INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO - IFSP

Disciplina	EDDA2	**Semestre**	2º
Professores	Eurides Balbino	**Atividade-lista 2**	28/08/2020
Aluno		**Prontuário**	

LISTA 2

Considere as funções a seguir para responder as questões 1 e 2.

```
float soma (float * vetor, int tamanho_vet)
{
   int    i;
   float  res=0;
   printf("\nres= %.2f\ttamanho_vet = %i", res, tamanho_vet);
   for (i = 0; i < tamanho_vet; i++)
   {
       res+=vetor[i];
       printf("\nres= %.2f\ttamanho_vet = %i", res, tamanho_vet);
   }
   return res;
}

float maior (float * vetor, int tamanho_vet)
{
   int    i;
   float  res = vetor[0];
   printf("\nmaior= %.2f\ttamanho_vet = %i", res, tamanho_vet);
   for (i = 1; i < tamanho_vet; i++)
   {
       if ( vetor[i] > res)
          res=vetor[i];
       printf("\nmaior= %.2f\ttamanho_vet = %i", res, tamanho_vet);
   }
   return res;
}
```

Questão 1) Para cada exemplo apresentado, elabore o programa-teste.

Questão 2) Elabore o algoritmo que calcule a soma dos números pares de um vetor.

Questão 3) Determine a complexidade de tempo para o algoritmo elaborado no item 2.

Questão 4) Considerando o algoritmo do ***bubble_sort***, elabore o programa para testar esse algoritmo.

```
void bubble_sort(float * vetor)
{
  int i, j; float aux;
  for ( j = 0; j<TAMANHO_VETOR-1; j++)
  {
    for ( i = 0; i<TAMANHO_VETOR-1-j; i++)
    {
      printf ("\nComparando %.2f com %.2f ", vetor[i], vetor[i+1]);
      if ( vetor[i] > vetor[i+1])
      {
         printf ("->empurra %.2f para o fundo");
         printf ("-> troca com %.2f", vetor[i], vetor[i+1]);
         aux    = vetor[i];
       vetor[i]        = vetor[i+1];
       vetor[i+1]  = aux;
      }
    }
    if (j<TAMANHO_VETOR-1)
       mostra_notas(vetor); /*exibe os valores das notas */
   }
}
```

Questão 5) Considerando o algoritmo do ***selection_sort***, elabore o programa para testar esse algoritmo.

```
void selection_sort (float * vetor)
{
    int      pos_min, i, j;
    float    aux;
    /* Percorre todo o vetor até TAMANHO_VETOR-1,
       pois a última posição não precisa testar, pois já estará ordenada */
    for(i=0; i < TAMANHO_VETOR-1; i++)
    {
        pos_min = i; /* A posição do menor valor recebe a posição que está passando */
        /* Percorre o vetor da posição i+1 até o final */
        for (j=i+1; j < TAMANHO_VETOR; j++)
        {
             /* Testa se o elemento da posição que está passando
               é menor que o elemento daquela que tem o menor valor */
            if (vetor[j] < vetor[pos_min])
            {
                pos_min = j; /* pos_min recebe a posição do menor valor */
            }
        }
        /* Se a posição do menor for diferente da que está passando, ocorre a troca */
        if (pos_min != i)
        {
            aux            = vetor[i];
            vetor[i]       = vetor[pos_min];
            vetor[pos_min] = aux;
        }
    }
}
```

Questão 6) Considerando o algoritmo do ***insertion_sort***, elabore o programa para testar esse algoritmo.

```
void insertion_sort(float * vetor)
{
    float escolhido;
    int   anterior, i;
    for (i = 1; i < TAMANHO_VETOR; i++)
    {
       escolhido = vetor[i];
       anterior  = i - 1;

          while ( (anterior >= 0) && (vetor[anterior] > escolhido) )
       {
           vetor[anterior + 1] = vetor[anterior];
           anterior--;
       }
       vetor[anterior + 1] = escolhido;
    }
}
```