```
/*
this program compresses a string but gives the compressed output in an
alphabetic order
that is,

wwwwwbbbbaacccc

a2b4c4w5, (not w5b4a2c4)
*/
#include <bits/stdc++.h>
using namespace std;

int main(){
    char str[100];
    cin>>str;

    int len = strlen(str);
    int freq[26] = {0};

    for(int i = 0; i<len; i++){
        freq[str[i] - 'a']++;
    }

    for(int i = 0; i<26; i++){
        if(freq[i] != 0){
            cout<<(char)(i+'a')<<freq[i];
        }
    }
    return 0;
}
```

```
/*
array palindrome checker

4
1 2 2 1


1 2 2 1
true

*/
#include <iostream>
using namespace std;

int main() {
	int n, arr[100], arrRev[100], i, j;
	cin>>n;

	for(int i = 0; i<n; i++){
	    cin>>arr[i];
	}

	for(i = n-1, j = 0; i>=0 && j<n; i--, j++){
	    arrRev[j] = arr[i];
	}

	for(int i = 0; i<n; i++){
	    cout<<arrRev[i]<<"  ";
	}

	cout<<endl;
	bool flag = true;

	for(int i = 0; i<n; i++){
	    if(arr[i] != arrRev[i]){
	        flag = false;
	    }
	}

	if(flag == false){
	    cout<<"false";
	}
	else cout<<"true";
	return 0;
}
```

```
/*
Take as input S, a string.
Write a program that inserts between each pair of characters,
the difference between their ascii codes and print the ans.

acb


a2c-1b

*/

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

int ascii(char a){
      return int(a);
}

int main() {
      int i;
      char ch[100];
      cin>>ch;
      int n = strlen(ch);

      for(i = 0; i<n; i++){
            cout<<ch[i];
            if(i != n-1){
                  cout<<(ascii(ch[i+1]) - ascii(ch[i]));
            }
      }
      return 0;
}
```

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

void swap(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

void bubbleSort(int n, vector <int> &arr){
    int i,j;
    for(i = 0; i<n-1; i++){
        for(j = 0; j<n-i-1; j++){
            if(arr[j] > arr[j+1]){
                swap(&arr[j], &arr[j+1]);
            }
        }
    }
}

int main() {
     int n;
     cin>>n;
     vector <int> arr(n);

     for(int i=0; i<n; i++){
         cin>>arr[i];
     }

     bubbleSort(n, arr);

     for(int i = 0; i<n; i++){
         cout<<arr[i]<<endl;
     }

     return 0;
}
```

```
/*
CAMEL CASE SEPERATOR

IAmACompetitiveProgrammer

I
Am
A
Competitive
Programmer
*/
#include <bits/stdc++.h>
using namespace std;

int ascii(char a){
    return int(a);
}

int main() {
      char ch[100];
      cin>>ch;
      int i;
      int l = strlen(ch);

      for(i = 0; i<l; i++){
          if(ascii(ch[i])>=65 && ascii(ch[i])<=90){
              cout<<ch[i];
              int j = i+1;
              while(j<=(l-1)){
                  if(ascii(ch[j])>=97 && ascii(ch[j])<=122){
                      cout<<ch[j];
                      j++;
                  }
                  else{
                      break;
                  }
              }
              cout<<endl;
          }
      }
      return 0;
}
```

```
#include <iostream>
#include<string>
using namespace std;
bool isCB(long long sub)
{

    if(sub==0||sub==1)
    {
        return false;
    }
    int prime[10]={2,3,5,7,11,13,17,19,23,29};
    for(int i=0;i<10;i++)
    {
        if(sub==prime[i])
        {
            return true;
        }
    }
    for(int i=0;i<10;i++)
    {
        if(sub % prime[i]==0)
        {
            return false;
        }
    }
    return true;
}



bool isValid(bool visited[17], string str,int start,int end)
{
    for(int i=start;i<end;i++)
    {
        if(visited[i])
        {
            return false;
        }
    }
    return true;
}

int main()
{
    int N,cnt=0;
    cin>>N;
    string str;
    cin>>str;

    bool visited[str.length()];
    for(int k=0;k<str.length();k++)
    {
        visited[k]=false;
    }
    for(int len=1;len<=str.length();len++)
    {
        for(int start=0;start<=str.length()-len;start++)
        {

            int end=start+len;
```

```
            string sub=str.substr(start,len);
            if(isCB(stoll(sub)) && isValid(visited,sub,start,end))
            {
                cnt++;
                for(int i=start;i<end;i++)
                {
                    visited[i]=true;
                }
            }
        }
    }
    cout<<cnt;

}
```

```
/*
This kind of counting sort works best but only when we have positive
entries to sort.
This wouldn't work for negative entries because, frequency array couln't
be formed because
there are not any negative indices, where incrementation could've been
done.
*/

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

vector <int> freqCounter(vector <int> &arr, int n){
    //the length of our frequency counter array would be max - min + 1
    int max = *max_element(arr.begin(), arr.end());
    int min = *min_element(arr.begin(), arr.end());
    int range = max - min + 1;

    vector <int> freqArr(range);
    for(int i = 0; i<n; i++){
        freqArr[arr[i]]++;
    }
    return freqArr;
}

void countingSort(vector <int> &arr, int n){
    //the length of our frequency counter array would be max - min + 1
    int max = *max_element(arr.begin(), arr.end());
    int min = *min_element(arr.begin(), arr.end());
    int range = max - min + 1;

    vector <int> freq(range);
    freq = freqCounter(arr, n);

    for(int i = 0; i<freq.size(); i++){
        for(int j = freq[i]; j>0; j--){
            cout<<i<<" ";
        }
    }
}

int main(){
    int n;
    cin>>n;
    vector <int> arr(n);
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }
    countingSort(arr, n);
    return 0;
}
```

```
/*
count sort for negative numbers also.
bit tricky sob
*/
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void countSort(vector <int> &arr, int n)
{
    int max = *max_element(arr.begin(), arr.end());
    int min = *min_element(arr.begin(), arr.end());
    int range = max - min + 1;

    vector<int> count(range), output(n);
    for(int i = 0; i < n; i++)
        count[arr[i]-min]++;

    for(int i = 1; i < count.size(); i++)
           count[i] += count[i-1];

    for(int i = n-1; i >= 0; i--)
    {
         output[ count[arr[i]-min] -1 ] = arr[i];
              count[arr[i]-min]--;
    }

    for(int i=0; i < n; i++)
            arr[i] = output[i];
}

void printArray(vector <int> &arr, int n)
{
    for (int i=0; i < n; i++)
        cout << arr[i] << " ";
    cout << "\n";
}

int main()
{
    int n;
    cin>>n;
    vector<int> arr(n);
    for(int i = 0; i<n; i++){
        cin>>arr[i];
    }
    countSort (arr, n);
    printArray (arr, n);
    return 0;
}
```

```
/*
frequency counter

dnvjfnvlksdmknnvvvvvvvvvvvv

27
0  0  0  2  0  1  0  0  0  1  2  1  1  4  0  0  0  0  1  0  0  14  0  0
0  0
v
14

*/

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

int main() {
      char ch1[100];
      cin>>ch1;
      int l1 = strlen(ch1);
      cout<<l1<<endl;

    //26-element frequency measuring array
      int freq[26] = {0};
      for(int i = 0; i<l1; i++){
          freq[ch1[i] - 'a']++;
      }
      for(int i = 0; i<26; i++){
        cout<<freq[i]<<"  ";
      }

      cout<<endl;

    //finding the maximum index of the freq array.
      int index = 0, max = freq[0];
      for(int i = 1; i<=25; i++){
          if(freq[i]>max){
              max = freq[i];
              index = i;
          }
      }

      cout<<char(index + 'a'); //typecasting, because the '+' sign would
convert the value to integer.
    cout<<max<<endl;  //the frequency of the max repeating character.

      return 0;
}
```

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

void insertionSort(int n, vector <int> &arr){
    int i, j;
    for(i = 1; i<n; i++){
        int curr = arr[i];
        j = i-1;
        while(arr[j]>curr && j>=0){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = curr;
    }
}

int main(){
    int n;
    cin>>n;
    vector <int> arr(n);
    for(int i =0; i<n; i++){
        cin>>arr[i];
    }

    insertionSort(n, arr);
    for(int i =0; i<n; i++){
        cout<<arr[i]<<endl;
    }
    return 0;
}
```

```
/*
LOWER UPPER
Print "lowercase" if user enters a character between 'a-z' ,
Print "UPPERCASE" is character lies between 'A-Z' and
print "Invalid" for all other characters like $,.^# etc.
*/

#include <iostream>
using namespace std;

int main() {
      char ch;
      cin>>ch;

      int ascii = int(ch); //typecasting

      if(ch>=65 && ch<=90){
          cout<<"UPPERCASE";
      }
      else if(ch>=97 && ch<=122){
          cout<<"lowercase";
      }
      else
          cout<<"Invalid";
      return 0;
}
```

```
/*
Piyush enters the park with strength S.
The park is filled with some obstacles denoted by '.' ,
some magical beans denoted by '*' and some blockades denoted as '#'.
If he encounters an obstacle (.) , strength decreases by 2.
If he encounters a magic bean (' * ') , his strength increases by 5.
Piyush can only walk row wise, so he starts from left of a row and moves
towards right
and he does this for every row. However when he encounters a blockade (#)
,
he cannot go any further in his current row and
simply jumps to the start of a new line without losing any strength.
Piyush requires a strength of 1 for every step.
His strength should always be greater than K while traversing
or else Piyush will get lost. Assume that Piyush can shift immediately
from last of one row to the start of next one without loss of any
strength,
help out Piyush to escape the park. His escape is successful if he is
able to return home with atleast K strength.

Sample Input
4 4 5 20
. . * .
. # . .
* * . .
. # * *

Sample Output
Yes
13
*/
#include<iostream>
using namespace std;

int main() {
  int r, c, s, k;
  cin >> r >> c >> k >> s;

  char a[100][100];

  for (int i = 0; i < r; i++) {
    for (int j = 0; j < c; j++) {
      cin >> a[i][j];
    }
  }

  for (int i = 0; i < r; i++) {
    for (int j = 0; j < c; j++) {
      if (s >= k) {
        if (a[i][j] == '.') {
          s = s - 2;
        } if (a[i][j] == '*') {
          s = s + 5;
        } if (a[i][j] == '#') {
          break;
        } if (j<c-1) {
          s--;
        }
      }
    }
```

```
  }

  if (s >= k) {
    cout << "Yes" << endl << s;
  } else {
    cout << "No";
  }

  return 0;
}
```

```
/*
1) Calculate the medians m1 and m2 of the input arrays ar1[]
   and ar2[] respectively.
2) If m1 and m2 both are equal then we are done.
     return m1 (or m2)
3) If m1 is greater than m2, then median is present in one
   of the below two subarrays.
    a)  From first element of ar1 to m1 (ar1[0...|_n/2_|])
    b)  From m2 to last element of ar2  (ar2[|_n/2_|...n-1])
4) If m2 is greater than m1, then median is present in one
   of the below two subarrays.
   a)  From m1 to last element of ar1  (ar1[|_n/2_|...n-1])
   b)  From first element of ar2 to m2 (ar2[0...|_n/2_|])
5) Repeat the above process until size of both the subarrays
   becomes 2.
6) If size of the two arrays is 2 then use below formula to get
  the median.
    Median = (max(ar1[0], ar2[0]) + min(ar1[1], ar2[1]))/2
*/

/*
two sorted arrays,
merge them, sort them,
now find the median of the merged array.
*/

#include <iostream>
using namespace std;

int medianCalculator(int a[], int n){
    if(n%2 == 0){
        return (a[n/2] + a[n/2 - 1])/2;
    }
    return a[n/2];
}

int median(int arr1[], int arr2[], int n){
    if(n==0){
        return -1;
    }

    else if(n==1){
        return (arr1[0] + arr2[0])/2;
    }

    else if(n == 2){
        return (max(arr1[0], arr2[0]) + min(arr1[1], arr2[1]))/2;
    }

    int m1 = medianCalculator(arr1, n);
    int m2 = medianCalculator(arr2, n);

    if(m1 == m2)
    return m1;

    //recursive
    else if (m1 < m2){
        if (n % 2 == 0){
            return median(arr1 + n / 2 - 1, arr2, n - n / 2 + 1);
        }
```

```
            return median(arr1 + n / 2, arr2, n - n / 2);
    }

    else if(m1 > m2){
    if (n % 2 == 0){
        return median(arr2 + n / 2 - 1, arr1, n - n / 2 + 1);
    }
        return median(arr2 + n / 2, arr1, n - n / 2);
    }
}

int main() {
     int n;
     cin>>n;

     int arr1[1000], arr2[1000];

     for(int i = 0; i<n; i++){
         cin>>arr1[i];
     }
     for(int i = 0; i<n; i++){
         cin>>arr2[i];
     }

     cout<<median(arr1, arr2, n);
     return 0;
}
```

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

char *mystrtok(char str[], char delim[]){
    int i;
    //make a copy of the input, for str not equal to NULL.
    static char *input = NULL;
    if(str != NULL){
        input = str;
    }
    //check if the input is null
    if(input == NULL) return NULL;
    //create an output string, of the size, one more than the size of the
input string.
    char *output = new char[strlen(input) + 1];
    //iterate over the string, and where input:
    for(i = 0; input[i] != '\0'; i++){
        //if not delimiter
        if(input[i] != delim[0]){
            output[i] = input[i];
        }
        //if it is a delimiter
        else{
            output[i] = '\0';
            input = input +i+1;
            return output;
        }
    }
    output[i] = '\0';
    input = NULL;
    return output;
}

int main(){
    //enter the string, that can have whitespaces
    string str;    getline(cin, str);
    //enter the delimiter
    string delimiter;    cin>>delimiter;
    //create pointer for string tokenising
    char *ptr;
    //convert the string to the pointer to the dynamic array
    ptr = mystrtok((char*)str.c_str(), (char*)delimiter.c_str());

    while(ptr != NULL){
        cout<<ptr<<endl;
        ptr = mystrtok(NULL, (char*)delimiter.c_str());
    }

    return 0;
}
```

```
/*
give threshold, count
arrange them im order of salary (print only those who have salaries
greater than threshold).
If salaries are equal, arrange lexicographically.

sample input:

100 5
nick 1000
ram 90
amit 200
yug 300
john 300

output:
amit 200
john 300
yug 300
nick 1000
*/
#include <bits/stdc++.h>
using namespace std;

bool comparator(pair<string, int>p1, pair<string, int>p2){
    if(p1.second == p2.second){
        return p1.first < p2.first;
    }
    return p1.second < p2.second;
}

int main() {
      int min_salary, n;
      cin>>min_salary>>n;

      pair <string, int> empData[100005];
      string name;
      int salary;

      for(int i=0; i<n; i++){
          cin>>name>>salary;
          empData[i].first = name;
          empData[i].second = salary;
      }

      sort(empData, empData + n, comparator);

    for(int i = 0; i<n; i++){
        if(empData[i].second >= min_salary){
            cout<<empData[i].first<<" "<<empData[i].second;
        }
    }
      return 0;
}
```

```
/*
string rotate

hello
2

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

void rotate(char a[], int k){
    int i = strlen(a);

    //shifting entire string by k places
    while(i>=0){
        a[i+k] = a[i];
        i--;
    }

    i = strlen(a);
    int j = i-k;
    int s = 0;

    while(j<=(i-1)){
        a[s] = a[j];
        s++;
        j++;
    }
    a[i-k] = '\0';

    cout<<a;
}

int main() {
     char str[100];
    cin>>str;
     int n;
     cin>>n;

     rotate(str, n);
     return 0;
}
```

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

void swapFunc(int *a, int *b){
    int temp = *a;
    *a = *b;
    *b = temp;
}

void selectionSort(int n, vector <int> &arr){
    int i, j;
    int minValue = arr[0], minIndex;
    for(i = 0; i<n-1; i++){
        minIndex = i;
        for(j = i+1; j<n; j++){
            if(arr[j] < arr[minIndex]){
                minValue = arr[j];
                minIndex = j;
            }
        }
        swapFunc(&arr[i], &arr[minIndex]);
    }
}

int main(){
    int n;
    cin>>n;

    vector <int> arr(n);
    for(int i = 0; i<n; i++){
        cin>>arr[i];
    }
    selectionSort(n, arr);

    for(int i=0; i<n; i++){
        cout<<arr[i];
    }

    return 0;
}
```

```
/*
Sample Input
1   //testcases
10111 10000

Sample Output
00111

compare the two strings characterwise, if equal => append 0, if not equal
=> append 1;
*/
#include <bits/stdc++.h>
using namespace std;

int main() {
      int times;
      cin>>times;

      while(times--){
          int len = 0;
          string s1,s2,s3;
          cin>>s1;
          cin>>s2;

         len = s1.length();
          for(int i = 0; i<len; i++){
              if(s1[i] == s2[i]){
                  s3.append(1,'0');
              }
              else{
                  s3.append(1, '1');
              }
          }
         cout<<s3<<endl;
      }
      return 0;
}
```

```
/*
Sample Input
aaabbccds

Sample Output
a3b2c2d1s1
*/#include<bits/stdc++.h>
using namespace std;

int main(){
    char str[100];
    cin>>str;
    int len = strlen(str);
    int freq[26] = {0};

    for(int i = 0; i<len; i++){
        int counter = 1;

        while(i<len && str[i] == str[i+1]){
            counter++;
            i++;
        }

        cout<<str[i]<<counter;
    }

    return 0;
}
```

```
/*
i/p:
champ
champions
champion
championss

o/p:
championsss  champions  champion  champ
*/

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

bool comparator(string a, string b){
    return a>b;
}

int main() {
      string str[100];
    for(int i=0; i<4; i++){
        getline(cin, str[i]);
    }
    sort(str, str+4, comparator);
    for(int i = 0; i<4; i++){
        cout<<str[i]<<"  ";
    }
      return 0;
}
```

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

int main(){
    char str[] = "Hello C++, I am Akshat.";
    char *ptr; //make the pointer for tokenising

    ptr = strtok(str, " "); //space is the delimiter
    while(ptr != NULL){
        //first will the string element tokenised from str parameter.
        cout<<ptr<<endl;
        /*
            Now for the further successive terms in the string, null is
provided as the parameter.
            NUll in the subsequent calls will give next tokens.
            repeatedly giving str as the parameter will give the same
first word again and again.
        */

        /*
            When strtok is passed a static pointer is ctreated that
points to the next
            position already (after delimiter), and a dynamic array,
storing this address of
            the next token is created.
            Upon calling NULL this address is visited, but on calling
str, the first index is revisted.
        */
        ptr = strtok(NULL, " ");
    }
    return 0;
}
```

```
/*
ARRAYS-SUM OF TWO ARRAYS

Take as input N, the size of array. Take N more inputs and store that in
an array.
Take as input M, the size of second array and take M more inputs and
store that in second array.
Write a function that returns the sum of two arrays.
Print the value returned.
*/

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

int countDigits(int num){
    return floor(log10(num) + 1);
}

int main() {
     int n, m;

     cin>>n;
    vector<int> arr1(n);
    for(int i = 0; i<n; i++){
        cin>>arr1[i];
    }
    cin>>m;
    vector<int> arr2(m);
    for(int i = 0; i<m; i++){
        cin>>arr2[i];
    }

    int num1 = 0, num2 = 0, p = 1;
    for(int i = n-1; i>=0; i--){
        num1 = num1 + (arr1[i] * p);
        p *= 10;
    }
    // cout<<num1<<endl;
    p=1;
    for(int i = m-1; i>=0; i--){
        num2 = num2 + (arr2[i] * p);
        p *= 10;
    }
    // cout<<num2<<endl;

    int ans = 0;
    ans = num1 + num2;
    // cout<<ans<<endl;
    int length = 0;
    length = countDigits(ans);
    // cout<<length<<endl;

    vector<int> arr3(length);


    for(int i  = length-1; i>=0; i--){
        if(ans > 0){
            // cout<<ans<<"  ";
            arr3[i] = ans%10;
            ans /= 10;
```

```
            // cout<<arr3[i]<<endl;
        }
    }

    for(int i = 0; i<arr3.size(); i++){
        cout<<arr3[i]<<", ";
    }
    cout<<"END";

      return 0;
}
```