1. Considerando o algoritmo de ordenação por inserção:

a) Na função *insertionSort*, troque a comparação $A[i] > x$ por $A[i] >= x$. A nova função continua produzindo uma ordenação crescente de $v[0..n-1]$?

Sim

b) O que acontece se trocarmos $for\ (j = 1$ por $for\ (j = 0$ no código da função *insertionSort*?

Continua ordenando, mas o primeiro passo, o primeiro elemento é inserido na sua própria posição.

c) Que acontece se trocarmos $A[i+1] = x$ por $A[i] = x$ no código da função *insertionSort*?

Irá sobrescrecer registros. Imagine que tenhamos o seguinte vetor [2,3,1]. Na primeira execução j=1, A[j]=x= 3 e i=0. No condicional o i é >=0, mas A[i]=2 não é maior que x=3. Então A[i] receberá 3 resultando no seguinte vetor [3,3,1]

d) Altere o algoritmo de ordenação por inserção para permutar os elementos de um vetor inteiro $A[0..n-1]$ de modo que eles fiquem em ordem decrescente.

Basta apenas alterar a condição de teste de valor para (A[i] < x)

2. Considerando o algoritmo de ordenação por seleção:

a) Na função *selecao*, o que acontece se trocarmos $for\ (i = 0$ por $for\ (i = 1$? O que acontece se trocarmos $for\ (i = 0;\ i < n - 1$ por $for\ (i = 0;\ i < n$?

A primeira troca deixa de ordenar o primeiro elemento. Já a segunda troca só realizará uma execução a mais do loop trocando o último elemento com ele mesmo.

b) Na função *selecao*, troque a comparação $v[j] < v[\min]$ por $v[j] <= v[\min]$. A nova função continua correta?

Continuará correto pois apenas irá trocar posições de valores iguais.

c) MOVIMENTAÇÃO DE DADOS: Quantas vezes, no pior caso, o algoritmo de seleção copia um elemento do vetor de um lugar para outro? Quantas vezes isso ocorre no melhor caso?

a troca é constante. Da primeira a última interação há uma troca, totalizando n-1 trocas na execução completa.

d) Escreva uma função que permute os elementos de um vetor inteiro v[0..n-1] de modo que eles fiquem em ordem decrescente. Inspire-se no algoritmo de seleção.

Basta realizar a inversão do teste de valores para if (v[j] > v[min])

3. Considerando o algoritmo de ordenação por flutuação

a) MOVIMENTAÇÃO DE DADOS: Quantas vezes, no pior caso, o algoritmo de flutuação copia um elemento do vetor de um lugar para outro? Quantas vezes isso ocorre no melhor caso?

No pior caso há n^2-1 e no melhor caso não há nenuma troca.

b) Escreva uma função que permute os elementos de um vetor inteiro v[0..n-1] de modo que eles fiquem em ordem decrescente. Inspire-se no algoritmo de flutuação.

Basta alterar o condicional de valor para if (v[j] < v[j + 1])

CÓDIGO

```c
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>

#define tamanho 5
```

```c
void bubble(int v[tamanho], int n)
{
int i, j, aux;
int k = n - 1;
for (i = 0; i < n; i++)
{
for (j = 0; j < k; j++)
{
if (v[j] > v[j + 1])
{
aux = v[j];
v[j] = v[j + 1];
v[j + 1] = aux;
}
}
k--;
}
}
```

```c
void bubbleDesc(int v[tamanho], int n)
{
int i, j, aux;
int k = n - 1;
for (i = 0; i < n; i++)
{
for (j = 0; j < k; j++)
{
if (v[j] < v[j + 1])
{
aux = v[j];
v[j] = v[j + 1];
v[j + 1] = aux;
}
}
k--;
}
}
```

```c
void selecao(int v[tamanho], int n)
{
int i, j, min, x;
for (i = 0; i < n - 1; ++i)
{
min = i;
for (j = i + 1; j < n; ++j)
if (v[j] < v[min])
min = j;
```

```
x = v[i];
v[i] = v[min];
v[min] = x;
}
}
```

```
void selecaoDesc(int v[tamanho], int n)
{
int i, j, min, x;
for (i = 0; i < n - 1; ++i)
{
min = i;
for (j = i + 1; j < n; ++j)
if (v[j] > v[min])
min = j;
x = v[i];
v[i] = v[min];
v[min] = x;
}
}
```

```
void insertionSort(int A[tamanho], int n)
{
int i, j, x, count = 0;
for (j = 1; j < n; j++)
{
x = A[j];
i = j - 1;
while ((i >= 0) && (A[i] > x))
{
A[i + 1] = A[i];
i = i - 1;
count++;
}
A[i + 1] = x;
}
printf("troca: %d \n", count);
}
```

```
void insertionSortDesc(int A[tamanho], int n)
{
int i, j, x;
for (j = 1; j < n; j++)
{
x = A[j];
i = j - 1;
while ((i >= 0) && (A[i] < x))
{
A[i + 1] = A[i];
i = i - 1;
}
A[i + 1] = x;
}
}
```

```
int main()
{
int vetor[tamanho];
vetor[0] = 3;
vetor[1] = 1;
vetor[2] = 5;
vetor[3] = 9;
vetor[4] = 7;
```

```
bubbleDesc(vetor, tamanho);
for (int i = 0; i < tamanho; i++)
{
printf("%d ", vetor[i]);
}
```

```
printf("\n");
```

```
return 0;
}
```