Lista 3

Nome: Ravi Antônio Gonçalves de Assis

Curso: Engenharia de Software

Obs: para os exercícios abaixo considere que todos os arranjos (vetores e matrizes) são de inteiros.

1 – Faça um método que receba um vetor e retorne a soma dos seus elementos.

```
public static int sum(int[] vet) {
	int soma = 0;
	for (int i = 0; i < vet.length; i++) soma += vet[i];
	return soma;
}
```

2 – Faça um método que receba um vetor e retorne o menor elemento.

```
public static int lessElement(int[] vet) {
	// TODO Auto-generated method stub
	int menor = vet[0];
	for (int i = 0; i < vet.length; i++) menor = ( menor
> vet[i] ) ? vet[i] : menor ;
	return menor;
}
```

3 – Faça um método que receba um vetor e um inteiro n. Retorne a primeira posição onde n ocorre ou -1 se ele não estiver presente no vetor.

```
public static int indexOf(int[] vet, int n) {
	// TODO Auto-generated method stub
	for (int i = 0; i < vet.length; i++) {
		if (n == vet[i]) return i;
	}
	return -1;
}
```

4 – Faça um método que receba um vetor e um inteiro n. Retorne a última posição onde n ocorre ou -1 se ele não estiver presente no vetor.

```
public static int lastIndexOf(int[] vet, int n) {
	// TODO Auto-generated method stub
	for (int i = vet.length -1; i >= 0; i--) {
		if (n == vet[i]) return i;
	}
	return -1;
}
```

5 – Faça um método que receba um vetor e um inteiro n. Retorne quantas vezes n ocorre no vetor.

```
public static int countElements(int[] vet, int n) {
	// TODO Auto-generated method stub
	int count = 0;
	for (int i = 0; i < vet.length; i++) {
		if(n == vet[i]) count++;
	}
```

```
            return count;
      }
```

6 – Faça um método que receba uma matriz. Retorne um vetor contendo os elementos da diagonal principal ou um vetor vazio (caso a matriz não seja quadrada).

```
      public static int[] mainDiagonal(int[][] mat) {
            // TODO Auto-generated method stub
            int vetDiagonal[] = new int[mat.length];
            for (int i = 0; i < mat.length; i++) {
                  if (mat.length != mat[i].length) return
vetDiagonal;;
            }

            for (int i = 0; i < mat.length; i++) vetDiagonal[i] =
mat[i][i];
            return vetDiagonal;

      }
```

7 – Faça um método que linearize uma matriz, retornando seus elementos em um vetor.

```
       public static int[] alingArray(int[][] mat) {
            // TODO Auto-generated method stub
            int tam = 0;
            int aux = 0;
            for (int i = 0; i < mat.length; i++) {
                  tam += mat[i].length;
            }
            int vet[] = new int[tam];

            for (int i = 0; i < mat.length; i++) {
                  for (int j = 0; j < mat[i].length; j++) {
                        vet[aux] = mat[i][j];
                        aux++;
                  }
            }

            return vet;
      }
```

8 – Faça um método em Java que receba um vetor e o retorne invertido.

```
       public static int[] invertArray(int[] vet) {
            // TODO Auto-generated method stub
            int tam = vet.length;
            int vetInvert[] = new int[tam];
            for (int i = 0, j = tam-1; i < vet.length; i++, j--)
{
                  vetInvert[i] = vet[j];
            }
            return vetInvert;
      }
```

9 – Faça um método em Java que receba um vetor e os índices esq (esquerda) e dir (direita). Seu método deve retornar um vetor contendo todos os elementos entre esq e dir. Obs: o método deve testar se os valores de esq e dir são válidos.

```
	public static int[] arrayPart(int[] vet, int esq, int dir)
{
		// TODO Auto-generated method stub
		if (esq > dir || esq < 0 || esq >= vet.length || dir
< 0 || dir >= vet.length) return null ;
		else {
			int partVet[] = new int[dir - esq +1];
			for (int i = 0; i < partVet.length; i++ ) {
				partVet[i] = vet[esq+i];
			}
			return partVet;
		}
	}
```

10 – Faça um método em Java que receba uma matriz e os índices l1, c1, l2, c2. Seu método deve retornar a submatriz contendo os elementos entre l1, c1 e l2, c2.

```
	public static int[][] subMatriz(int[][] mat, int l1, int
l2, int c1, int c2) {
		// TODO Auto-generated method stub
		int subMatriz[][];
		if (l1 > l2 || l1 < 0 || l2 < 0 || l1 >= mat.length
|| l2 >= mat.length )
			return null;
		else {
			for (int i = 0; i < mat.length; i++) {
				if ( arrayPart(mat[i], c1, c2) == null)
return null;
			}
			subMatriz = new int[l2 - l1 + 1][c2 - c1 + 1];
			for (int i = 0; i < subMatriz.length; i++) {
				subMatriz[i] = arrayPart(mat[i + l1], c1,
c2);
			}
			return subMatriz;
		}
	}
```

11 - Faça um método em Java que receba como parâmetro: um arranjo de 3 dimensões, a face f e os índices l1, c1, l2, c2. Seu método deve retornar uma matriz contendo os elementos entre l1, c1 e l2, c2 da face f passada por parâmetro. Obs: teste se os parâmetros são válidos. Se não forem, retorne null.

```
	public static int[][] subMatriz(int[][][] mat3d, int f, int
l1, int l2, int c1, int c2) {
		// TODO Auto-generated method stub
		if (f < 0 || f >= mat3d.length) {
			return null;
		}
		else return subMatriz(mat3d[f], l1, l2, c1, c2);
```

```
}
```

12 – Faça um método em Java que receba dois inteiros n e op e retorne um vetor com n elementos preenchido da seguinte forma: • Se op = 0, preencha o vetor com os valores entre 0 e n-1 • Se op = 1, preencha o vetor com os valores entre n-1 e 0

Se op = 2, preencha o vetor com os valores aleatórios inteiros entre 1 e n.

```
public static int[] fillArray(int n, int op) {
	// TODO Auto-generated method stub
	int vet[] = new int[n];
	if (op == 0) {
		for (int i = 0; i < vet.length; i++) {
			vet[i] = i;
		}
		return vet;
	}
	else if (op == 1) {
		int num = n-1;
		for (int i = 0; i < vet.length; i++, num--) {
			vet[i] = num;
		}
		return vet;
	}
	else if (op == 2) {
		for (int i = 0; i < vet.length; i++) {
			vet[i] = (int)( Math.random() * 10 );
		}
		return vet;
	}
	else return null;
}
```