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

Olimpiada Kombëtare e Informatikës **Viti shkollor 2014–2015**

Faza III **28 mars 2015**

Kërkesa	1	2	3
Pikët	15	15	20

1. Loja me gërma **15 pikë**

Në një skedar të krijuar paraprakisht **fjale.txt** vendosni fjalë me 3 deri në 7 karaktere(të gjitha me gërma të vogla). Si fillim bëni renditjen e fjalëve në këtë skedar(renditja të bëhet fillimisht mbi bazën e numrit total të shkronjave që ka fjala dhe më pas sipas rendit alfabetik a-z).Fundi i skedarit tregohet me një (.). Më pas në skedarin **input.txt** fusni një string që përmban nga 3 deri në maksimalisht 7 gërma. Si output nxirrni gjithë kombinimet e mundshme të fjalëve me kuptim që janë paraprakisht te skedari fjale.txt (gjithmonë të renditura).

SHEMBULL :

```
| FJALE.txt |    | INPUT.TXT |     | OUTPUT.TXT |
|___________|    |___________|     |____________|
| profili   |    | prmgroa   |     | program    |
| program   |    |___________|     | gram       |
| pro       |                      | pro        |
| gram      |                      | ram        |
| ram       |                      | rom        |
| rom       |                      |____________|
| .         |
|___________|
```

Zgjidhja :

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <string>
using namespace std;

ifstream fin("fjale.txt");
ifstream din("input.txt");
ofstream fout("output.txt");



 int main ()
 {
      string a, fjale[100], temp;
      int c=0;
      fin>>a;
      while(a!=".")
      {
            fjale[c]=a;
            c++;
            fin>>a;
```

```
	}
	for(int l=1; l<c; l++)
	{
		for(int t=1; t<c; t++)
		{
			if(fjale[t].length()>fjale[t-1].length())
			{
				temp=fjale[t];
				fjale[t]=fjale[t-1];
				fjale[t-1]=temp;
			 }
			 else if(fjale[t].length()==fjale[t-1].length())
			{
				if(fjale[t]<fjale[t-1])
			{
				temp=fjale[t];
				fjale[t]=fjale[t-1];
				fjale[t-1]=temp;
			 }
			 }
		 }
	}

	din>>a;

	 bool flag, bot;
	for(int i=0; i<c; i++)
	{
		bot=1;
		 for(int j=0; j<fjale[i].length(); j++)
		{
			flag=0;
			 for(int k=0; k<a.length(); k++)
			{
				if(a[k]==fjale[i][j])
				{
					flag=1;
					break;
				 }
			 }
			 if(flag==0)bot=0;
		 }
		 if(bot==1)fout<<fjale[i]<<endl;
	 }
	system ("pause");
	return 0;
}
```

2. Kampionati i futbollit **15 pikë**

Në një kampionat futbolli ka *n* ekipe (supozojmë se numri *n* është çift). Gjatë një sezoni, secili ekip ndeshet me çdo ekip tjetër ekzaktësisht një herë. Sezoni konsiston në *n-1* turne. Cdo ekip luan një herë gjatë turnit. Eshtë mirë që një ekip të luajë ndeshjet e njëpasnjëshme në stadiume të ndryshme, një në shtëpi pra në stadiumin e tij dhe një jashtë, ne stadiumin e ekipit kundërshtar. Fatkeqësisht nuk është gjithmonë i mundur ndërtimi i një kalendari në mënyrë që një ekip mos të luajë dy herë në stadiumin/fushën e tij, ose dy ndeshje rresht jashtë fushe. Kur krijohet kalendari, numri i këtyre situatave duhet minimizuar (psh, nëse nje ekip luan jashte një herë, më pas katër ndeshje në shtëpi dhe më pas një jashtë, i numëron si tri situata).
Detyra juaj është të minimizoni numrin e situatave gjatë të cilave një ekip luan dy herë rresht në shtëpi ose jashtë dhe të ndërtoni një kalendar për gjithë sezonin.Cdo ndeshje luhet në një nga stadiumet e ekipeve – një ekip luan në stadiumin në shtëpi dhe një tjetër luan jashtë. Numri total i situatave në të cilin një ekip luan dy ndeshje rresht në shtëpi ose jashtë duhet të jetë minimal.

DETYRA
Shkruaj një program, që:
- Lexon numrin e ekipeve nga një input standard
- Llogarit numrin minimal total të situatave kur një ekip luan dy herë në fushën e tij ose jashtë dhe ndërton kalendarin
- Shkruan rezultatin në një output standard

INPUT
Rreshti i parë dhe i vetëm i inputit përmban një integer çift ($2 \le n \le 1000$) që është numri i ekipeve pjesëmarrëse

OUTPUT
Rreshti i parë i outputit duhet të përmbajë një integer – numri minimal i situatave ku nje ekip luan dy herë në shtëpi ose jashtë. *n-1* reshtat pasardhës duhet të përmbajnë kalendarin : rreshti *k+1* duhet të përmbajë pershkrimin e turnit të *k*.
Përshkrimi i turnit konsiston në *n* numra të ndryshëm $d_1,d_2,...,d_n$ nga {1,2,...,n} të ndarë nga 1 hapësirë boshe. Për i = 1,2,..., n/2 çifti d_{2i-1},d_{2i} tregon një ndeshje midis ekipeve d_{2i-1} dhe d_{2i}. Ekipi d_{2i-1} luan në shtëpi ndërsa ekipi d_{2i} luan jashtë.

SHEMBULL

Për inputin:	rezultati korrekt është:
4	2 1 2 3 4 4 1 2 3 1 3 4 2

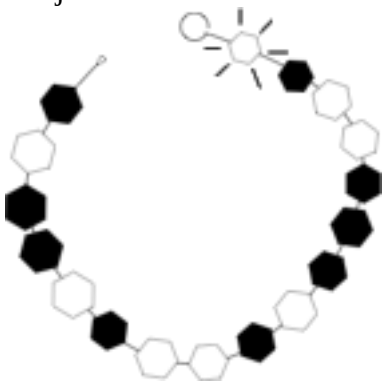
Zgjidhja:

```
#include <stdio.h>
#include <conio.h>
int  i, k, l, n;

/*Ky funksion merr nr. e turneve (k) dhe nr. e skuadrave (i),
me vlera midis 0 dhe n-2 dhe llogarit ekipin qe do te ndeshet me ekipin i ne turnin
k*/

int p(int k,int i)
{
  int j,d;
  j = (k - i) % (n - 1);
  if (j < 0)
```

```
    j = j + n - 1;
  if (i != j)
      d = j;
  else
   d = n - 1;
  return d;
}

/* Llogaritja e kalendarit */

void kalendari(int n)
{
  for (k = 0;k<=n - 2;k++)
  {
    l = (k + 1) / 2;
    for (i = 0;i<=n/2 - 1;i++)
    {
      if (k%2==1)
        printf("%d  %d",p(k,l + i) + 1,l + i + 1);
      else
        printf("%d  %d",l + i + 1,p(k,l + i) + 1);
      if (i < n / 2 - 1 )
        printf("  ");
    }
   printf("\n");
  }
}

int main()
{
  printf ("Jepni n:");
  scanf("%d",&n); //nr. i ekipeve
  printf("%d \n",n-2);
  kalendari(n);
  getch();
}
```

3. Varësja me perla **20 pikë**

Në një mal jetojnë dy grupe xhuxhash: xhuxhat e kuq dhe xhuxhat jeshil. Gjatë një ekspedite të përbashkët, një grup xhuxhash jeshil dhe të kuq gjejnë një varëse te përbërë nga perla xhami të pavlerë të bardha dhe të zeza me përjashtim të një diamanti në fund të varëses.



Të dy klanet e xhuxhave duan të zotërojnë diamantin. Xhuxhat vendosin ta zgjidhin situatën në mënyrë miqësore dhe luajnë një lojë :
Secilit prej *N* xhuxhave i caktohet një numër ID unik nga 1 në *N*. Secilit xhuxh i jepen dy lista publike të njohura me numrat ID të xhuxhave, një listë e bardhë dhe një listë e zezë. (Listat e xhuxhave mund të jenë të ndryshme nga njëra-tjetra). Secila listë mund të mbajë numra ID të xhuxhave jeshil dhe të kuq. Gjatë lojës varësja kalohet rreth e rrotull në bazë të këtyre rregullave : kur një xhuxh merr varësen, ai heq perlën e parë; nëse perla është e bardhë, ai ja jep pjesën e mbetur të varëses një xhuxhi që zgjedh ai vetë

nga lista e bardhë (që mund të jetë po vetë ai) ; nëse perla është e zezë, pjesa e mbetur i shkon një xhuxhi të zgjedhur nga ai vetë nga lista e zezë (që mund të jetë po ai vetë). Për të nisur lojën, xhuxhi që merr i pari varësen zgjidhet në mënyrë rastësore.
Në përfundim të lojës varësja do të përbëhet vetëm nga diamanti. Xhuxhi që merr varësen në këtë gjendje e mban atë për klanin e vetë dhe loja përfundon.
Shkruani një program që ndihmon xhuxhat jeshil të fitojnë diamantin. Përdor libraritë e përshkruara më poshtë. Mund të konsideroni sikur xhuxhat e kuq luajnë në mënyren më optimale.

Librari
Duhet të ndërtoni një librari (perla_lib.h ose perla_lib.pas) që përmban funksionet e mëposhtme:
getNext ( ) duhet të thirret kur është radha e xhuxhave të kuq. Kthen numrin ID të xhuxhit të cilit xhuxhi i kuq i dorëzon varësen.
setNext (d) duhet të thirret kur është radha e xhuxhave jeshil. Parametri d specifikon numrin ID të xhuxhit të cilit xhuxhi jeshil i dorëzon varësen.
finish ( ) duhet thirrur kur loja mbaron. Ai mbyll programin.

Supozoni që xhuxhat e kuq gjithmonë ja kalojnë varësen xhuxhit të parë në listën e caktuar.

Specifikime per programistët C/C++
Përdorni direktivën #include “perla_lib.h” për të përfshirë një librari që ofron këto funksione:
int getNext (void);
void setNext (int d);
void finish (void);

Specifikime për programistët Pascal
Përdorni statement uses perla_lib; për të përfshirë librarine që mundëson këto funksione:
function getNext: Integer;
procedure setNext (d : Integer);
procedure finish;

INPUT

Rreshti i parë i file-t input perla.in përman gjatësine fillestare L të varëses ($1 \leq L \leq 1000$), numrin N të xhuxhave ($1 \leq N \leq 1000$), si edhe numrin e ID F të xhuxhit që e merr i pari varësen ($1 \leq F \leq N$). Të kihet parasysh se ($1 \leq i \leq N$) për çdo numër ID xhuxhi i.
Rreshti i dytë përmban L karaktere që përshkruajnë varësen. Secili prej $L-1$ karaktereve të para është ose shkronja Z ose shkronja B. Z përfaqëson një perlë të zezë dhe B përfaqëson një perlë të bardhë. Karakteri i fundit është shkronja D që përfaqëson diamantin.
N rreshtat pasardhës përshkruajnë xhuxhat. Rreshti i i-të përshkruan xhuxhin me numër ID i. Secili prej këtyre rreshtave fillon me një numër që specifikon ngjyrën e xhuxhit: 0 për xhuxhin jeshil, 1 për xhuxhin e kuq. Më pas vjen gjatësia L_Z e listës së zezë të xhuxhit ($1 \leq L_Z \leq 20$), e ndjekur nga numrat ID të xhuxhit në listën e zezë. Kjo listë ndiqet nga gjatësia L_B e listës së bardhë të xhuxhit ($1 \leq L_B \leq 20$), e ndjekur nga L_B numrat ID të xhuxhit në listën e bardhë.

OUTPUT

Si output duhet të shfaqni të gjitha thirrjet e librarive.

SHEMBULL

perla.in	Thirrjet e librarive
6 4 2 ZBBZZD 0 1 2 1 4 0 2 1 3 1 1 1 1 4 1 4 1 2 2 3 1 1	setNext (1) setNext (4) getNext ( ) -> 1 setNext (2) setNext (1) finish ( )

Zgjidhja:

```
#include <conio.h>
#include <stdlib.h>
#include <fstream>
#include <vector>
#include "perla_lib.h"

using namespace std;

#define MAXDWARFES 1000
#define MAXLEN 1000

typedef vector<int> VI;

struct dwarfes {
	int c;
	VI list[2];
}d[MAXDWARFES];

char pos[MAXLEN][MAXDWARFES];
int mov[MAXLEN][MAXDWARFES];

ifstream in("perla.in");
ofstream outputT("perla.OUT");

int main() {
	int l,n,f;
	in >> l >> n >> f;
	f--;

	int type[MAXLEN];
	for (int i=0; i<l; i++) {
		char c;
		while(in >> c)
			if (c == 'D' || c == 'B' || c == 'Z')
				break;
		type[i] = (c == 'B');
	}

	for (int i=0; i<n; i++) {
		in >> d[i].c;
		pos[l-1][i] = d[i].c;
		for (int j=0; j<2; j++) {
			int len;
			in >> len;
			while(len--) {
				int temp;
				in >> temp;
				d[i].list[j].push_back(temp-1);
			}
		}
	}
	for (int p=l-2; p>=0; p--) {
```

```
            int c = type[p];
            for (int i=0; i<n; i++) {
                  pos[p][i] = 1-d[i].c;
                  for (VI::iterator it=d[i].list[c].begin(); it<d[i].list[c].end();
it++) {
                        mov[p][i] = *it;
                        if (pos[p+1][*it] == d[i].c) { /*levizja fituese */
                              pos[p][i] = d[i].c;
                              break;
                        }
                  }
            }
      }
      for (int p=0; p<l-1; p++) {
            if (d[f].c == 1)
                  f = getNext()-1;
            else {
                  f = mov[p][f];
                  setNext(f+1);
            }
      }
      finish();
      getch();
    system("PAUSE");
      return(0);
}

//libraria perla_lib.h
#include <stdio.h>
#include <conio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>

#define LIBMAXLEN 1000
#define LIBMAXDWARFES 1000

char LIBinitialized = 0;

struct LIBlist {
  struct LIBlist *next;
  int i;
}*LIBp;

struct LIBdwarfes {
  int c;
  struct LIBlist *p[2];
}LIBd[LIBMAXDWARFES];

int LIBtype[LIBMAXLEN],LIBf,LIBl,LIBn,LIBact;

FILE *LIBout;

void LIBreport(int EXIT, char *fmt, ...) {
  va_list ap;
  va_start( ap, fmt );

  vfprintf( LIBout, fmt, ap );
  if ( fmt[ strlen(fmt)-1 ] != '\n' ) fprintf(LIBout,"\n");
  if (EXIT){
    exit (0);
  }
}

void init() {
  int n,l,i,j,p;
```

```
  FILE *in = fopen("perla.in","r");
  LIBout = stdout;

  if (!in) LIBreport(1, "File perla.in nuk mund te hapet.");
  if (fscanf(in,"%d %d %d",&l,&n,&LIBf)!=3) LIBreport(1, "Gabim ne skedarin
perla.in");

  LIBinitialized = 1;
  LIBact = 0;
  LIBn = n;
  LIBl = l;
  LIBf--;

  LIBreport(0, "init()");
  for (i=0; i<l; i++){
    char c;
    if (fscanf(in," %c",&c)!=1 || c!='Z' && c!='B' && (i<l-1 || c!='D'))
      LIBreport(1,"Gabim ne skedarin perla.in");
    LIBtype[i] = (c == 'B');
  }

  for (i=0; i<n; i++) {
    if (fscanf(in,"%d",&LIBd[i].c)!=1) LIBreport(1,"Gabim ne skedarin perla.in");
    for (j=0; j<2; j++) {
      int len;
      LIBd[i].p[j] = NULL;
      if (fscanf(in,"%d",&len)!=1) LIBreport(1,"Gabim ne skedarin perla.in");
      while(len--) {
      int temp;
      if (fscanf(in,"%d",&temp)!=1) LIBreport(1,"Gabim ne skedarin perla.in");
      LIBp = (struct LIBlist *)malloc(sizeof(struct LIBlist));
      LIBp->i = temp-1;
      LIBp->next = LIBd[i].p[j];
      LIBd[i].p[j] = LIBp;
      }
    }
  }
  fclose(in);

}

void finish() {
  if (!LIBinitialized) init();
  LIBreport(0, "finish()");
  int i;
  if (LIBact!=LIBl-1) LIBreport(1,"Thirret funx.finish() pa arritur fundin e
lojes");
  if (LIBd[LIBf].c==1)
    LIBreport(1,"Xhuxhat e kuq fituan");
    LIBreport(1,"Xhuxhat jeshile fituan");
}

int getNext() {
  if (!LIBinitialized) init();
  if (LIBact>=LIBl-1) LIBreport(1,"Shume thirrje per getNext()/setNext()");
  if (LIBd[LIBf].c==0)LIBreport(1,"U thirr funx. getNext(),por xhuxhi %d eshte
jeshil", LIBf+1);
  LIBf = LIBd[LIBf].p[LIBtype [LIBact]]->i;
  LIBreport(0, "getNext() -> %d", LIBf+1);
  LIBact++;
  return LIBf+1;
}

void setNext(int dwarfes) {
  if (!LIBinitialized) init();
  if (LIBact>=LIBl-1) LIBreport(1,"Shume thirrje per getNext()/setNext()");
```

```
  if (LIBd[LIBf].c==1)LIBreport(1," U thirr funx.setNext(),por xhuxhi %d eshte i
kuq", LIBf+1);
  LIBp = LIBd[LIBf].p[LIBtype[LIBact]];
  while(LIBp!=NULL) {
    if (LIBp->i == dwarfes-1) {
      LIBf = LIBp->i;
      break;
    }
    LIBp = LIBp->next;
  }
  if (LIBp == NULL) {
    LIBreport(1, "nuk ka xhuxh me nr. %d ne listen e xhuxhit %d", dwarfes, LIBf+1);
  }
  LIBreport(0, "setNext(%d)", dwarfes);
  LIBact++;
}
```