Paulo Henrique e Aki Natan

1.

a) >> v(32)

b) >> v(end-2:end)

c) >> p = v(mod(v, 2) == 0)

2.

a) De 1 até 7 em um passo de 2 atribua a um vetor 'u' os valores requeridos do vetor 'z'

>> u = 10 30 50 70

b) De 7 até 1 em um passo de -2 atribua a um vetor 'v' os valores requeridos do vetor 'z'

>> v = 70 50 30 10

c) Atribui a uma variável 'w' um vetor com os valores referente a cada posição do vetor 'z' referenciados, os valores das posições 3, 4, 5 e 1

>> w = 30 40 50 10

d) Atribui o valor '0' a cada posição do vetor 'z', começando da posição 1 até a 7 em um passo de 2

>> z = 0 20 0 40 0 60 0 80

e) Atribui o valor '1' até ‘4’ a cada posição do vetor 'z', começando da posição 7 até a 1 em um passo de -2

z = 4 20 3 40 2 60 1 80

f) Apaga as posições 1 a 3 do vetor

>> z = 40 1 60 1 80

3.

a) Gera uma matriz resultante de 'A', entretanto apenas com os valores das colunas 2 e 3 da linha 1

```
>>  7   9
```

b) Gera uma matriz resultante de 'A', entretanto apenas com os valores das colunas 1 e 4 de todas as linhas

```
>> 2   0
   3   6
   8   5
```

c) Gera um vetor resultante de 'A', entretanto com todos os valores de suas colunas referentes a linha 2

```
>> 3   0   5   6
```

d) Atribui o valor '5' a todas as posições da linha 2 em todas as colunas da matriz 'A'

```
>> 2   7   9   0
   5   5   5   5
   8   2   0   5
```

e) Gera uma matriz em que todas as colunas se unem em apenas uma

```
>> 2
   3
   8
   7
   0
   2
   9
   5
   0
   0
   6
   5
```

f) Através da atribuição de '[]' se apaga todos os elementos em todas as posições de colunas da linha 1 da matriz 'A', ou seja, se apaga a linha 1

```
>>  3   0   5   6
    8   2   0   5
```

g) Atribui uma matriz 'B' com todas as linhas de 'A', entretanto com 3 colunas e seus valores são respectivos ao segundo elemento de cada linha da matriz 'A'

```
B =
  7   7   7
  0   0   0
  2   2   2
```

h) Atribui a 'C' uma matriz com todos os elementos da coluna 2 de 'A', em que a matriz 'C' possui um total de 3 de cópias repetidas da coluna 2 de A' em suas colunas

```
C =
  7   7   7
  0   0   0
  2   2   2
```

i) Somatório de todos os valores de cada posição da coluna na posição pontual de cada linha da matriz 'A'

```
>>  13    9   14   11
```

j) Somatório de todos os valores de cada posição da coluna na posição pontual de cada linha da matriz transposta de 'A'

```
>>  18   14   15
```

k) Somatório de todos os valore de cada posição da coluna na posição pontual de cada linha da matriz 'A'

```
>> 18
   14
   15
```

l) Gera um vetor do quarto elemento ao nono elemento da matriz na ordem crescente de colunas

```
>> 7   0   2   9   5   0
```

m) Concatena a matriz 'A' com a primeira e segunda linha e suas respectivas colunas delas mesmas

```
>> 2   7   9   0
   3   0   5   6
   8   2   0   5
   2   7   9   0
   3   0   5   6
```

n) Cria um vetor de tamanho 6 que contém todos os elementos da coluna 2 e 3 da matriz 'A'

```
>>  7   0   2   9   5   0
```

o) Pega o décimo primeiro elemento da matriz em ordem crescente de coluna

```
>> 6
```

p) Funcao que recebe como primeiro parâmetro a dimensão da matriz e como segundo parâmetro um valor, ela retorna a posicao i,j deste valor caso ele exista dentro da matriz

```
>> [l, c] = ind2sub(size(A), 11)
l =  2
c =  4
```

q) Retorna a quantidade total de índices ou posições de elementos presente na estrutura matricial

```
>> índices = find(A)
índices =

   1
   2
   3
   4
   6
```

7
8
11
12

r) Funcao retorna duas variáveis que indicam índices de coluna e linha que apresentam números '0' por exemplo:

A =

2 7 9 0
3 0 5 6
8 2 0 5

[il, ic] = find(A)
il = ic=

1 1
2 1
3 1
1 2
3 2
1 3
2 3
2 4
3 4

A variável il possui elementos em que percorre coluna por coluna identificando se em cada linha da
coluna possui um elemento diferente de '0' e caso seja verdadeiro retorna o elemento referente ao índice
da linha, como a coluna '1' possui todos os elementos diferentes de '0' ela retorna o número da linha corrente,
caso nao ela nao retorna nada.
A variável ic possui elementos do índice de linha em que se repete caso o elemento seja diferente de '0',
no caso a coluna '1' possui todos os elementos diferentes de '0', e então retorna o número da linha tres vezes,
já no caso da coluna '2', possui um elemento igual a '0', daí ela retorna duas vezes o número da linha '2'.

s) Retorna uma cópia da matriz revertida, ou seja, "virada de cabeça para baixo"

```
>> A
A =

   2   7   9   0
   3   0   5   6
   8   2   0   5

>> flipud(A)
ans =

   8   2   0   5
   3   0   5   6
   2   7   9   0
```

t) Rotaciona a matriz em 90 graus em sentido anti horário

```
>> A
A =
   2   7   9   0
   3   0   5   6
   8   2   0   5

>> rot90(A)
ans =
   0   6   5
   9   5   0
   7   0   2
   2   3   8
```

4.

a) Retorna um vetor com valores 0 ou 1 caso a comparação de maior de cada elemento do vetor x seja verdadeiro ou falso

```
>> x > y
ans =
  0  1  0  1  1  0  0
```

b) Retorna um vetor com valores 0 ou 1 caso a comparação de igual de cada elemento do vetor x seja verdadeiro ou falso

```
>> x == y
ans =
  0  0  1  0  0  1  0
```

c) Retorna um vetor com valores 0 ou 1 caso a comparação de menor/igual de cada elemento do vetor x seja verdadeiro ou falso

```
>> x <= y
ans =
  1  0  1  0  0  1  1
```

d) Operador lógico booleano 'or' em que caso exista pelo menos um elemento em x ou y diferente de '0' ele retorne verdadeiro se não falso

```
>> 1  1  1  1  1  0  1
```

e) Operador lógico booleano 'and' em que caso exista pelo menos um elemento em x ou y igual a '0' ele retorna falso se não verdadeiro

```
>> 1  1  1  1  0  0  1
```

f) Operador lógico booleano 'and' em que caso exista pelo menos um elemento em x ou y igual a '0' ele retorna falso se não verdadeiro

```
>> 1  1  1  1  0  0  1
```

g) Operador lógico booleano 'and' em que caso exista pelo menos um elemento em x ou y igual a '0' ele retorna falso se não verdadeiro

```
>> 1  1  1  1  0  0  1
```

h) Operador lógico booleano 'or' em que caso exista pelo menos um elemento em x ou y diferente de '0' ele retorne verdadeiro se não falso
>> 1 1 1 1 1 0 1

5.

a) Retorna os elementos em x nas posições acima de 5

```
>> x(x > 5)
ans =
    6    7    8    9   10
```

b) Conta quantos elementos com posições menores e iguais a 4 no vetor x e retorna um vetor de elementos de y correspondentes às posições de 1 a 4 do vetor x

```
>> y(x <= 4)
ans =
   3   1   5   6
```

c) Retorna todos um vetor de elementos com posições menores que 2 e maiores iguais a posição 8 do vetor x

```
>> x((x < 2)|(x >= 8))
ans =
    1    8    9   10
```

d) Pega todos os elementos com posições menores que 2 ou maiores iguais a posição 8 do vetor x e retorna
um vetor de y com os elementos correspondentes às posições do vetor x de acordo com a condição lógica da chamada

```
>> y((x < 2)|(x >= 8))
ans =
   3   4   7   0
```

e) Retorna um elemento de determinada posição do vetor y referente a condição da posição encontrada
na condicional lógica em que a posição em y tem que ser aquela referente à posição menor que 2 e 8 do vetor x

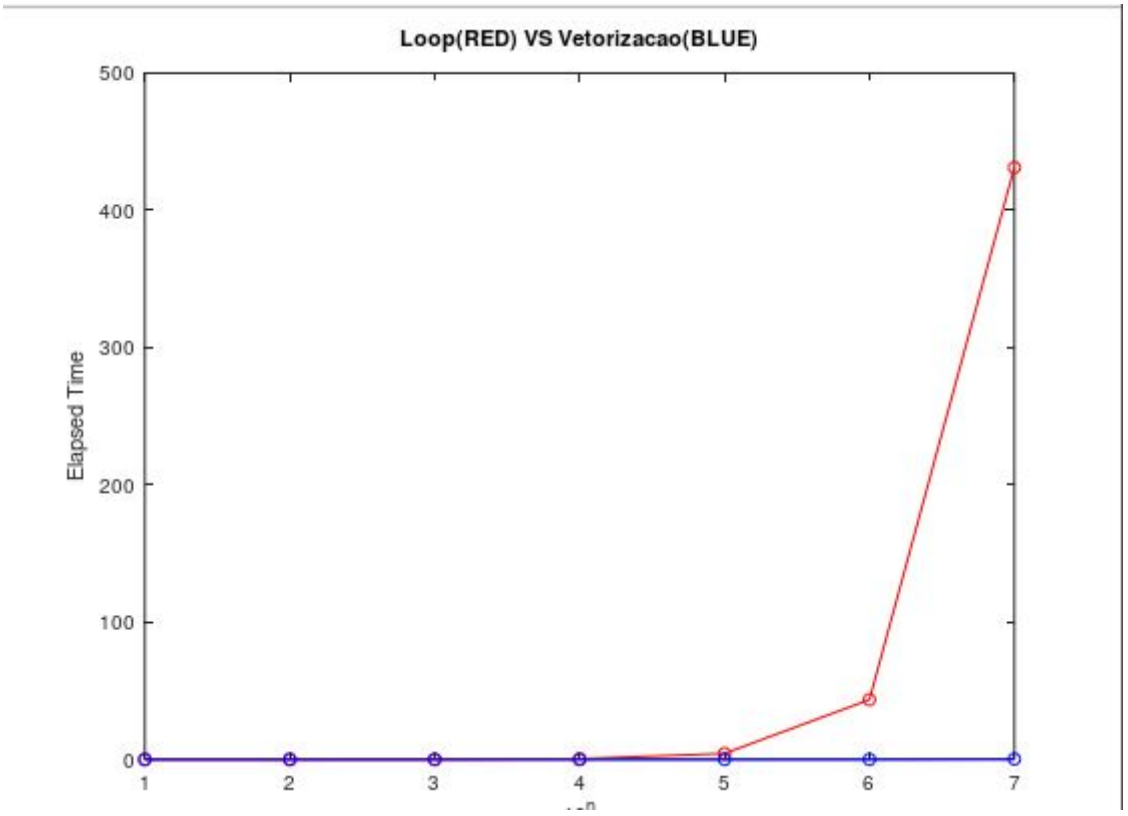
```
>> y((x < 2)&(x < 8))
ans =  3
```

f) Retorna um elemento referente a posição no vetor y que seja menor que 0

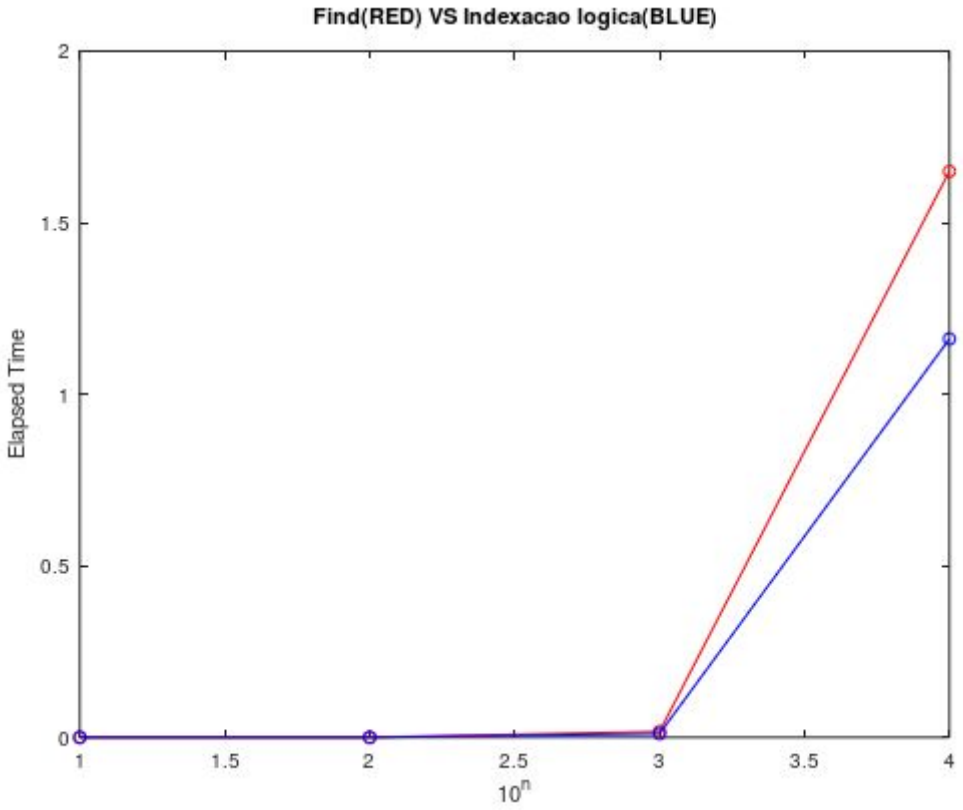
```
>> x(y<0)
ans = [](1x0)
```

6.

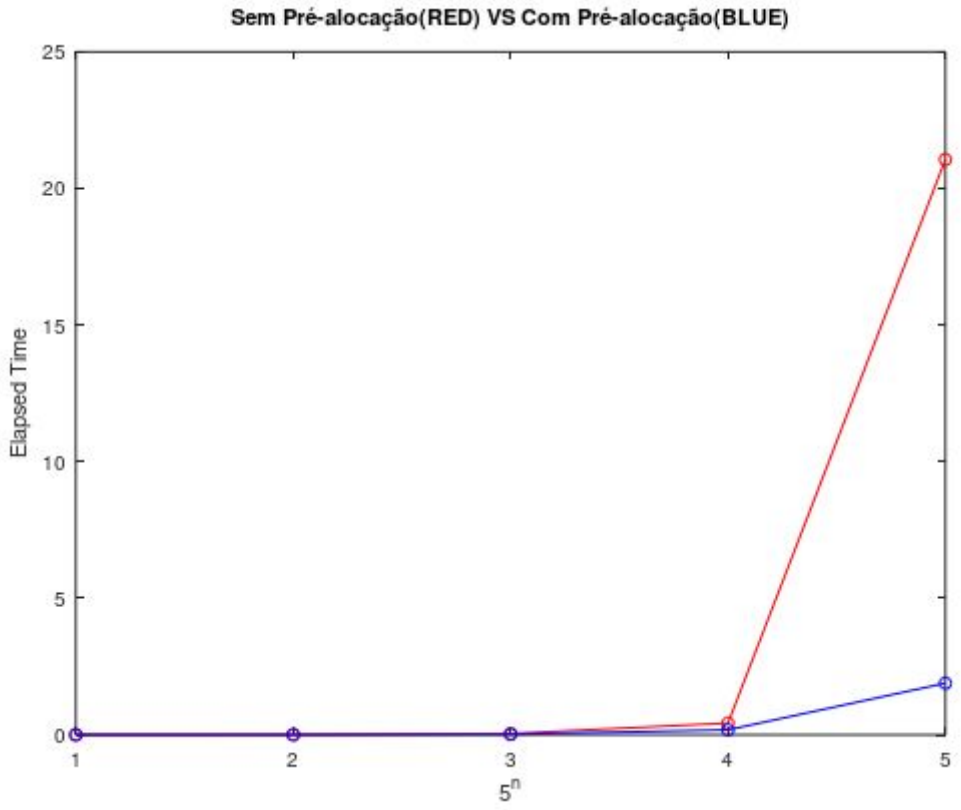
10^n	loop	vetor
10^1	0.0016618	0.000070095
10^2	0.0044539	0.000050068
10^3	0.0446520	0.000107050
10^4	0.4306018	0.000218153
10^5	4.3169088	0.001976013
10^6	43.5481730	0.034039021
10^7	431.2580452	0.327347994



10^n	find	indentação lógica
10^1	0.000090122	0.000028133
10^2	0.000168085	0.000089884
10^3	0.016906977	0.010884047
10^4	1.649744987	1.161145926



10^n	com pré-alocação	sem pré-alocação
10^1	0.00072813	0.00042796
10^2	0.00256515	0.00473404
10^3	0.01947093	0.03489804
10^4	0.19523287	0.46872497
10^5	2.83983397	24.23955297



7.

repmat *(A , m , n)* *- > Cria uma matriz m por n com uma copia de cada elemento da matriz A.*

Se n não é especificado, se cria uma matriz m por m matri.

Para copiar a matriz em mais de duas dimensões, é preciso especificar o número de vezes para copiar em cada dimensão m , n , p ,…, em um vetor no segundo argumento.

Ex:

```
>> A = repmat(10,3,2)
>> A =
    10    10
    10    10
    10    10

>> B = repmat(A,2)
>> B =
   100     0     0   100     0     0
     0   200     0     0   200     0
     0     0   300     0     0   300
   100     0     0   100     0     0
     0   200     0     0   200     0
     0     0   300     0     0   300
```

C = bsxfun(fun,A,B) *->Cria um vetor aplicando a operação binária elementar especificada pelo identificador de função funpara matrizes Ae B.*

Ex:

```
>> A = [1 2 10; 3 4 20; 9 6 15];
>> C = bsxfun (@minus, A, média (A));
>> D = bsxfun (@rdivide, C, std (A))
>> D = 3 × 3
   -0,8006   -1,0000   -1,0000
   -0,3203         0    1,0000
    1,1209    1,0000         0
```