REPUBLIKA E SHQIPËRISË
MINISTRIA E ARSIMIT
DHE SPORTIT
AGJENCIA KOMBËTARE E PROVIMEVE

OLIMPIADA KOMBËTARE E INFORMATIKËS

Faza e tretë

Viti mësimor 2015-2016 **16 prill 2016**

1. Renditja e vektorit

Jepet një vektor me **N** numra të plotë. Rendisni elementet e tij në një mënyrë të tillë që shuma e çdo **K** elementeve të njëpasnjëshëm të vektorit të renditur të jetë e plotpjestueshme me numrin **M**.

Input 25 pikë

Rreshti i pari i inputit përmban një numër të plotë **T,** numrin e rasteve për testim (jo më shumë se 5).
Çdo rast testimi përshkruhet nëpërmjet dy rreshtave : Rreshti i parë përmban tre numra të plotë, **N**, **M** dhe **K** të tillë që : $1 \leq \mathbf{K} \leq 70$, $2*\mathbf{K} \leq \mathbf{N} \leq 10000$, $1 \leq \mathbf{M} \leq 10^9$.
Rreshti i dytë përmban N numra të plotë $\mathbf{A_i}$ ($0 \leq \mathbf{A_i} \leq 10^9$), që përfaqësojnë elementet e vektorit, të ndarë me hapësirë.

Output

Për çdo rast testimi, programi duhet të japë si output vetëm një rresht, që përmban numrin -1 në rast se nuk ka zgjidhje, ose një sekuencë prej N numrash të plotë që plotësojnë kriterin dhe janë një renditje e elementeve të vektorit A nga input. Nëse ka disa mënyra për ta plotësuar kriterin, secila prej tyre do të pranohet si e saktë.

Shembull

Input:
```
3
6 4 3
6 1 1 9 2 5
8 2 3
2 5 0 9 1 9 9 4
8 5 3
1 9 0 1 1 9 9 1
```

Output:
```
2 9 1 6 5 1
2 5 9 0 9 9 4 1
-1
```

Shpjegimi

Ka 6 sekuenca prej 3 numrash të njëpasnjëshëm në (2 5 9 0 9 9 4 1). Formalisht:

- (2 5 9), shuma = 16, që plotëpjestohet me 2
- (5 9 0), shuma = 14, që plotëpjestohet me 2
- (9 0 9), shuma = 18, që plotëpjestohet me 2
- (0 9 9), shuma = 18, që plotëpjestohet me 2
- (9 9 4), shuma = 22, që plotëpjestohet me 2
- (9 4 1), shuma = 14, që plotëpjestohet me 2
  Një tjetër zgjidhje e mundshme është (4 5 9 0 1 9 2 9).

Një mënyrë zgjidhje në C++ është:

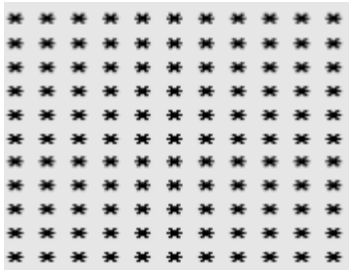
```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <string>
#include <map>
#include <vector>
#include <stack>

using namespace std;

template <typename T> T sqr(T x) { return x * x; }
template <typename T> T abs(T x) { return x < 0? -x : x; }

const int MAXN = 10007;
const int MAXK = 80;

#define A first
#define S second

#define left dsalkhfas
#define right sdjkfhkas

typedef map < int, stack <int> > mmap;

int ans[MAXN];
mmap M;
int n, m, k;
int L, H, N;
int left, right;

vector < pair <int, int> > a, b;
map < int, int > f[MAXK][MAXK], g[MAXK][MAXK];

void solve(vector < pair <int, int> > &a, map < int, int > f[MAXK][MAXK])
{
        for (int i = 0; i <= (int)a.size(); i++)
            for (int j = 0; j <= N; j++)
                f[i][j].clear();

        f[0][0][0] = 0;
        for (int i = 0; i < (int)a.size(); i++)
            for (int j = 0; j <= N; j++)
                for (int k = 0; k <= N - j && k * H <= a[i].S; k++)
                    if ((a[i].S - k * H) % L == 0)
                        for (map < int, int > :: iterator iter = f[i][j].begin();
iter != f[i][j].end(); iter++)
                            f[i + 1][j + k][(iter->first + (k + (a[i].S - k * H) / L)
* (long long)a[i].A) % m] = k;
}


void recovery(vector < pair <int, int> > &a, map < int, int > f[MAXK][MAXK], int x,
int y, int z)
{
    while (x)
    {
        int c = f[x][y][z];
        x--;

        stack <int> &st = M[a[x].A];
        for (int i = 0; i < c; i++)
        {
```

```
            for (int j = 0; j < H; j++)
            {
                ans[left + k * j] = st.top();
                st.pop();
            }
            left++;
        }

        for (int i = 0; i < (a[x].S - c * H) / L; i++)
        {
            for (int j = 0; j < L; j++)
            {
                ans[right + k * j] = st.top();
                st.pop();
            }
            right--;
        }

        z = ((z - (c + (a[x].S - c * H) / L) * (long long)a[x].A) % m + m) % m;
        y -= c;
    }
}

int main()
{
    #ifndef ONLINE_JUDGE
        freopen("in ", "r", stdin);
        freopen("out", "w", stdout);
    #endif
    int t;
    scanf("%d", &t);
    while (t--)
    {
        M.clear();
        scanf("%d %d %d", &n, &m, &k);
        for (int i = 0; i < n; i++)
        {
            int x;
            scanf("%d", &x);
            M[x % m].push(x);
        }
        if ((int)M.size() > k)
        {
            puts("-1");
            continue;
        }
        a.clear();
        b.clear();
        for (mmap :: iterator iter = M.begin(); iter != M.end(); ++iter)
        {
            if (a.size() == b.size())
                a.push_back(make_pair(iter->first, iter->second.size()));
            else
                b.push_back(make_pair(iter->first, iter->second.size()));
        }

        H = n / k + 1;
        L = H - 1;
        N = n % k;

        solve(a, f);
        solve(b, g);

        bool flg = false;
        left = 0;
        right = k - 1;
```

```
        for (int i = 0; i <= N && !flg; i++)
            for (map < int, int > :: iterator iter = f[(int)a.size()][i].begin();
iter != f[(int)a.size()][i].end(); iter++)
                if (g[(int)b.size()][N - i].find((m - iter->first) % m) !=
g[(int)b.size()][N - i].end())
                {
                    recovery(a, f, a.size(), i, iter->first);
                    recovery(b, g, b.size(), N - i, (m - iter->first) % m);
                    flg = true;
                    break;
                }

        if (!flg)
        {
            puts("-1");
            continue;
        }

        for (int i = 0; i < n; i++)
        {
            if (i) printf(" ");
            printf("%d", ans[i]);
        }
        printf("\n");
    }

    return 0;
}
```

2. Labirinti

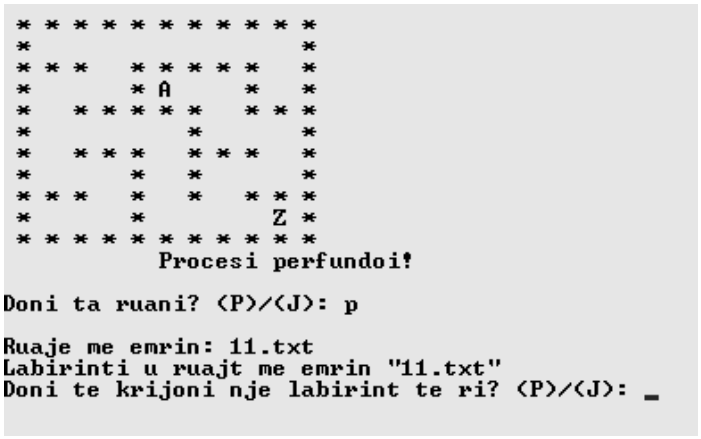
Secili prej nesh ka luajtur me labirintet. Shpesh kalonim orë e orë për të gjetur zgjidhjen, por shumë pak mund të jenë munduar të krijojnë një të tillë vetë.
Ndërtoni një program që gjeneron në mënyrë random një labirint në formë drejtkëndore ku të shfaqet pika e fillimit dhe ajo e mbarimit.
Më poshtë po ju japim një mënyrë si të krijoni labirintin:
Konfigurimi fillestar duhet të jetë në mënyrë të tillë që të gjithë muret të jenë të ngritura si në figurën e mëposhtme, gjë e cila krijohet duke përfshirë çdo qelizë brenda grupit të saj.



Merrni një mur në mënyrë të rastësishme dhe kontrolloni nëse qelizat në dy anët e murit i përkasin të njëjtit grup apo jo. Nëse po, atëhere ato janë në të njëjtin grup dhe janë të lidhura me njëra-tjetrën.
Nëse jo, atëhere hiqeni murin dhe bashkoni qelizat duke i përfshirë ato në të njëjtin grup.
Në qoftë se i heqim muret në këtë mënyrë, pas disa veprimesh, të gjithë qelizat do t'i përkasin të njëjtit grup, ndërkohë që vetëm disa mure janë hequr.
Në këtë pikë keni krijuar labirintin tuaj. **25 pikë**

Output: *(Shih figurën më poshtë)*

Labirinti i krijuar në mënyrë rastësore dhe një mesazh që tregon krijimin e tij.
Mundësi për ta ruajtur labirintin dhe për të krijuar një të ri ose jo.



(me A është shënuar fillimi dhe Z fundi)

Një mënyrë zgjidhje në C++ është:

```
#include <iostream>
#include <ctime>
#include <windows.h>
#include <conio.h>
#include <stack>
#include <fstream>
```

```
using namespace std;

#define SIZE 11

struct Cell
{
        bool visited;
        bool top_wall;
        bool bot_wall;
        bool left_wall;
        bool right_wall;
        char display;
};

void Initialize(Cell Level[][SIZE]);
void ClearScreen();
void Redraw(Cell Level[][SIZE]);
void GenerateMaze(Cell Level[][SIZE], int &posX, int &posY, int &goalX, int &goalY);
void SaveMaze(Cell Level[][SIZE]);

int main() {
        Cell Level[SIZE][SIZE];
        int posX = 0;
        int posY = 0;
        int goalX = 0;
        int goalY = 0;
        bool game_over = false;

        while(!game_over) {
                system("cls");
                Initialize(Level);
                Redraw(Level);
                GenerateMaze(Level, posX, posY, goalX, goalY);
                SaveMaze(Level);

                char input;
                cout << "Doni te krijoni nje labirint te ri? (P)/(J): ";
                cin >> input;

                if((input != 'j') && (input != 'J') && (input != 'p') && (input !=
'P'))
                        cout << "Po ose Jo" << endl;
                else if((input == 'j') || (input == 'J')) {
                        game_over = true;
                        cout << "Faleminderit!" << endl;
                }
        }

        _getch();
        return 0;
}

void Initialize(Cell Level[][SIZE]) {
        for(int i=0; i<SIZE; i++) {
                for(int j=0; j<SIZE; j++) {
                        Level[i][j].display = '*';
                        Level[i][j].visited = false;
                        Level[i][j].top_wall = true;
                        Level[i][j].bot_wall = true;
                        Level[i][j].left_wall = true;
                        Level[i][j].right_wall = true;
                }
        }
        for(int i=1; i<SIZE-1; i++) {
```

```
                for(int j=1; j<SIZE-1; j++) {
                        Level[1][j].top_wall = false;
                        Level[SIZE-2][j].bot_wall = false;
                        Level[i][1].left_wall = false;
                        Level[i][SIZE-2].right_wall = false;
                }
        }
}

void ClearScreen()
{
        HANDLE hOut;
        COORD Position;
        hOut = GetStdHandle(STD_OUTPUT_HANDLE);
        Position.X = 0;
        Position.Y = 0;
        SetConsoleCursorPosition(hOut, Position);
}

void Redraw(Cell Level[][SIZE]) {
        for(int i=0; i<SIZE; i++) {
                cout << endl;
                for(int j=0; j< SIZE; j++)
                        cout << " " << Level[i][j].display;
        }
}

void GenerateMaze(Cell Level[][SIZE], int &posX, int &posY, int &goalX, int &goalY) {
        srand((unsigned)time(NULL));
        int random = 0;
        int randomX = ((2*rand())+1)%(SIZE-1);
        int randomY = ((2*rand())+1)%(SIZE-1);
        posX = randomX;
        posY = randomY;
        int visitedCells = 1;
        int totalCells = ((SIZE-1)/2)*((SIZE-1)/2);
        int percent = 0;
        stack<int> back_trackX, back_trackY;

        Level[randomY][randomX].display = 'A';
        Level[randomY][randomX].visited = true;

        while(visitedCells < totalCells)
        {
                if(((Level[randomY-2][randomX].visited == false) &&
(Level[randomY][randomX].top_wall == true && Level[randomY-2][randomX].bot_wall ==
true)) ||
                   ((Level[randomY+2][randomX].visited == false) &&
(Level[randomY][randomX].bot_wall == true && Level[randomY+2][randomX].top_wall ==
true)) ||
                   ((Level[randomY][randomX-2].visited == false) &&
(Level[randomY][randomX].left_wall == true && Level[randomY][randomX-2].right_wall ==
true)) ||
                   ((Level[randomY][randomX+2].visited == false) &&
(Level[randomY][randomX].right_wall == true && Level[randomY][randomX+2].left_wall ==
true)))
                {
                        random = (rand() % 4) + 1;


                        if((random == 1) && (randomY > 1)) {
                                if(Level[randomY-2][randomX].visited == false) {
                                        Level[randomY-1][randomX].display = ' ';
                                        Level[randomY-1][randomX].visited = true;
                                        Level[randomY][randomX].top_wall = false;
```

```
                        back_trackX.push(randomX);
                        back_trackY.push(randomY);

                        randomY -= 2;
                        Level[randomY][randomX].visited = true;
                        Level[randomY][randomX].display = ' ';
                        Level[randomY][randomX].bot_wall = false;
                        visitedCells++;
                }
                else
                        continue;
        }

        else if((random == 2) && (randomY < SIZE-2)) {
                if(Level[randomY+2][randomX].visited == false) {
                        Level[randomY+1][randomX].display = ' ';
                        Level[randomY+1][randomX].visited = true;
                        Level[randomY][randomX].bot_wall = false;

                        back_trackX.push(randomX);
                        back_trackY.push(randomY);

                        randomY += 2;
                        Level[randomY][randomX].visited = true;
                        Level[randomY][randomX].display = ' ';
                        Level[randomY][randomX].top_wall = false;
                        visitedCells++;
                }
                else
                        continue;
        }

        else if((random == 3) && (randomX > 1)) {
                if(Level[randomY][randomX-2].visited == false) {
                        Level[randomY][randomX-1].display = ' ';
                        Level[randomY][randomX-1].visited = true;
                        Level[randomY][randomX].left_wall = false;

                        back_trackX.push(randomX);
                        back_trackY.push(randomY);

                        randomX -= 2;
                        Level[randomY][randomX].visited = true;
                        Level[randomY][randomX].display = ' ';
                        Level[randomY][randomX].right_wall = false;
                        visitedCells++;
                }
                else
                        continue;
        }

        else if((random == 4) && (randomX < SIZE-2)) {
                if(Level[randomY][randomX+2].visited == false) {
                        Level[randomY][randomX+1].display = ' ';
                        Level[randomY][randomX+1].visited = true;
                        Level[randomY][randomX].right_wall = false;

                        back_trackX.push(randomX);
                        back_trackY.push(randomY);

                        randomX += 2;
                        Level[randomY][randomX].visited = true;
                        Level[randomY][randomX].display = ' ';
                        Level[randomY][randomX].left_wall = false;
                        visitedCells++;
                }
```

```
                                else
                                        continue;
                        }

                        percent = (visitedCells*100/totalCells*100)/100;
                        cout << endl << "       Duke gjeneruar labirintin... " <<
percent << "%" << endl;
                }
                else {
                        randomX = back_trackX.top();
                        back_trackX.pop();

                        randomY = back_trackY.top();
                        back_trackY.pop();
                }

                ClearScreen();
                Redraw(Level);
        }

        goalX = randomX;
        goalY = randomY;
        Level[goalY][goalX].display = 'Z';
        system("cls");
        ClearScreen();
        Redraw(Level);
        cout << endl << "\a\t   Procesi perfundoi!" << endl;
}

void SaveMaze(Cell Level[][SIZE]) {
        ofstream output;
        char file[20];
        char input;

        cout << endl << "Doni ta ruani? (P)/(J): ";
        cin >> input;

        if ((input == 'p') || (input == 'P')) {
                cout << endl << "Ruaje me emrin: ";
                cin >> file;

                output.open(file);

                for (int i = 0; i < SIZE; i++) {
                        output << endl;
                        for (int j = 0; j < SIZE; j++) {
                                output << Level[i][j].display << " ";
                        }
                }

                cout << "Labirinti u ruajt me emrin " << "\"" << file << "\"" <<
endl;
                output.close();
        }

}
```

3. Loja: Vargu me fjalë

Për të zhvilluar këtë lojë si fillim merrni një fjalë. Duke filluar nga fjala e parë, në çdo hap do ndryshoni vetëm një gërmë. Kështu do t'ju krijohet një varg me fjalë ku çdo fjalë ndryshon nga fqinjët e vet vetëm me një gërmë.

25 pikë

Shembull :
Filloni lojën me fjalën **loja** dhe mbarojeni me fjalën **besa** (kalimi të bëhet me 7 fjalë sipas rregullit të mësipërm)

loja – boja – bora – dora – dera – vera – vesa – besa

Programi do të funksionojë në këtë mënyrë:

- Si input merr një file **fjalori.txt** që do të përmbajë fjalë me **4** gërma. Në qoftë se ka fjalë me më pak apo më shumë gërma zgjidhni t'i injoroni ato.
- Kërkon nga përdoruesi gjatësinë e vargut (nga 1-20). Fjala e fillimit nuk merret parasysh.
- Programi kërkon në file-n **fjalor.txt** dhe gjen vargjet me atë gjatësi (në qoftë se nuk gjendet një varg me gjatësinë e dhënë afishon një mesazh gabimi).
- Programi jep fjalën e parë dhe të fundit të vargut dhe përdoruesi duhet ta plotësojë atë.
- Në qoftë se përdoruesi ka futur një fjalë gabimisht jepini mundësinë ta korigjojë atë.
- Në qoftë se përdoruesi e plotëson vargun në menyrë të saktë, ai njoftohet se është fitues.

Fjalët duhet të jenë me katër gërma dhe nuk duhen përsëritje fjalësh në varg : *Psh : loja – boja – loja*

Shembull:
Input:
File **fjalor.txt** me fjalët si më poshtë:
ariu
besa
boja
bora
dera
dora
foto
loja
mora
soba
soja
vera
vesa

```
Jepni gjatesine e vargut (1 - 20 - fjala e fillimit nuk numerohet): 5
Kliko d per te dale, r per te rifilluar lojen,
dhe h per ndihme (help).

Plotesoni vargun me fjalet nga  vesa ne boja:
vesa
vera
dera
dora
bora
boja

Urime!Ju fituat lojen!!
```

Output:

Një mënyrë zgjidhje në C++ është:

```
#include <iostream>
#include <fstream>
#include <cctype>
#include <ctime>
#include <vector>
#include <string>
#include <cstdlib>
using namespace std;

bool bitmap[26][26][26][26];
vector<string> bitmapv, words;

bool isValid(string, string);
void ladder(string, int, bool&);
bool isRepeated(string);
void remove(string);
bool allLetter(string);

int main() {
    int i, i2, i3, i4, transition, n;
    ifstream fin;
    string word, currWord;
    bool success;
    srand(time(0));
    for(i=0; i<26; ++i)
        for(i2=0; i2<26; ++i2)
            for(i3=0; i3<26; ++i3)
                for(i4=0; i4<26; ++i4)
                    bitmap[i][i2][i3][i4] = false;

    fin.open ("fjalor.txt");

    if(!fin.is_open()) {
        cout <<" File nuk mund te hapet!\n";
        exit(2);
    }
    while(fin>>word) {
        if(word.length()!=4 || !allLetter(word)) {
            cout << "Fjalet nuk jane ne gjatesine e kerkuar\n";
            exit(3);
        }
        bitmap[word[0]-'a'][word[1]-'a'][word[2]-'a'][word[3]-'a'] = true;
        bitmapv.push_back(word);
    }
    fin.close();
    cout << "Jepni gjatesine e vargut (1 - 20 - fjala e fillimit nuk numerohet): ";
    while(!(cin>>transition) || transition<1 || transition>20) {
        if(!cin) {
            cin.clear();
            while(cin.get()!='\n') ;
        }
        cout << "Jepni gjatesine e vargut (1 - 20 - fjala e fillimit nuk numerohet):
";
    }
    success = false;

    while(!success) {
        words.clear();
        if(bitmapv.size()==0) {
            cout << "Me vleren e dhene nuk mund te krijohet varg nga lista e fjaleve
qe gjenden ne file.\n";
```

```
            exit(4);
        }
        word = bitmapv[rand()%bitmapv.size()];
        remove(word);
        ladder(word, transition, success);
    }

    cout << "Kliko d per te dale, r per te rifilluar lojen,\n";
    cout << "dhe h per ndihme (help).\n";
    cout << "\nPlotesoni vargun me fjalet nga  " << words.front() << " ne " <<
words.back()              << ":\n";
    cout << words.front() << endl;
    n = 0;
    currWord = words.front();
    while(n<transition) {
        cin >> word;
        if(word=="d") {
            break;
        }
        if(word=="r") {
            cout << "\nrifillo...";
            n = 0;
            currWord = words.front();
            cout << "Mbaroi\n";
            cout << "\nPlotesoni vargun me fjalet nga  " << words.front() << " ne "
<< words.back()              << ":\n";
            cout << words.front() << endl;
            continue;
        }
        if(word=="h") {
            cout << "\nZgjidhja:\n";
            for(i=0; i<words.size(); i++)
                cout << words[i] << endl;
            break;
        }
        if(word.length()!=4 || !allLetter(word) || !isValid(currWord, word)) {
            cout << word << " nuk eshte e vlefshme.Jepni nje fjale tjeter..: ";
            continue;
        }
        currWord = word;
        ++n;
    }
    if(currWord==words.back())
        cout << "\nUrime!Ju fituat lojen!!\n";
    else
        cout << "\nNa vjen keq!Provoni perseri!.\n";
    return 0;
}

bool isValid(string first, string second) {
    int i, counter;
    counter = 0;
    if(!bitmap[second[0]-'a'][second[1]-'a'][second[2]-'a'][second[3]-'a'])
        return false;
    for(i=0; i<4; i++) {
        if(first[i]!=second[i])
            ++counter;
    }
    return (counter==1);
}

void ladder(string w, int t, bool & success) {
    words.push_back(w);
    if(t==0) {
        success = true;
        return;
```

```
    }
    string newWord = w;
    char c;
    for(c='a'; c<='z' && !success; ++c) {
        newWord[0] = c;
        if(bitmap[newWord[0]-'a'][newWord[1]-'a'][newWord[2]-'a'][newWord[3]-'a'] &&
!isRepeated(newWord)) {
            ladder(newWord, t-1, success);
            if(!success)
                words.pop_back();
        }
    }
    newWord = w;
    for(c='a'; c<='z' && !success; ++c) {
        newWord[1] = c;
        if(bitmap[newWord[0]-'a'][newWord[1]-'a'][newWord[2]-'a'][newWord[3]-'a'] &&
!isRepeated(newWord)) {
            ladder(newWord, t-1, success);
            if(!success)
                words.pop_back();
        }
    }
    newWord = w;
    for(c='a'; c<='z' && !success; ++c) {
        newWord[2] = c;
        if(bitmap[newWord[0]-'a'][newWord[1]-'a'][newWord[2]-'a'][newWord[3]-'a'] &&
!isRepeated(newWord)) {
            ladder(newWord, t-1, success);
            if(!success)
                words.pop_back();
        }
    }
    newWord = w;
    for(c='a'; c<='z' && !success; ++c) {
        newWord[3] = c;
        if(bitmap[newWord[0]-'a'][newWord[1]-'a'][newWord[2]-'a'][newWord[3]-'a'] &&
!isRepeated(newWord)) {
            ladder(newWord, t-1, success);
            if(!success)
                words.pop_back();
        }
    }
}

bool isRepeated(string w) {
    int i;
    for(i=0; i<words.size(); ++i)
        if(words[i]==w)
            return true;
    return false;
}

void remove(string w) {
    vector<string>::iterator vi;
    for(vi=bitmapv.begin(); vi!=bitmapv.end(); ++vi)
        if(*vi==w) {
            bitmapv.erase(vi);
            break;
        }
}

bool allLetter(string w) {
    return (isalpha(w[0])&&isalpha(w[1])&&isalpha(w[2])&&isalpha(w[3]));
}
```

4. Ju jepet një string **S** dhe një numër **N**, ku **N** tregon numrin e kërkimeve në këtë string. Çdo kërkim përcaktohet nga dy numra të plotë L_i dhe P_i. Numëroni stringjet me gjatësi L_i që gjënden ekzaktësisht P_i herë në stringun **S**.

25 pikë

Input

Rreshti i parë i input-it përmban stringun **S**. Rreshti i dytë përmban numrin **N**. Më pas vijnë **N** rreshta të tjerë ku rreshti i i-të përmban numrat L_i dhe P_i për kërkimin e **i**-të.

Output

Për çdo kërkim afishoni zgjidhjen.

Shembull

Input:

abacaba
2
3 2
2 2

Output:

1
2

Shpjegime për zgjidhjen e dhënë
Për kërkimin e parë zgjidhja e vetme është **aba**, ndërsa për kërkimin e dytë zgjidhjet janë: **ab** dhe **ba**.

Një mënyrë zgjidhje në C++ është:

```
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <vector>
#include <cstring>
using namespace std;

#define maxn 200000
#define clone Clone
#define link Link

int next[maxn * 2 + 10][26], len[maxn * 2 + 10], cnt[maxn * 2 + 10], link[maxn * 2 +
10], clone, j;
int ls, sz, last, i, p, q, cur, n, k, l, node, x, y, occ, leng;
pair<int, int> sorter[maxn * 2 + 10];
char s[maxn + 5], c, a[maxn + 5];
vector<int> vb[maxn + 1], ve[maxn + 1];

void build_sa () {

    ls = strlen(s);
    sz = last = 1;
    for(i = 0; i < ls; i++) {
```

```
        c = s[i] - 'a';
        len[cur = ++sz] = len[last] + 1;
        cnt[cur] = 1;
        for(p = last; p > 0 && !next[p][c]; p = link[p]) next[p][c] = cur;
        if (!p) link[cur] = 1; else {
            q = next[p][c];
            if (len[p] + 1 == len[q]) link[cur] = q; else {
                len[clone = ++sz] = len[p] + 1;
                link[clone] = link[q];
                for(j = 0; j < 26; j++) next[clone][j] = next[q][j];
                for(; p && next[p][c] == q; p = link[p]) next[p][c] = clone;
                link[q] = link[cur] = clone;
            }
        }
        last = cur;
    }

    for(i = 1; i <= sz; i++) sorter[i] = make_pair(len[i], i);
    sort(sorter + 1, sorter + sz + 1);
    for(i = sz; i >= 1; i--) {
        k = sorter[i].second;
        cnt[link[k]] += cnt[k];
    }

    for(i = 1; i <= sz; i++) {
        y = len[i];
        x = len[link[i]] + 1;

        vb[cnt[i]].push_back(x);
        ve[cnt[i]].push_back(y + 1);
    }

    for(i = 1; i <= ls; i++) {
        sort(vb[i].begin(), vb[i].end());
        sort(ve[i].begin(), ve[i].end());
    }
}

int count_not_greater (vector<int> &a, int k) {
    if (!a.size()) return 0;
    int l = 0, r = a.size() - 1, mid;
    while (l < r) {
        mid = (l + r + 1) / 2;
        if (a[mid] > k) r = mid - 1; else l = mid;
    }
    if (a[l] > k) return 0; else return l + 1;
}

int main (int argc, char * const argv[]) {
    gets(s);
    build_sa();
    scanf("%d", &n);
    for(i = 1; i <= n; i++) {

        scanf("%d %d", &leng, &occ);
        printf("%d\n", count_not_greater(vb[occ], leng) - count_not_greater(ve[occ],
leng));
    }
    return 0;
}
```