1) Retorna o valor de um array de Ints

2ª)

```
data Mobile = Pendente Int | Barra Mobile Mobile
```

a)

```
peso :: Mobile -> Int
peso (Pendente n) = n
peso (Barra m1 m2) = peso m1 + peso m2
```

b)

```
balanceado :: Mobile -> Bool
balanceado (Pendente n) = True
balanceado (Barra m1 m2) = do
if peso m1 == peso m2 then
left && right
else False
where
left = balanceado m1
right = balanceado m2
```

3ª)

b)

```
public class RMR extends Thread{
	int [] recipiente = new int[2];
	boolean isEmpty;
	boolean isMixed;

	RMR () {
		this.recipiente[0] = 0;
		this.recipiente[1] = 0;
		this.isEmpty = true;
		this.isMixed = false;
	}

	synchronized public void mix () {
		while (!(this.recipiente[0] == 30 && this.recipiente[1] == 40)) {
			try {
				wait();
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		System.out.println("Misturando sorvete");
		this.isMixed = true;
		notify();
	}

	synchronized public void retirarSorvete () {
		while (!(this.isMixed && !this.isEmpty)) {
			try {
				wait();
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		System.out.println("Retirando sorvete");
		this.isEmpty = true;
```

```
            this.isMixed = false;
            this.recipiente[0] = 0;
            this.recipiente[1] = 0;
            notify();
      }

      synchronized public void inserirIngreditente (int position, int
quantidade) {
            while (this.recipiente[position] != 0) {
                  try {
                        wait();
                  } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }
            }
            System.out.println("Adicionando substancia " + position);
            this.recipiente[position] = quantidade;
            this.isEmpty = false;
            notifyAll();
      }

      public static void main(String[] args) {
            RMR recipiente = new RMR();
            ProdutorIng1 pd1 = new ProdutorIng1(recipiente);
            ProdutorIng2 pd2 = new ProdutorIng2(recipiente);
            Misturador mixer = new Misturador(recipiente);
            RetiradorSorvete rs = new RetiradorSorvete(recipiente);
            pd1.start();
            pd2.start();
            mixer.start();
            rs.start();
      }

}

class ProdutorIng1 extends Thread {
      RMR recipiente;
      ProdutorIng1(RMR r) {
            this.recipiente = r;
      }

      public void run() {
            while (true) {
                  this.recipiente.inserirIngreditente(0, 30);
            }
      }
}

class ProdutorIng2 extends Thread {
      RMR recipiente;
      ProdutorIng2(RMR r) {
            this.recipiente = r;
      }

      public void run() {
            while (true) {
                  this.recipiente.inserirIngreditente(1, 40);
            }
      }
}
```

```
class Misturador extends Thread {
	RMR recipiente;
	Misturador(RMR r) {
		this.recipiente = r;
	}

	public void run() {
		while (true) {
			this.recipiente.mix();
		}
	}
}

class RetiradorSorvete extends Thread {
	RMR recipiente;
	RetiradorSorvete(RMR r) {
		this.recipiente = r;
	}

	public void run() {
		while (true) {
			this.recipiente.retirarSorvete();
		}
	}
}
```

a)

```
import Control.Concurrent

main :: IO ()
main = do
rmr <- newMVar $ makeNewRecipiente
forkIO (mix rmr)
forkIO (insert rmr "substancia1" 30)
forkIO (insert rmr "substancia2" 40)
forkIO (remove rmr)
return ()

data Recipiente = Recipiente
{ substancia1 :: Int
, substancia2 :: Int
, isMixed :: Bool } deriving (Show)

makeNewRecipiente :: Recipiente
makeNewRecipiente = Recipiente { substancia1 = 0, substancia2 = 0, isMixed = False }

mix :: MVar (Recipiente) -> IO ()
mix rmr = do
recipiente <- takeMVar rmr
if substancia1 recipiente == 30 && substancia2 recipiente == 40 then
recipiente { isMixed = True }
else retry

insert :: MVar (Recipiente) -> String -> Int -> IO ()
insert rmr position value = do
```

```
recipiente <- takeMVar rmr
if position == "substancia1" && substancia1 recipiente == 0 then
recipiente { substancia1 = value }
else if position == "substancia2" && substancia2 recipiente == 0 then
recipiente { substancia2 = value }
else retry

remove :: MVar (Recipiente) -> IO ()
remove rmr = do
recipiente <- takeMVar rmr
if isMixed recipiente then
recipiente { isMixed = False, substancia1 = 0, substancia2 = 0 }
else retry
```