Gabarito Prova 2
Turma W

Questão 1a

- Defina um novo tipo de dados chamado **Atleta** a partir de uma estrutura que tenha os seguintes campos: código do atleta (inteiro), número de medalhas de ouro, número de medalhas de prata, número de medalhas de bronze.

Questão 1a

```
typedef struct atleta {
    int cod_atleta;
    int ouro;
    int prata;
    int bronze;
} Atleta;
```

Questão 1b

- Implemente um procedimento de nome **preencheAtleta** que recebe um **Atleta** 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 preencheAtleta(Atleta a) {

}
```

Questão 1b

A passagem de parâmetro deve ser feita por referência.

```
void preencheAtleta(Atleta *a) {

}
```

Questão 1b

```
void preencheAtleta(Atleta *a) {
    int cod, o, p, b;
    scanf("%d %d %d %d", &cod, &o, &p, &b);
    a->cod_atleta = cod;
    a->ouro = o;
    a->prata = p;
    a->bronze = b;
}
```

Questão 1b

```
void preencheAtleta(Atleta *a) {
    int cod, o, p, b;
    scanf(“%d %d %d %d”, &cod, &o, &p, &b);
    (*a).cod_atleta = cod;
    (*a).ouro = o;
    (*a).prata = p;
    (*a).bronze = b;
}
```

Questão 1b

```
void preencheAtleta(Atleta *a) {
    int cod, o, p, b;
    scanf("%d %d %d %d", &a->cod_atleta, &a->ouro,
    &a->prata, &a->bronze);
}
```

Questão 1c

- Implemente uma função RECURSIVA de nome **numTotalMedalhasPorTipo** que recebe um vetor de atletas **atletas[]** e o tipo da medalha (ouro=1, prata=2 ou bronze=3) e retorna o número total de medalhas desse tipo que os atletas brasileiros obtiveram. A função não deve fazer uso de variáveis globais, apenas da definição NUM_ATLETAS.

Questão 1c

```
int numTotalMedalhasPorTipo(Atleta atletas[], int tipoMedalha, int i) {
    if(i==NUM_ATLETAS)
        return 0;
    if(tipoMedalha == 1)
        return atletas[i].ouro + numTotalMedalhasPorTipo(atletas,
        tipoMedalha, i+1);
    if(tipoMedalha == 2)
        return atletas[i].prata + numTotalMedalhasPorTipo(atletas,
        tipoMedalha, i+1);
    return atletas[i].bronze + numTotalMedalhasPorTipo(atletas,
    tipoMedalha, i+1);
}
```

Questão 1d

- Escreva um programa para ler as informações de NUM_ATLETAS atletas do teclado e imprimir o número total de medalhas de ouro que eles obtiver Para este programa vocˆe deve usar as funções preencheAtleta e numTotalMedalhasPorTipo, considerando que elas estão implementadas de forma correta.

Questão 1d

```
void main() {
    int i=0;
    Atleta atletas[NUM_ATLETAS];
    for(i=0; i<NUM_ATLETAS; i++)
        preencheAtleta(&atletas[i]);
    int total = numTotalMedalhasPorTipo(atletas, 1, 0);
    printf("\ntotal: %d", total);
    getchar();
}
```

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 identidade é uma matriz quadrada em que os elementos da diagonal principal têm valor um, e os demais elementos da matriz são zero. Escreva uma função que retorna 1 caso uma matriz quadrada N×N seja uma matriz identidade e 0 caso contrário. Essa função deve ter o seguinte protótipo:

  int verificaIdentidade(int M[][N]);

Questão 2b

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

Questão 2c

- Escreva um programa para gerar uma uma matriz quadrada N × N identidade usando somente a função **preencheMatriz** para popular a matriz. Esse programa pode usar a função **verificaldentidade** e 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(verificaIdentidade(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;
}
```