Aula 03 – LISTAS LINEARES – Alocação Sequencial

01. Considere os quatro trechos de código apresentados a seguir, que realizam buscas em listas lineares alocadas seqüencialmente (*arrays*).

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7


int busca1 (int vet[], int n)
{
 int i = 0;
  while(i < MAX){
     if (vet[i] == n)
       return i+1;
     else
       i++;
  }
  return -1;
}
int main()
{
  int v[]={2,4,5,1,6,9,3};
  int resultado;
  resultado = busca1(v, ???);
  printf("%d\n", resultado);
  system("pause");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca2(int vet[], int n)
{
  int i = 0;
  vet[MAX] = n;
  while(vet[i]!=n)
       i++;
  if (i == MAX)
     return -1;
  else
     return i+1;
}
int main()
{
  int v[]={2,4,5,1,6,9,3,0};
  int resultado;
  resultado = busca2(v, ???);
  printf("%d\n", resultado);
  system("pause");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca3(int vet[], int n)
{
 int i = 0;

  vet[MAX] = n;
  while(vet[i]<n)
       i++;
  if ((i == MAX) || (vet[i]!= n))
     return -1;
  else
     return i+1;
}
int main()
{
  int v[]={1,2,4,5,6,7,8,0};
  int resultado;
  resultado = busca3(v, ???);
  printf("%d\n", resultado);
  system("pause");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca4(int vet[], int n)
{
  int inf = 0, sup = MAX -1, meio;

  while(inf <= sup) {
     meio = (inf + sup)/2;
     if (vet[meio] == n) {
           inf = sup + 1;
           return meio;
           }
     else if (vet[meio] < n) inf = meio +1;
           else sup = meio - 1;
 }
 if (inf >= sup) return –1;
}
int main()
{
  int v[]={1,2,4,5,6,7,8};
  int resultado;
  resultado = busca4(v, ???);
  printf("%d\n", resultado);
  system("pause"); }
```

1 – O que faz o trecho de código **busca1()?**

2 – O trecho de código **busca1()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
( ) lista desordenada

3 – O que faz o trecho de código **busca2()?**

4 – O trecho de código **busca2()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
( ) lista desordenada

5 – Considerando as funções **busca1()** e **busca2()**, qual delas é mais "eficiente" considerando o número de comparações executadas**?** Justifique sucintamente sua resposta.

6 – O que faz o trecho de código **busca3()?**

7 – O trecho de código **busca3()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
( ) lista desordenada

8 – O que faz o trecho de código **busca4()?**

9 – O trecho de código **busca4()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
( ) lista desordenada

10 – Considerando as funções **busca3()** e **busca4()**, qual delas é mais "eficiente" considerando o número de comparações executadas**?** Justifique sucintamente sua resposta.