2. Na função selectionsort, que acontece se trocarmos for (i = 0 por for (i = 1? Que acontece se trocarmos for (i = 0; i < n-1 por for (i = 0; i < n ?
R=
Se trocarmos for (i = 0 por for (i = 1 o primeiro elemento do vetor não será considerado, e isso pode deixar o menor valor fora de posição. Resultado: o vetor pode não ser ordenado corretamente.
Se trocarmos for (i = 0; i < n-1 por for (i = 0; i < n o índice `i` pode chegar a `n-1`, e nesse caso o laço interno acessaria j = n, o que está fora dos limites do vetor. Isso pode causar erro de execução.

3. Na função selectionsort, troque a comparação v[j] < v[min] por v[j] <= v[min]. A nova função continua correta?
R=
Sim, a função continua correta e acontece a ordenação dos elementos, mas, com <=, o algoritmo pode realizar trocas desnecessárias quando há elementos iguais, o que pode afetar a estabilidade do algoritmo.

5. PIOR CASO. Descreva e analise uma instância (ver rodapé) de pior caso para o algoritmo Selectionsort, ou seja, um vetor v[0..n-1] que leva o algoritmo a executar o maior número possível de comparações.
R=
O pior caso do algoritmo SelectionSort acontece quando o vetor está em ordem decrescente, ou seja, os maiores elementos estão no início e os menores no fim. Nessa situação, o algoritmo precisa realizar o máximo possível de comparações, pois sempre terá que percorrer todo o subvetor restante para encontrar o menor elemento e fazer a troca, essa instância de uma ordem decrescente faz com que o algoritmo caia no pior caso, como por exemplo, um vetor ordenado = {9,8,7,6,5,4,3,2,1}, isso causa o pior caso. Assim, o número de comparações se dá por : n (n-1)/2

.6. MELHOR CASO. Descreva e analise uma instância de melhor caso para o algoritmo Selectionsort, ou seja, um vetor v[0..n-1] que leva o algoritmo a executar o menor número possível de comparações.
R=
No melhor caso com o vetor já ordenado, o algoritmo ainda realiza o mesmo número de comparações, o algoritmo não reconhece que o vetor já está ordenado. Ele ainda percorre todas as posições possíveis em busca do menor elemento, realizando todas as comparações planejadas, como se estivesse lidando com um vetor desordenado. Dessa forma, o SelectionSort não melhora no melhor caso e realiza o mesmo número de comparações: n (n-1)/2 .

7. MOVIMENTAÇÃO DE DADOS. Quantas vezes, no pior caso, o algoritmo Selectionsort copia um elemento do vetor de um lugar para outro? Quantas vezes isso ocorre no melhor caso?
R=

No pior caso, quando o vetor está em ordem decrescente, o algoritmo precisa efetuar a troca em todas as passagens, já que o menor elemento sempre estará o mais distante possível da posição correta. O SelectionSort realiza uma troca por iteração do laço externo, totalizando um número de (n-1) trocas, cada troca feita envolve 3 atribuições totais, armazenar, mover o valor 1 e mover o valor 2. Dessa forma, seria 3 x (n-1) movimentações totais.

Já no melhor caso, o algoritmo ainda realiza as mesmas n−1 troca, pois não verifica se a posição já está correta. Assim, o número de movimentações permanece o mesmo: 3(n−1). Portanto, o SelectionSort movimenta os dados da mesma forma no melhor e no pior caso.