Gabarito Prova 2
Turma W

Questão 1a

- Defina um novo tipo de dados chamado Aluno a partir de uma estrutura que tenha os seguintes campos: número de matrícula do aluno, nota da prova 1 e nota da prova 2.

Questão 1a

```
typedef struct aluno {
    int matricula;
    int prova1;
    int prova2;
} aluno;
```

Questão 1b

- Implemente um procedimento de nome **preencheAluno** que recebe um **Aluno** como parâmetro e que preenche os seus campos com valores lidos do teclado. **A passagem de parâmetro deve ser feita por referência.**

Questão 1b

… que recebe um **Aluno** como parâmetro

```
void preencheAluno(aluno a) {

}
```

Questão 1b

A passagem de parâmetro deve ser feita por referência.

```
void preencheAluno(aluno *a) {

}
```

Questão 1b

```
void preencheAluno(aluno *a) {
    int mat, p1, p2;
    scanf("%d %d %d", &mat, &p1, &p2);
    a->matricula = mat;
    a->prova1 = p1;
    a->prova2 = p2;
}
```

Questão 1b

```
void preencheAluno(aluno *a) {
    int mat, p1, p2;
    scanf("%d %d %d", &mat, &p1, &p2);
    (*a).matricula = mat;
    (*a).prova1 = p1;
    (*a).prova2 = p2;
}
```

Questão 1b

```
void preencheAluno(aluno *a) {
    int mat, p1, p2;
    scanf("%d %d %d", &a->matricula, &a->prova1,
    &a->prova2);
}
```

Questão 1c

- Implemente uma função RECURSIVA de nome **alunosAprovados** que recebe um vetor de alunos e retorna o número de alunos que tiveram a sua nota final maior que 60 pontos. A função não deve fazer uso de variáveis globais, apenas da definição NUM_ALUNOS.

Questão 1c

… recebe um vetor de alunos e retorna o número de alunos que tiveram a sua nota final maior que 60 pontos.

```
int alunosAprovados(aluno alunos[], ...) {
    return 0;
}
```

Questão 1c

```
int alunosAprovados(aluno alunos[], int i) {
    if(i == NUM_ALUNOS)
        return 0;
    if(alunos[i].prova1 + alunos[i].prova1 > 60)
        return 1 + alunosAprovados(alunos, i+1);
    return alunosAprovados(alunos, i+1);
}
```

Questão 1d

- Escreva um programa para ler as informações de NUM_ALUNOS alunos do teclado e imprimir o número de alunos que tiveram nota final maior que 60. Para este programa você deve usar as funções preencheAluno e alunosAprovados, considerando que elas estão implementadas de forma correta.

Questão 1d

```
#include <stdio.h>
#include “prova2.h”
void main() {
    int i=0;
    aluno alunos[NUM_ALUNOS];
    for(i=0; i<NUM_ALUNOS; i++)
        preencheAluno(&alunos[i]);
    printf("\nap.: %d", alunosAprovados(alunos, 0));
}
```

Questão 2a

Escreva um procedimento para preencher aleatoriamente uma matriz quadrada N × N com 0s ou 1s. A probabilidade de preencher uma dada posição (i, j) com 0 deve ser a mesma de preencher com 1. Essa função deve ter o seguinte protótipo:

```
int preencheMatriz(int M[][N]);
```

Questão 2a

```
void preencheMatriz(int M[][N]) {
    int i,j;
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            M[ i ][ j ] = random(2);
}
```

Questão 2b

- Uma matriz simétrica possui a linha i (por exemplo, a segunda linha) igual à sua coluna i (por exemplo, a segunda coluna) para todas as suas linhas e colunas. Escreva uma função que retorna 1 caso uma matriz quadrada N × N seja simétrica e 0 caso contrário. Essa função deve ter o seguinte protótipo:

```
int verificaSimetrica(int M[][N]);
```

Questão 2b

```
int verificaSimetrica(int M[][N]) {
    int i,j;
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            if(M[ i ][ j ] != M[ j ][ i ])
                return 0;
    return 1;
}
```

Questão 2c

- Escreva um programa para gerar uma uma matriz quadrada N ×N simétrica usando somente a função preencheMatriz. Esse programa deve informar o número de vezes que a função preencheMatriz foi chamada até que matriz tenha sido gerada de forma simétrica.

Questão 2c

```
#include <stdio.h>
#include “prova2.h”
void main() {
    int count = 0, M[N][N];
    do {
        preencheMatriz(M);
        count++;
    } while(verificaSimetrica(M) == 0);
    printf("\n count = %d", count);
}
```

Questão 3

Implemente um procedimento RECURSIVO que imprime o inverso de um número inteiro n. Exemplo: imprimeInverso(4567) imprime 7654 na tela. O procedimento não deve usar variáveis globais e deve ter o seguinte protótipo:

```
void imprimeInverso(int n);
```

Questão 3

```
void imprimeInverso(int n) {
    if(n < 10)
        printf("%d", n);
    else {
        int div = n/10;
        int r = n%10;
        printf("%d", r);
        imprimeInverso(div);
    }
}
```

Questão 3

```
void imprimeInverso(int n) {
    int l = log10(n);
    int d = pow(10,l);
    if(l > 0) {
        imprimeInverso(n%d);
        printf("%d", n/d);
    }
    else
        printf("%d", n);
}
```

Questão 4

Implemente uma função RECURSIVA que retorna o inverso de um número inteiro n. Exemplo: retornaInverso(4567) retorna 7654. A função não deve usar variáveis globais e deve ter o seguinte protótipo:

```
int retornaInverso(int n);
```

Questão 4

- int retornaInverso(int n) {

```
    int l = log10(n);
    int m = pow(10,l);
    int r = n%10;
    int d = n/10;
    if(l > 0)
        return m*r + retornaInverso(d);
    return n;
}
```