# eFólio A - resolução (2016-2017)

## Alínea A

```
    /* Mostre um tabuleiro de N linhas e M colunas */
    int i,j;
    for(i=0;i<N;i++) {
        for(j=0;j<M;j++)
            printf("+");
        printf("\n");
    }
```

## Alínea B

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

#define MAXSIZE 40

void InicializarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
            tabuleiro[i][j]='+';
}

void ColocarMina(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int casa=-1;
    while(casa<0) {
        casa=randaux()%(N*M); // casa aleatória
        //printf("Casa aleatoria: %d (linha %d coluna %d)\n",casa,casa/M,casa%M);
        if(tabuleiro[casa/M][casa%M]=='+')
            tabuleiro[casa/M][casa%M]='#';
        else
            casa=-1; // já tem mina, procurar outra casa
    }
}

void MostrarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++) {
        for(j=0;j<M;j++)
            printf("%c",tabuleiro[i][j]);
        printf("\n");
    }
}

int main() {
    int N, M, K;
    char tabuleiro[MAXSIZE][MAXSIZE];
    printf("N: ");
    scanf("%d",&N);
    printf("M: ");
    scanf("%d",&M);
    printf("K: ");
    scanf("%d",&K);
```

```
    printf("\n");

    /* posicionar as minas e mostrar o mapa com as minas */
    InicializarTabuleiro(tabuleiro,N,M);
    for(int i=0;i<K;i++)
        ColocarMina(tabuleiro,N,M);
    MostrarTabuleiro(tabuleiro,N,M);
}
```

## Alínea C

```
#define MAXSIZE 40

void InicializarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
            tabuleiro[i][j]='+';
}

void ColocarMina(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int casa=-1;
    while(casa<0) {
        casa=randaux()%(N*M); // casa aleatória
        //printf("Casa aleatoria: %d (linha %d coluna %d)\n",casa,casa/M,casa%M);
        if(tabuleiro[casa/M][casa%M]=='+')
            tabuleiro[casa/M][casa%M]='#';
        else
            casa=-1; // já tem mina, procurar outra casa
    }
}

void MostrarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++) {
        for(j=0;j<M;j++)
            printf("%c",tabuleiro[i][j]);
        printf("\n");
    }
}

int ExplorarCasa(int casa, char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int resultado=0, linha=casa/M, coluna=casa%M, i, j;
    // explorando uma casa minada?
    if(tabuleiro[linha][coluna]=='#') {
        tabuleiro[linha][coluna]='X'; // fim do jogo
        return -1;
    }
    // somar todas as minas nas casas adjacentes
    for(i=-1;i<2;i++)
        for(j=-1;j<2;j++)
            if(linha+i>=0 && linha+i<N &&
              coluna+j>=0 && coluna+j<M &&
              (i!=0 || j!=0) && tabuleiro[linha+i][coluna+j]=='#')
                resultado++;
    // colocar marca no tabuleiro
    tabuleiro[linha][coluna]='0'+resultado;
    return resultado;
}
```

```
int main() {
    int N, M, K, i, casa;
    char tabuleiro[MAXSIZE][MAXSIZE];
    printf("N: ");
    scanf("%d",&N);
    printf("M: ");
    scanf("%d",&M);
    printf("K: ");
    scanf("%d",&K);
    printf("\n");

    /* posicionar as minas */
    InicializarTabuleiro(tabuleiro,N,M);
    for(int i=0;i<K;i++)
        ColocarMina(tabuleiro,N,M);

    /* ler casas e explorar */
    for(i=0;i<N*M-K;i++) {
        scanf("%d",&casa);
        if(casa<0 || casa>=N*M) // casa inválida, terminar
            break;
        if(ExplorarCasa(casa,tabuleiro,N,M)<0)
            break;
    }

    /* posição final */
    MostrarTabuleiro(tabuleiro,N,M);
}
```

## Alínea D

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

#define MAXSIZE 40

void InicializarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
            tabuleiro[i][j]='+';
}

void ColocarMina(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int casa=-1;
    while(casa<0) {
        casa=randaux()%(N*M); // casa aleatória
        //printf("Casa aleatoria: %d (linha %d coluna %d)\n",casa,casa/M,casa%M);
        if(tabuleiro[casa/M][casa%M]=='+')
            tabuleiro[casa/M][casa%M]='#';
        else
            casa=-1; // já tem mina, procurar outra casa
    }
}

void MostrarTabuleiro(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int i,j;
    for(i=0;i<N;i++) {
        for(j=0;j<M;j++)
            printf("%c",tabuleiro[i][j]);
```

```
        printf("\n");
    }
}

int TemMina(char valor) {
    if(valor=='#' || valor=='x' || valor=='X')
        return 1;
    return 0;
}

int ExplorarCasa(int casa, char tabuleiro[MAXSIZE][MAXSIZE], int N, int M) {
    int resultado=0, linha=casa/M, coluna=casa%M, i, j;
    // explorando uma casa minada?
    if(tabuleiro[linha][coluna]=='#') {
        tabuleiro[linha][coluna]='X'; // fim do jogo
        return -1;
    }
    // somar todas as minas nas casas adjacentes
    for(i=-1;i<2;i++)
        for(j=-1;j<2;j++)
            if(linha+i>=0 && linha+i<N &&
              coluna+j>=0 && coluna+j<M &&
              (i!=0 || j!=0) && TemMina(tabuleiro[linha+i][coluna+j]))
                resultado++;
    // colocar marca no tabuleiro
    tabuleiro[linha][coluna]='0'+resultado;
    return resultado;
}

int NaoExplorado(char valor) {
    if(valor=='+' || valor=='#')
        return 1;
    return 0;
}

int Analisar(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M, int linha, int coluna)
{
    int i,j, resultado=0, W, minas, naoExploradas;
    // Caso exista uma casa explorada com 0 minas adjacentes,
    //   explorar todas as casas adjacentes não exploradas;
    if(tabuleiro[linha][coluna]=='0') {
        // ver casas adjacentes não exploradas
        for(i=-1;i<2;i++)
            for(j=-1;j<2;j++)
                if(linha+i>=0 && linha+i<N &&
                  coluna+j>=0 && coluna+j<M &&
                  (i!=0 || j!=0) && NaoExplorado(tabuleiro[linha+i][coluna+j])) {
                    ExplorarCasa((linha+i)*M+coluna+j,tabuleiro,N,M);
                    resultado++;
                }
    }
    // Caso exista uma casa explorada com W minas adjacentes,
    //   existindo W casas marcadas como tendo minas, explorar as restantes casas
adjacentes;
    // Caso exista uma casa explorada com W minas adjacentes,
    //   existindo W casas não exploradas ou marcadas, marcar todas essas casas
como tendo minas.
    W=tabuleiro[linha][coluna]-'0';
    if(W>=1 && W<=8) {
        // situação anterior verificada, calcular o número de casas marcadas como
tendo minas
        minas=0;
        naoExploradas=0;
```

```
        for(i=-1;i<2;i++)
            for(j=-1;j<2;j++)
                if(linha+i>=0 && linha+i<N &&
                  coluna+j>=0 && coluna+j<M &&
                  (i!=0 || j!=0)) {
                    if(tabuleiro[linha+i][coluna+j]=='x')
                        minas++;
                    else if(NaoExplorado(tabuleiro[linha+i][coluna+j]))
                        naoExploradas++;
                 }
        if(W==minas) {
            // todas as minas descobertas, explorar as restantes casas
            for(i=-1;i<2;i++)
                for(j=-1;j<2;j++)
                    if(linha+i>=0 && linha+i<N &&
                      coluna+j>=0 && coluna+j<M &&
                      (i!=0 || j!=0) && NaoExplorado(tabuleiro[linha+i][coluna+j]))
{
                        ExplorarCasa((linha+i)*M+coluna+j,tabuleiro,N,M);
                        resultado++;
                    }


        } else if(W==naoExploradas+minas) {
            // todas as casas não exploradas são minas, marcar
            for(i=-1;i<2;i++)
                for(j=-1;j<2;j++)
                    if(linha+i>=0 && linha+i<N &&
                      coluna+j>=0 && coluna+j<M &&
                      (i!=0 || j!=0) && NaoExplorado(tabuleiro[linha+i][coluna+j]))
{
                        tabuleiro[linha+i][coluna+j]='x';
                        resultado++;
                    }
        }
    }
    return resultado;
}

int Jogar(char tabuleiro[MAXSIZE][MAXSIZE], int N, int M, int K) {
    int i,j, alterado=0, casa;
    // O primeiro lance é aleatório sobre todo o tabuleiro,
    //   da mesma forma que as minas são colocadas. Essa casa passará a ser a
explorada;
    // Ao explorar uma casa, se esta contiver uma mina, o jogo pára
    //   com a explosão nessa casa, caso contrário a casa ficará
    //   com a indicação do número de minas nas casas adjacentes;
    // Caso não seja aplicável nenhum dos pontos anteriores,
    //   selecionar uma casa aleatória para explorar, aceitando a casa a explorar
    //   caso não esteja explorada nem marcada como tendo uma mina.
    do {
        casa=randaux()%(N*M);
        if(NaoExplorado(tabuleiro[casa/M][casa%M])) {
            printf("Casa %d\n",casa); // mostrar o lance realizado à sorte
            if(ExplorarCasa(casa,tabuleiro,N,M)<0)
                return -1; // muito azar acertar na mina, nada a fazer, perdeu o
jogo
        } else
            casa=-1; // tentar outra casa
    } while(casa<0);

    // Caso exista uma casa explorada com 0 minas adjacentes,
    //   explorar todas as casas adjacentes não exploradas;
```

```
    // Caso exista uma casa explorada com W minas adjacentes,
    //   existindo W casas marcadas como tendo minas, explorar as restantes casas
adjacentes;
    // Caso exista uma casa explorada com W minas adjacentes,
    //   existindo W casas não exploradas, marcar todas essas casas como tendo
minas.
    do {
        alterado=0;
        for(i=0;i<N;i++)
            for(j=0;j<M;j++)
                alterado+=Analisar(tabuleiro,N,M,i,j);
    } while(alterado>0);

    // Caso já não existam casas por explorar, terminar o jogo
    for(i=0;i<N;i++)
        for(j=0;j<M;j++)
            if(NaoExplorado(tabuleiro[i][j]))
                return 1;

    return -1;
}

int main() {
    int N, M, K, i, casa;
    char tabuleiro[MAXSIZE][MAXSIZE];
    printf("N: ");
    scanf("%d",&N);
    printf("M: ");
    scanf("%d",&M);
    printf("K: ");
    scanf("%d",&K);
    printf("\n");

    /* posicionar as minas */
    InicializarTabuleiro(tabuleiro,N,M);
    for(int i=0;i<K;i++)
        ColocarMina(tabuleiro,N,M);

    /* efetuar uma jogada, enquanto possível */
    while(Jogar(tabuleiro,N,M,K)>=0);

    /* posição final */
    MostrarTabuleiro(tabuleiro,N,M);
}



2
3
2
N: M: K:
Casa 4
+++
+X#


3
2
3
N: M: K:
Casa 0
X+
++
```

```
##

3
3
1
N: M: K:
Casa 8
Casa 7
Casa 4
011
01x
011

4
4
2
N: M: K:
Casa 14
Casa 4
Casa 1
Casa 12
Casa 9
001#
112+
+X++
1+1+




10
2
4
N: M: K:
Casa 9
Casa 4
Casa 18
Casa 2
Casa 5
Casa 1
#X
2+
11
+#
+1
++
++
#+
11
00


4
4
5
N: M: K:
Casa 12
Casa 6
Casa 14
+#+#
#+3+
+#++
1+X+
```

```
4
5
6
N: M: K:
Casa 18
Casa 2
Casa 4
##2+X
++#+#
++++#
+++1+

10
4
10
N: M: K:
Casa 25
+###
#+++
+#++
++#+
++++
#+++
#X+#
++++
++++
++#+



8
8
8
N: M: K:
Casa 18
Casa 49
Casa 59
Casa 41
0001x100
00011211
000001x1
11211111
2x3x2211
+X++#+#+
111113++
000001#+

8
16
16
N: M: K:
Casa 38
1101x10000000000
x112210011100111
112x20001x2112x2
002x2000223x12x2
012211111x211111
12x101x122322100
2x3101111x2xx100
2x20000011222100
```

```
10
10
20
N: M: K:
Casa 2
Casa 53
Casa 92
Casa 82
Casa 16
Casa 18
Casa 95
x101##++++
1102++2+1+
0002#++#++
1223#+#+++
+###+#++++
+++3++++#+
+###+++#+#
++++++++#+
+#2+++++++
+#2++X++++


10
20
40
N: M: K:
Casa 122
Casa 133
Casa 73
Casa 64
++++#++++++#++++++++
+#++++++#++++++++#+++
+#++++++#++++++++++++
++++X++#+#+++2++++#+
+#++++++++++#+#++++#
#+###+++++++++++#+#+
++3+#+#++++++3#+++#+
++#++#++++++++#++++#+
+##++++++++#+#++++++++
++#+++++++++#+++#++++


20
40
80
N: M: K:
Casa 356
Casa 640
Casa 782
1110000000000000000111001112x200011101x10
2x211100000000000001x2101x23x20001x101110
2x32x211000000000012x3223x31111122200000
12x212x10000000000012xx12x2001x11x100000
011212110000000000001221111001111211000
1212x212110011100000000000001110001x1111
x2x212x3x2012x210000000000001x10001222x1
12110113x311x3x10111000000122100001x211
111000012x33221101x101110001x11121111100
1x10000012xx1000011101x10001222x2x100000
1111110001221000000011211012x2121100000
0001x10000111000000000001x212x22110000111
```

00022200001x10000000000112x2111x100002x2
0112x10000111000000000000011110122101112x2
01x223210011100000001121100001x1002x3211
01111xx2101x100000001x3x21000111002xx100
0000123x11221011101134x5x100000111122100
0011101111x1123x101x3xxx21111001x2111110
001x100012211xx211223xx3101x100112x11x10
001110001x10122101x112210011100001111110