Exercício de Programação Estruturada 9

Exercício de Programação Estruturada 9

1.

```c
#include <stdio.h>

void fillMatrix(int columns, int rows, float mat[rows][columns])

{

  int i, j;

  for (i = 0; i < columns; i++)

  {

    printf("Preenchendo coluna %d:\n", i + 1);

    for (j = 0; j < rows; j++)

    {

      printf("Valor da linha %d: ", j + 1);

      scanf("%f", & mat[j][i]);

    }

  }

}

float findBiggest(int columns, int rows, float mat[rows][columns])

{

  int i, j;

  float biggest;
```

```
  biggest = mat[0][0];
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < rows; j++)
    {
      if (biggest < mat[i][j])
      {
        biggest = mat[i][j];
      }
    }
  }
  return biggest;
}
float multiply(int columns, int rows, float mat[rows][columns])
{
  int i;
  float prod, biggest;
  prod = 1;
  biggest = findBiggest(columns, rows, mat);
  for (i = 0; i < rows; i++)
  {
    prod = mat[i][i] * prod;
  }
```

```
  prod = prod * biggest;
  return prod;
}
void printMatrix(int columns, int rows, float mat[rows][columns])
{
  int i, j;
  for (i = 0; i < rows; i++)
  {
    for (j = 0; j < rows; j++)
    {
      printf("%.2f ", mat[i][j]);
    }
    printf("\n");
  }
}
int main()
{
  int rows, columns;
  rows = 5;
  columns = 5;
  float mat[rows][columns], prod;
  fillMatrix(columns, rows, mat);
```

```
  prod = multiply(columns, rows, mat);
  printf("A matriz inserida:\n");
  printMatrix(columns, rows, mat);
  printf("Valor da multiplicação: %.2f.\n", prod);
  return 0;
}
```

2.

```
#include <stdio.h>
void fillMatrix(int size, float mat[size][size])
{
  int i, j;
  for (i = 0; i < size ; i++)
  {
    printf("Preenchendo coluna %d:\n", i + 1);
    for (j = 0; j < size ; j++)
    {
      printf("Valor da linha %d: ", j + 1);
      scanf("%f", & mat[j][i]);
    }
  }
}
int searchRow(int size, int row, float mat[size][size])
```

```
{
  int i, sum;
  sum = 0;
  for (i = 0; i < size ; i++)
  {
    sum = mat[row][i] + sum;
  }
  if (sum == 1.0)
  {
    return 1;
  }
  return 0;
}
int searchColumn(int size, int column, float mat[size][size])
{
  int i, sum;
  sum = 0;
  for (i = 0; i < size ; i++)
  {
    sum = mat[i][column] + sum;
  }
```

```
  if (sum == 1.0)
  {
    return 1;
  }
  return 0;
}
int isPermut(int size, float mat[size][size])
{
  int i, sum;
  sum = 0;
  for (i = 0; i < size; i++)
  {
    sum = sum + searchColumn(size, i, mat);
  }
  if (sum == size)
  {
    return 1;
  }
  return 0;
}
int main()
{
  int size, permutValue;
```

```
  printf("Insira o número de colunas da matriz: ");

  scanf("%d", & size);

  float mat[size][size];

  fillMatrix(size, mat);

  permutValue = isPermut(size, mat);

  if (permutValue == 1)
  {

    printf("é permutação.\n");

  }
  else

  {

    printf("Não é permutação.\n");

  }

}
```

3.

```
#include <stdio.h>

void fillMatrix(int size, int mat[size][size])

{

  int i, j;

  printf("Preenchendo matriz:\n");
  for (i = 0; i < size ; i++)
  {
    for (j = 0; j < size ; j++)
    {
      printf("Valor na linha %d, coluna %d: ", i+ 1, j + 1);
      scanf("%d", &mat[i][j]);
    }
```

```
  }
}

int findNum(int size, int mat[size][size])
{
    int i, sum;
    sum = 0;
    for (i = 0; i < size; i++)
    {
        sum = sum + mat[0][i];
    }
    return sum;
}

int checkLines(int size, int mat[size][size], int num)
{
    int i, j, sum, ok;
    ok = 0;
    for (i = 0; i < size; i++)
    {
        sum = 0;
        for (j = 0; j < size; j++)
        {
            sum = sum + mat[i][j];
        }
        if (sum == num)
        {
            ok = ok + 1;
        }
    }
    if (ok == size)
    {
        return 1;
    }
    return 0;
}

int checkColumns(int size, int mat[size][size], int num)
{
    int i, j, sum, ok;
    ok = 0;
    for (i = 0; i < size; i++)
    {
```

```
        sum = 0;
        for (j = 0; j < size; j++)
        {
            sum = sum + mat[j][i];
        }
        if (sum == num)
        {
            ok = ok + 1;
        }
    }
    if (ok == size)
    {
        return 1;
    }
    return 0;
}

int checkDiagonals(int size, int mat[size][size], int num)
{
    int i, j, sumA, sumB;
    sumA = 0;
    sumB = 0;
    for (i = 0; i < size; i++)
    {
        sumA = sumA + mat[i][i];
    }
    for (i = 0; i < size; i++)
    {
        sumB = sumB + mat[i][size - (i + 1)];
    }
    if (sumA == num && sumB == num)
    {
        return 1;
    }
    return 0;
}

int isMagicCube(int size, int mat[size][size], int num)
{
    if (checkLines(size, mat, num) + checkColumns(size, mat, num) +
checkDiagonals(size, mat, num) == 3)
    {
        return 1;
```

```
    }
    return 0;
}

int main()
{
    int size, num;
    printf("Insira o número de colunas da matriz: ");
    scanf("%d", & size);
    int mat[size][size];
    fillMatrix(size, mat);
    num = findNum(size, mat);
    if (isMagicCube(size, mat,num) == 1)
    {
        printf("A matriz é um quadrado mágico\n");
    }
    else
    {
        printf("A matriz não é um quadrado mágico\n");
    }


    return 0;
}
```

4.

```
#include <stdio.h>

void fillMatrix(int size, int mat[size][size]) {
    int i, j;
    printf("Preenchendo a matriz:\n");
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("Valor na linha %d, coluna %d: ", i+ 1, j + 1);
            scanf("%d", &mat[i][j]);
        }
    }
}
int calculateSumAboveDiagonal(int size, int mat[size][size]) {
    int i, j, sum;
    sum = 0;
    for (int i = 0; i < size; i++) {
        for (int j = i + 1; j < size; j++) {
```

```
            sum =  sum + mat[i][j];
        }
    }
    return sum;
}
void printMatrix(int size, int mat[size][size]) {
    int i, j;
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}

int main() {
    int size = 4, mat[size][size], sum;
    fillMatrix(size, mat);
    printf("A matriz inserida:");
    printMatrix(size, mat);
    sum = calculateSumAboveDiagonal(size, mat);
    printf("A soma dos elementos acima da diagonal principal é: %d\n", sum);

    return 0;
}
```