eFólio R - resolução (2016-2017)

```
#include <stdio.h>

unsigned int randaux()
{
  static long seed=1;
  return(((seed = seed * 214013L + 2531011L) >> 16) & 0x7fff);
}
```

Alínea A ----------

```
#include <string.h>
#include <stdlib.h>

int main() {
    int K, W, i;
    printf("K: ");
    scanf("%d",&K);
    printf("W: ");
    scanf("%d",&W);

    for(i=0;i<K;i++)
        printf("%c",'A'+randaux()%W);
}
```

```
6
4
K: W: BDCABA

8
4
K: W: BDCABACC

6
12
K: W: FLKEFE

12
4
K: W: BDCABACCCABB


8
8
K: W: BDGEBEGG

10
10
K: W: BHEAJEIICE

12
3
K: W: CCBBCBAABCCC

6
2
```

```
K: W: BBAABA

12
2
K: W: BBAABAAAAABB


4
10
K: W: BHEA
```

Alínea B ----------

```
#include <string.h>
#include <stdlib.h>

#define MAXSTR 40

void GerarChave(char *chave, int K, int W) {
    int i;
    for(i=0;i<K;i++)
        chave[i]='A'+randaux()%W;
    chave[K]=0;
}

void AnaliseAposta(char *chave, char *aposta, int K, int W, int *pretas, int
*brancas) {
    int i;
    char dupChave[MAXSTR], dupAposta[MAXSTR], *pt;
    *pretas=0;
    *brancas=0;
    // atualizar as pretas
    for(i=0;i<K;i++) {
        if(chave[i]==aposta[i]) {
            (*pretas)++;
            dupChave[i]=dupAposta[i]=' '; // já foi utilizada, não copiar
        } else {
            dupChave[i]=chave[i]; // pode existir no local errado
            dupAposta[i]=aposta[i];
        }
    }
    // calcular as brancas
    dupAposta[K]=0;
    dupChave[K]=0;
    for(i=0;i<K;i++)
        if(dupChave[i]!=' ') {
            pt=strchr(dupAposta,dupChave[i]);
            if(pt!=NULL) {
                // certo no local errado
                (*brancas)++;
                *pt=' '; // assegurar que não é contabilizada novamente
            }
        }
}

int main() {
    int K, W, i, brancas,pretas;
    char chave[MAXSTR], aposta[MAXSTR];
    printf("K: ");
    scanf("%d",&K);
```

```
    printf("W: ");
    scanf("%d",&W);

    GerarChave(chave, K, W);

    printf("Aposta: ");
    gets(aposta);
    gets(aposta);

    AnaliseAposta(chave,aposta,K,W,&pretas,&brancas);

    printf("(chave: %s) p%d b%d",chave,pretas,brancas);
}
```

```
6
4
ABCDAB

K: W: Aposta: (chave: BDCABA) p1 b5

6
4
ABCAAA

K: W: Aposta: (chave: BDCABA) p3 b1


6
4
ADDDDA

K: W: Aposta: (chave: BDCABA) p2 b1


8
4
BACCBDCA

K: W: Aposta: (chave: BDCABACC) p4 b4

8
4
BBBBBDCA

K: W: Aposta: (chave: BDCABACC) p3 b2

6
12
BBBBBD

K: W: Aposta: (chave: FLKEFE) p0 b0

6
12
EFEFEF

K: W: Aposta: (chave: FLKEFE) p0 b4


12
4
BDCAABDDDABB
```

```
K: W: Aposta: (chave: BDCABACCCABB) p7 b2


12
4
ABABABABABAB

K: W: Aposta: (chave: BDCABACCCABB) p1 b6

12
4
BBABABADABDD

K: W: Aposta: (chave: BDCABACCCABB) p1 b7
```

Alínea C ----------

```
#include <string.h>
#include <stdlib.h>

#define MAXSTR 40

void GerarChave(char *chave, int K, int W) {
    int i;
    for(i=0;i<K;i++)
        chave[i]='A'+randaux()%W;
    chave[K]=0;
}

void AnaliseAposta(char *chave, char *aposta, int K, int W, int *pretas, int
*brancas) {
    int i;
    char dupChave[MAXSTR], dupAposta[MAXSTR], *pt;
    *pretas=0;
    *brancas=0;
    // atualizar as pretas
    for(i=0;i<K;i++) {
        if(chave[i]==aposta[i]) {
            (*pretas)++;
            dupChave[i]=dupAposta[i]=' '; // já foi utilizada, não copiar
        } else {
            dupChave[i]=chave[i]; // pode existir no local errado
            dupAposta[i]=aposta[i];
        }
    }
    // calcular as brancas
    dupAposta[K]=0;
    dupChave[K]=0;
    for(i=0;i<K;i++)
        if(dupChave[i]!=' ') {
            pt=strchr(dupAposta,dupChave[i]);
            if(pt!=NULL) {
                // certo no local errado
                (*brancas)++;
                *pt=' '; // assegurar que não é contabilizada novamente
            }
        }
}
```

```
int ApostaCompativel(char aposta[MAXSTR][MAXSTR],
                             int *pretas, int *brancas,
                             int nApostas, int K, int W) {
    int resultado=0,i, lPretas, lBrancas; // contador
    // processar a aposta mais recente, e ver se é compatível com as anteriores
    for(i=0;i<nApostas;i++) {
        AnaliseAposta(aposta[i],aposta[nApostas],K,W,&lPretas,&lBrancas);
        if(lPretas!=pretas[i] || lBrancas!=brancas[i]) // brancas e pretas têm de
ser exaxtamente iguais
            return i;
    }
    return -1;
}

int main() {
    int K, W, i, nApostas=-1;
    int pretas[MAXSTR], brancas[MAXSTR];
    char chave[MAXSTR], aposta[MAXSTR][MAXSTR];
    printf("K: ");
    scanf("%d",&K);
    printf("W: ");
    scanf("%d",&W);
    gets(aposta[0]); // remover o fim de linha do scanf

    GerarChave(chave, K, W);
    do {
        nApostas++;
        printf("\nAposta %d: ", nApostas);
        gets(aposta[nApostas]);
        if(strlen(aposta[nApostas])<K || nApostas>MAXSTR)
            break;

        AnaliseAposta(chave,aposta[nApostas],K,W,pretas+nApostas,brancas+nApostas);

        printf("(chave: %s) p%d b%d Incompativel com: %d",
              chave,pretas[nApostas],brancas[nApostas],
              ApostaCompativel(aposta,pretas,brancas,nApostas,K,W));
    } while(strcmp(aposta[nApostas],chave)!=0);
}
```

```
6
4
ABCDAB
ABCAAA
ADDDDA

K: W:
Aposta 0: (chave: BDCABA) p1 b5 Incompativel com: -1
Aposta 1: (chave: BDCABA) p3 b1 Incompativel com: 0
Aposta 2: (chave: BDCABA) p2 b1 Incompativel com: 0
Aposta 3:


6
4
ABCDAB
DACABB
BDCABA

K: W:
Aposta 0: (chave: BDCABA) p1 b5 Incompativel com: -1
Aposta 1: (chave: BDCABA) p3 b3 Incompativel com: 0
```

```
Aposta 2: (chave: BDCABA) p6 b0 Incompativel com: -1


6
4
CCCABB
ACAACB
CACDDB
BDCABA

K: W:
Aposta 0: (chave: BDCABA) p3 b1 Incompativel com: -1
Aposta 1: (chave: BDCABA) p1 b3 Incompativel com: -1
Aposta 2: (chave: BDCABA) p1 b3 Incompativel com: -1
Aposta 3: (chave: BDCABA) p6 b0 Incompativel com: -1



8
4
CABBBDBD
ADABABAB
DCDDAABA
ABBCCCDA
BDCABACC

K: W:
Aposta 0: (chave: BDCABACC) p1 b4 Incompativel com: -1
Aposta 1: (chave: BDCABACC) p1 b4 Incompativel com: -1
Aposta 2: (chave: BDCABACC) p1 b4 Incompativel com: -1
Aposta 3: (chave: BDCABACC) p0 b8 Incompativel com: -1
Aposta 4: (chave: BDCABACC) p8 b0 Incompativel com: -1

12
2
BBBBBABABAAB
BBBABBAAAAAA
BBBBAAAAAABA
BABBBAABAAAA
BBABABBAAAAA
BBAABAAAAABB

K: W:
Aposta 0: (chave: BBAABAAAAABB) p7 b2 Incompativel com: -1
Aposta 1: (chave: BBAABAAAAABB) p8 b4 Incompativel com: -1
Aposta 2: (chave: BBAABAAAAABB) p8 b4 Incompativel com: -1
Aposta 3: (chave: BBAABAAAAABB) p6 b6 Incompativel com: -1
Aposta 4: (chave: BBAABAAAAABB) p6 b6 Incompativel com: -1
Aposta 5: (chave: BBAABAAAAABB) p12 b0 Incompativel com: -1


4
6
FEAA
DACA
FEBB
FFEE
K: W:
Aposta 0: (chave: FFEE) p1 b1 Incompativel com: -1
Aposta 1: (chave: FFEE) p0 b0 Incompativel com: -1
Aposta 2: (chave: FFEE) p1 b1 Incompativel com: 0
Aposta 3: (chave: FFEE) p4 b0 Incompativel com: -1
```

```
5
5
EDDCE
EAEDB
BCADE
EDDEC
K: W:
Aposta 0: (chave: BCEAE) p1 b2 Incompativel com: -1
Aposta 1: (chave: BCEAE) p1 b3 Incompativel com: -1
Aposta 2: (chave: BCEAE) p3 b1 Incompativel com: -1
Aposta 3: (chave: BCEAE) p0 b3 Incompativel com: 0
Aposta 4:


4
12
FEGG
KGJG
ACGF
LEFK
FLKE

K: W:
Aposta 0: (chave: FLKE) p1 b1 Incompativel com: -1
Aposta 1: (chave: FLKE) p0 b1 Incompativel com: -1
Aposta 2: (chave: FLKE) p0 b1 Incompativel com: -1
Aposta 3: (chave: FLKE) p0 b4 Incompativel com: -1
Aposta 4: (chave: FLKE) p4 b0 Incompativel com: -1


4
24
REGG
WGVG
TRGK
GEEI
RWRE
RNWE
RLWE

K: W:
Aposta 0: (chave: RLWE) p1 b1 Incompativel com: -1
Aposta 1: (chave: RLWE) p0 b1 Incompativel com: -1
Aposta 2: (chave: RLWE) p0 b1 Incompativel com: -1
Aposta 3: (chave: RLWE) p0 b1 Incompativel com: -1
Aposta 4: (chave: RLWE) p2 b1 Incompativel com: -1
Aposta 5: (chave: RLWE) p3 b0 Incompativel com: -1
Aposta 6: (chave: RLWE) p4 b0 Incompativel com: -1


10
2
BBBBBBBABA
BBABAAAAAA
BAABBAAAAA
BAABAABAAA
BBAABAAAAA

K: W:
Aposta 0: (chave: BBAABAAAAA) p5 b0 Incompativel com: -1
Aposta 1: (chave: BBAABAAAAA) p8 b2 Incompativel com: -1
Aposta 2: (chave: BBAABAAAAA) p8 b2 Incompativel com: -1
Aposta 3: (chave: BBAABAAAAA) p6 b4 Incompativel com: -1
```

```
Aposta 4: (chave: BBAABAAAAA) p10 b0 Incompativel com: -1
```

Alínea D ----------

```
#include <string.h>
#include <stdlib.h>

#define MAXSTR 40

void GerarChave(char *chave, int K, int W) {
    int i;
    for(i=0;i<K;i++)
        chave[i]='A'+randaux()%W;
    chave[K]=0;
}

void AnaliseAposta(char *chave, char *aposta, int K, int W, int *pretas, int
*brancas) {
    int i;
    char dupChave[MAXSTR], dupAposta[MAXSTR], *pt;
    *pretas=0;
    *brancas=0;
    // atualizar as pretas
    for(i=0;i<K;i++) {
        if(chave[i]==aposta[i]) {
            (*pretas)++;
            dupChave[i]=dupAposta[i]=' '; // já foi utilizada, não copiar
        } else {
            dupChave[i]=chave[i]; // pode existir no local errado
            dupAposta[i]=aposta[i];
        }
    }
    // calcular as brancas
    dupAposta[K]=0;
    dupChave[K]=0;
    for(i=0;i<K;i++)
        if(dupChave[i]!=' ') {
            pt=strchr(dupAposta,dupChave[i]);
            if(pt!=NULL) {
                // certo no local errado
                (*brancas)++;
                *pt=' '; // assegurar que não é contabilizada novamente
            }
        }
}

int ApostaCompativel(char aposta[MAXSTR][MAXSTR],
                             int *pretas, int *brancas,
                             int nApostas, int K, int W) {
    int resultado=0,i, lPretas, lBrancas; // contador
    // processar a aposta mais recente, e ver se é compatível com as anteriores
    for(i=0;i<nApostas;i++) {
        AnaliseAposta(aposta[i],aposta[nApostas],K,W,&lPretas,&lBrancas);
        if(lPretas!=pretas[i] || lBrancas!=brancas[i]) // brancas e pretas têm de
ser exaxtamente iguais
            return i;
    }
    return -1;
}
```

```
void GerarAposta(char aposta[MAXSTR][MAXSTR],
    int *pretas, int *brancas,
    int nApostas, int K, int W) {
    int i,j,l,tentativas=0, limite;
    char restoAposta[MAXSTR];
    if(nApostas<0) {
        GerarChave(aposta[0],K,W);
        return;
    }
    do {
        tentativas++;
        // inicializar a aposta
        for(i=0;i<K;i++) {
            aposta[nApostas+1][i]=' ';
            restoAposta[i]=aposta[nApostas][i];
        }
        // escolher o número de pretas da última aposta, e copiar
        for(i=0;i<pretas[nApostas];) {
            j=randaux()%K;
            if(aposta[nApostas+1][j]==' ') {
                aposta[nApostas+1][j]=aposta[nApostas][j];
                restoAposta[j]=' ';
                i++;
            }
        }
        // para as brancas, sortear casa de origem e destino, desde que seja
diferente
        for(i=0, limite=0;i<brancas[nApostas] && limite<1000;limite++) {
            j=randaux()%K; // de onde vem
            l=randaux()%K; // para onde vai
            if(j!=l && aposta[nApostas+1][l]==' ' && restoAposta[j]!=' ' &&
              aposta[nApostas][l]!=aposta[nApostas][j]) {
                aposta[nApostas+1][l]=aposta[nApostas][j];
                restoAposta[j]=' ';
                i++;
            }
        }
        if(limite>=1000)
            continue;
        // sortear as restantes casas, desde que não incluam letras do resto da
aposta
        restoAposta[K]=0;
        for(i=0, limite=0;i<K-pretas[nApostas]-brancas[nApostas] &&
limite<1000;limite++) {
            j='A'+randaux()%W; // letra aleatória
            if(strchr(restoAposta,j)==NULL) {
                // letra não está no resto da aposta
                for(l=0;l<K;l++)
                    if(aposta[nApostas+1][l]==' ') {
                        aposta[nApostas+1][l]=j;
                        i++;
                        break;
                    }
            }
        }
        if(limite>=1000)
            continue;
    } while(limite>=1000 ||
            tentativas<1000 && ApostaCompativel(aposta, pretas, brancas,
nApostas+1, K,W)>=0);
}
```

```
int main() {
    int K, W, i, nApostas=-1;
    int pretas[MAXSTR], brancas[MAXSTR];
    char chave[MAXSTR], aposta[MAXSTR][MAXSTR];
    printf("K: ");
    scanf("%d",&K);
    printf("W: ");
    scanf("%d",&W);

    GerarChave(chave, K, W);
    do {
        nApostas++;
        printf("\nAposta %d: ", nApostas);
        GerarAposta(aposta,pretas,brancas,nApostas-1,K,W);
        aposta[nApostas][K]=0;
        printf("%s ",aposta[nApostas]);

        AnaliseAposta(chave,aposta[nApostas],K,W,pretas+nApostas,brancas+nApostas);

        printf("(chave: %s) p%d b%d incompativel: %d",
              chave,pretas[nApostas],brancas[nApostas],
              ApostaCompativel(aposta,pretas,brancas,nApostas,K,W));
    } while(nApostas<10 && strcmp(aposta[nApostas],chave)!=0);
}
```

```
3
3

K: W:
Aposta 0: BCB (chave: CCB) p2 b0 incompativel: -1
Aposta 1: BCC (chave: CCB) p1 b2 incompativel: -1
Aposta 2: CCB (chave: CCB) p3 b0 incompativel: -1

4
4

K: W:
Aposta 0: BACC (chave: BDCA) p2 b1 incompativel: -1
Aposta 1: BBCA (chave: BDCA) p3 b0 incompativel: -1
Aposta 2: BDCA (chave: BDCA) p4 b0 incompativel: -1

5
5

K: W:
Aposta 0: EDDCE (chave: BCEAE) p1 b2 incompativel: -1
Aposta 1: EAEDB (chave: BCEAE) p1 b3 incompativel: -1
Aposta 2: BCADE (chave: BCEAE) p3 b1 incompativel: -1
Aposta 3: BCEAE (chave: BCEAE) p5 b0 incompativel: -1


4
6

K: W:
Aposta 0: FEAA (chave: FFEE) p1 b1 incompativel: -1
Aposta 1: DACA (chave: FFEE) p0 b0 incompativel: -1
Aposta 2: FFFE (chave: FFEE) p3 b0 incompativel: -1
Aposta 3: FFEE (chave: FFEE) p4 b0 incompativel: -1

6
4
```

```
K: W:
Aposta 0: CCCABB (chave: BDCABA) p3 b1 incompativel: -1
Aposta 1: ACAACB (chave: BDCABA) p1 b3 incompativel: -1
Aposta 2: CACDDB (chave: BDCABA) p1 b3 incompativel: -1
Aposta 3: BDCABA (chave: BDCABA) p6 b0 incompativel: -1


8
4

K: W:
Aposta 0: CABBBDBD (chave: BDCABACC) p1 b4 incompativel: -1
Aposta 1: ADABABAB (chave: BDCABACC) p1 b4 incompativel: -1
Aposta 2: DCDDAABA (chave: BDCABACC) p1 b4 incompativel: -1
Aposta 3: ABBCCCDA (chave: BDCABACC) p0 b8 incompativel: -1
Aposta 4: BDCABACC (chave: BDCABACC) p8 b0 incompativel: -1

4
12

K: W:
Aposta 0: FEGG (chave: FLKE) p1 b1 incompativel: -1
Aposta 1: KGJG (chave: FLKE) p0 b1 incompativel: -1
Aposta 2: ACGF (chave: FLKE) p0 b1 incompativel: -1
Aposta 3: LEFK (chave: FLKE) p0 b4 incompativel: -1
Aposta 4: FLKE (chave: FLKE) p4 b0 incompativel: -1


4
24
K: W:
Aposta 0: REGG (chave: RLWE) p1 b1 incompativel: -1
Aposta 1: WGVG (chave: RLWE) p0 b1 incompativel: -1
Aposta 2: TRGK (chave: RLWE) p0 b1 incompativel: -1
Aposta 3: GEEI (chave: RLWE) p0 b1 incompativel: -1
Aposta 4: RWRE (chave: RLWE) p2 b1 incompativel: -1
Aposta 5: RNWE (chave: RLWE) p3 b0 incompativel: -1
Aposta 6: RLWE (chave: RLWE) p4 b0 incompativel: -1

10
2

K: W:
Aposta 0: BBBBBBBABA (chave: BBAABAAAAA) p5 b0 incompativel: -1
Aposta 1: BBABAAAAAA (chave: BBAABAAAAA) p8 b2 incompativel: -1
Aposta 2: BAABBAAAAA (chave: BBAABAAAAA) p8 b2 incompativel: -1
Aposta 3: BAABAABAAA (chave: BBAABAAAAA) p6 b4 incompativel: -1
Aposta 4: BBAABAAAAA (chave: BBAABAAAAA) p10 b0 incompativel: -1

12
2

K: W:
Aposta 0: BBBBBABABAAB (chave: BBAABAAAAABB) p7 b2 incompativel: -1
Aposta 1: BBBABBAAAAAA (chave: BBAABAAAAABB) p8 b4 incompativel: -1
Aposta 2: BBBBAAAAAABA (chave: BBAABAAAAABB) p8 b4 incompativel: -1
Aposta 3: BABBBAABAAAA (chave: BBAABAAAAABB) p6 b6 incompativel: -1
Aposta 4: BBABABBAAAAA (chave: BBAABAAAAABB) p6 b6 incompativel: -1
Aposta 5: BBAABAAAAABB (chave: BBAABAAAAABB) p12 b0 incompativel: -1

8
3
```

```
K: W:
Aposta 0: BCCCBABC (chave: CCBBCBAA) p1 b6 incompativel: -1
Aposta 1: CABCCBAB (chave: CCBBCBAA) p5 b3 incompativel: -1
Aposta 2: ABBCCCAB (chave: CCBBCBAA) p3 b5 incompativel: -1
Aposta 3: CCABCBAB (chave: CCBBCBAA) p6 b2 incompativel: -1
Aposta 4: CCBBCBAA (chave: CCBBCBAA) p8 b0 incompativel: -1

10
3

K: W:
Aposta 0: CCBABCBCAA (chave: CCBBCBAABC) p3 b6 incompativel: -1
Aposta 1: BCCCCAABAA (chave: CCBBCBAABC) p3 b5 incompativel: -1
Aposta 2: BBBBAACCCA (chave: CCBBCBAABC) p2 b7 incompativel: -1
Aposta 3: CCBBCBAABC (chave: CCBBCBAABC) p10 b0 incompativel: -1


12
4
K: W:
Aposta 0: BDBDDCDADACB (chave: BDCABACCCABB) p4 b4 incompativel: -1
Aposta 1: CCDDDBCCACCB (chave: BDCABACCCABB) p3 b5 incompativel: -1
Aposta 2: AACCDBDAAADC (chave: BDCABACCCABB) p2 b6 incompativel: -1
Aposta 3: BBDCAAABBDBC (chave: BDCABACCCABB) p3 b7 incompativel: 0
Aposta 4: ADBDABABDCAB (chave: BDCABACCCABB) p2 b7 incompativel: 0
Aposta 5: BABADABDBBBB (chave: BDCABACCCABB) p5 b3 incompativel: 0
Aposta 6: DADCCABABDCB (chave: BDCABACCCABB) p2 b8 incompativel: 0
Aposta 7: DCBBBBABCDAD (chave: BDCABACCCABB) p2 b7 incompativel: 0
Aposta 8: BACBACBCACDD (chave: BDCABACCCABB) p3 b8 incompativel: 0
Aposta 9: CCACBCABABBD (chave: BDCABACCCABB) p2 b10 incompativel: 0
Aposta 10: BACDBBCCAACB (chave: BDCABACCCABB) p7 b5 incompativel: 0
```