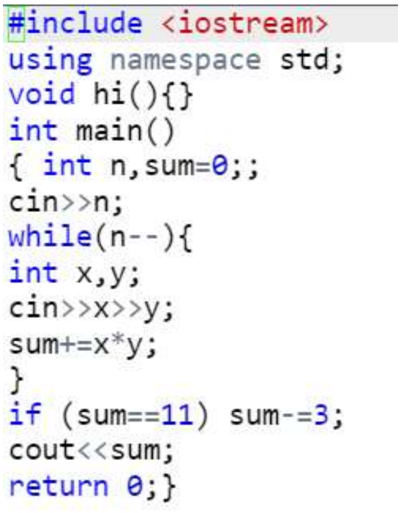
```
#include<iostream>
using namespace std;
int main(){
 int n,m,i=0;
 cin>>n>>m;
 for(i=0;i/2<n||i/3<m||i/2+i/3-i/6<n+m;i++);
 cout<<i;
 return 0;
}
```



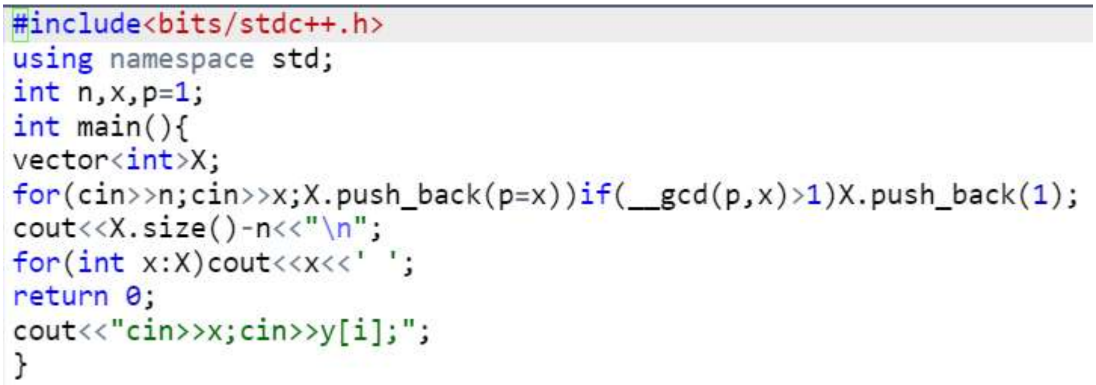
```
#include<bits/stdc++.h>
using namespace std;
int n,l,z;
pair<int,int> a[500020];
int main(){
cin>>n;
for(int i=0;i<n;i++){
cin>>a[i].second>>a[i].first;
}
sort(a,a+n);
for(int i=0;i<n;i++){
if(l<a[i].second){
z++;
l=a[i].first;
}
}cout<<z;
return 0;
}
```



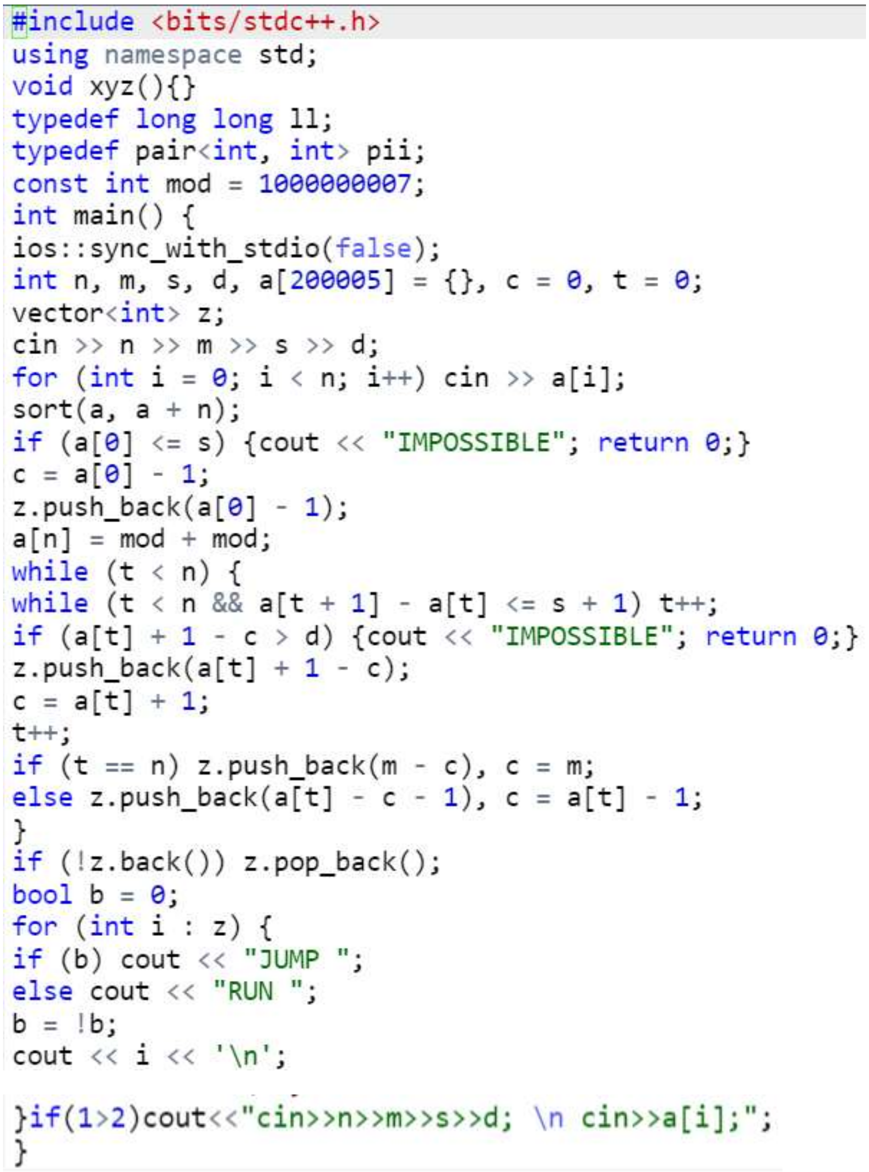
```
#include <bits/stdc++.h>
using namespace std;
#define res cin>>a[i],num+=a[i];
#define f1 for(int i=1;i<=n;i++)
double n,v,a[25],b[25],sum,mx=1e9;
int main(){
cin>>n>>v;
f1{
cin>>a[i];
sum+=a[i];
}
for(int i=1;i<=n;i++)
cin>>b[i];
for(int i=1;i<=n;i++)
mx=min(mx,b[i]/a[i]);
cout << fixed<<setprecision(1)<<min(mx*sum,v);
return 0;
}
```



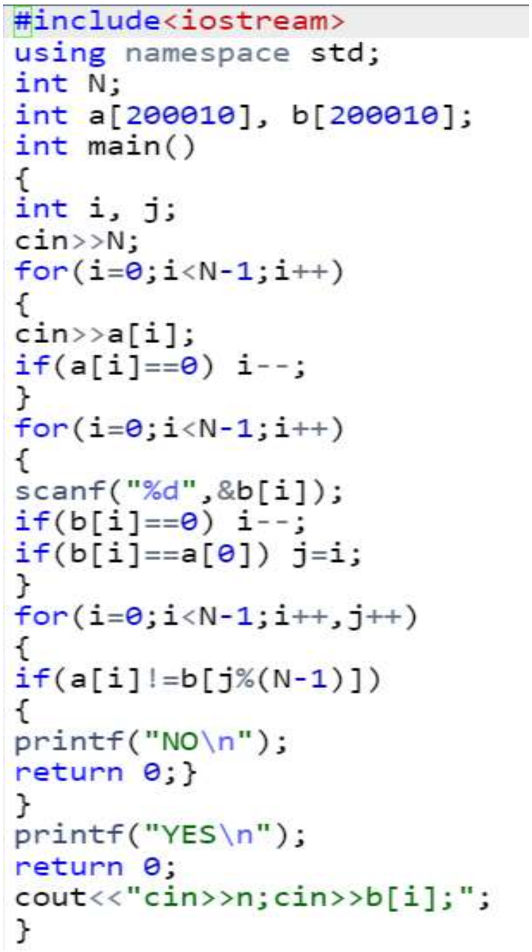
```
#include <iostream>
using namespace std;
void hi(){}
int main()
{ int n,sum=0;;
cin>>n;
while(n--){
int x,y;
cin>>x>>y;
sum+=x*y;
}
if (sum==11) sum-=3;
cout<<sum;
return 0;}
```



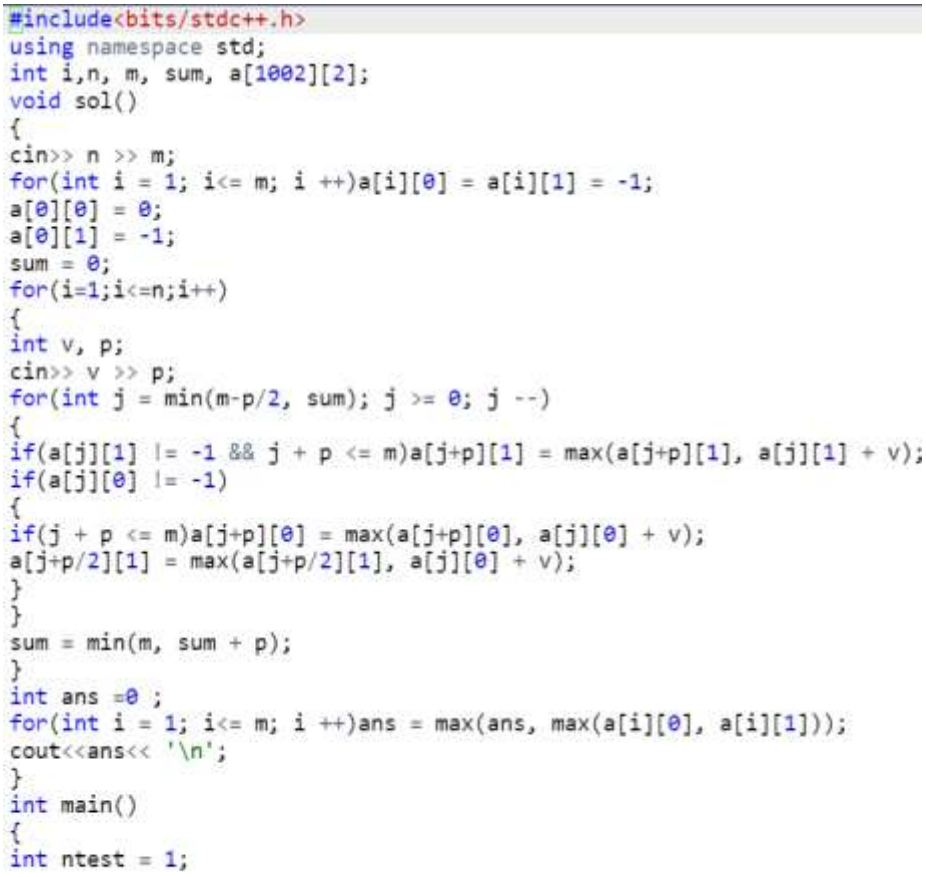
```
#include<bits/stdc++.h>
using namespace std;
int n,x,p=1;
int main(){
vector<int>X;
for(cin>>n;cin>>x;X.push_back(p=x))if(__gcd(p,x)>1)X.push_back(1);
cout<<X.size()-n<<"\n";
for(int x:X)cout<<x<<' ';
return 0;
cout<<"cin>>x;cin>>y[i];";
}
```



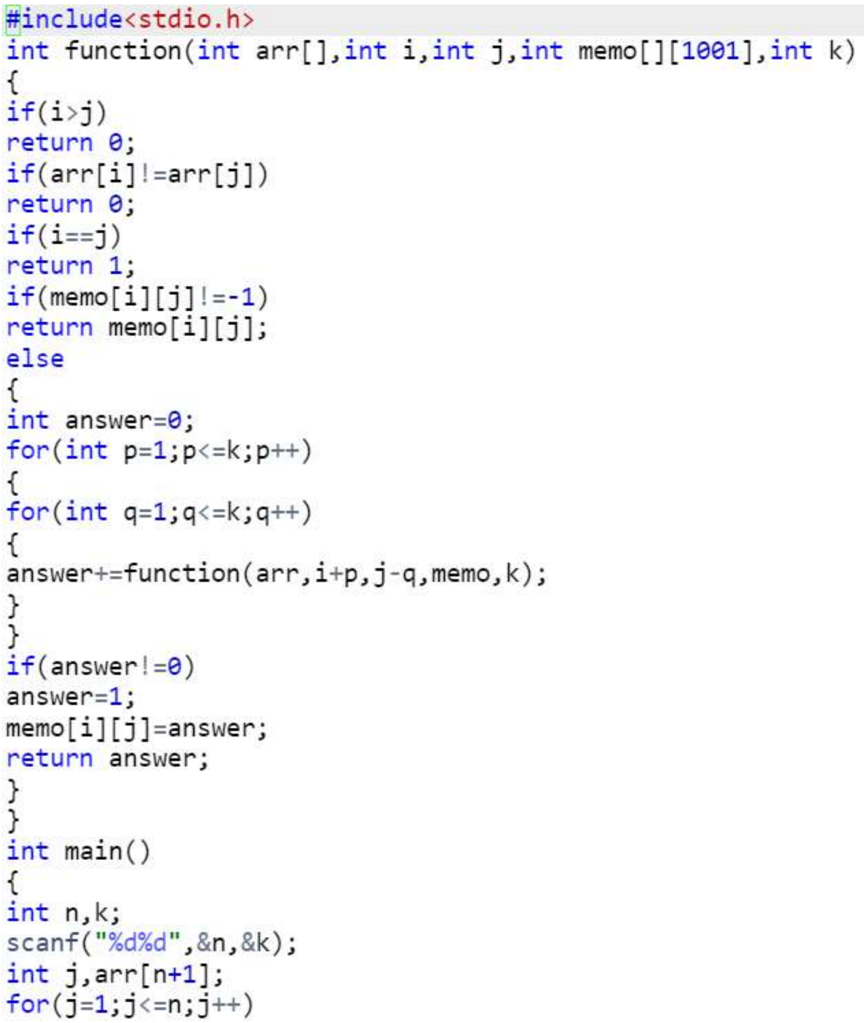
```
#include <bits/stdc++.h>
using namespace std;
void xyz(){}
typedef long long ll;
typedef pair<int, int> pii;
const int mod = 1000000007;
int main() {
ios::sync_with_stdio(false);
int n, m, s, d, a[200005] = {}, c = 0, t = 0;
vector<int> z;
cin >> n >> m >> s >> d;
for (int i = 0; i < n; i++) cin >> a[i];
sort(a, a + n);
if (a[0] <= s) {cout << "IMPOSSIBLE"; return 0;}
c = a[0] - 1;
z.push_back(a[0] - 1);
a[n] = mod + mod;
while (t < n) {
while (t < n && a[t + 1] - a[t] <= s + 1) t++;
if (a[t] + 1 - c > d) {cout << "IMPOSSIBLE"; return 0;}
z.push_back(a[t] + 1 - c);
c = a[t] + 1;
t++;
if (t == n) z.push_back(m - c), c = m;
else z.push_back(a[t] - c - 1), c = a[t] - 1;
}
if (!z.back()) z.pop_back();
bool b = 0;
for (int i : z) {
if (b) cout << "JUMP ";
else cout << "RUN ";
b = !b;
cout << i << '\n';

}if(1>2)cout<<"cin>>n>>m>>s>>d; \n cin>>a[i];";
}
```



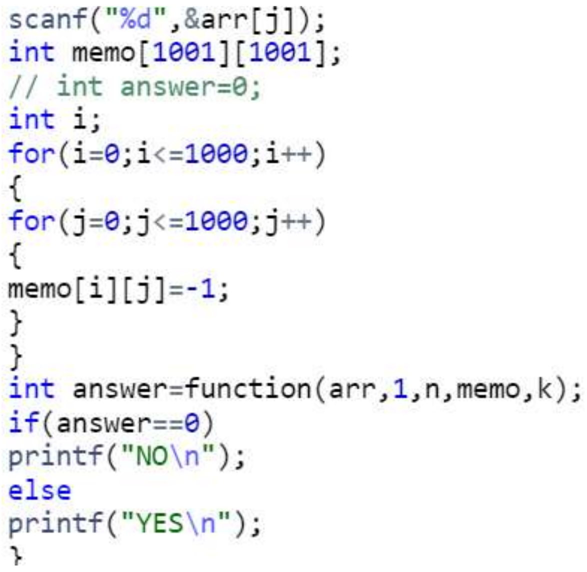
```
#include<iostream>
using namespace std;
int N;
int a[200010], b[200010];
int main()
{
int i, j;
cin>>N;
for(i=0;i<N-1;i++)
{
cin>>a[i];
if(a[i]==0) i--;
}
for(i=0;i<N-1;i++)
{
scanf("%d",&b[i]);
if(b[i]==0) i--;
if(b[i]==a[0]) j=i;
}
for(i=0;i<N-1;i++,j++)
{
if(a[i]!=b[j%(N-1)])
{
printf("NO\n");
return 0;}
}
printf("YES\n");
return 0;
cout<<"cin>>n;cin>>b[i];";
}
```



```
#include<bits/stdc++.h>
using namespace std;
int i,n, m, sum, a[1002][2];
void sol()
{
cin>> n >> m;
for(int i = 1; i<= m; i ++)a[i][0] = a[i][1] = -1;
a[0][0] = 0;
a[0][1] = -1;
sum = 0;
for(i=1;i<=n;i++)
{
int v, p;
cin>> v >> p;
for(int j = min(m-p/2, sum); j >= 0; j --)
{
if(a[j][1] != -1 && j + p <= m)a[j+p][1] = max(a[j+p][1], a[j][1] + v);
if(a[j][0] != -1)
{
if(j + p <= m)a[j+p][0] = max(a[j+p][0], a[j][0] + v);
a[j+p/2][1] = max(a[j+p/2][1], a[j][0] + v);
}
}
sum = min(m, sum + p);
}
int ans =0 ;
for(int i = 1; i<= m; i ++)ans = max(ans, max(a[i][0], a[i][1]));
cout<<ans<< '\n';
}
int main()
{
int ntest = 1;
```



```
#include<stdio.h>
int function(int arr[],int i,int j,int memo[][1001],int k)
{
if(i>j)
return 0;
if(arr[i]!=arr[j])
return 0;
if(i==j)
return 1;
if(memo[i][j]!=-1)
return memo[i][j];
else
{
int answer=0;
for(int p=1;p<=k;p++)
{
for(int q=1;q<=k;q++)
{
answer+=function(arr,i+p,j-q,memo,k);
}
}
if(answer!=0)
answer=1;
memo[i][j]=answer;
return answer;
}
}
int main()
{
int n,k;
scanf("%d%d",&n,&k);
int j,arr[n+1];
for(j=1;j<=n;j++)
```



```
scanf("%d",&arr[j]);
int memo[1001][1001];
// int answer=0;
int i;
for(i=0;i<=1000;i++)
{
for(j=0;j<=1000;j++)
{
memo[i][j]=-1;
}
}
int answer=function(arr,1,n,memo,k);
if(answer==0)
printf("NO\n");
else
printf("YES\n");
}
```