Here’s a detailed list of **viva questions with answers** for your **Selection Sort** code. These are categorized into:

1. ✅ **Expected Questions (based on the code and problem statement)**
2. 🧠 **Basic/Related Conceptual Questions (commonly asked in viva)**

**✅ 1. Expected Questions (Code + Problem Statement)**

**1. What is Selection Sort?**

**Answer:**  
Selection Sort is a comparison-based sorting algorithm. It repeatedly selects the minimum (or maximum) element from the unsorted part of the array and places it at the beginning. This process continues until the array is sorted.

**2. How does your code implement selection sort?**

**Answer:**

* The outer loop iterates over each position in the array.
* The inner loop finds the index of the smallest element in the unsorted part of the array.
* Then it swaps that smallest element with the current position from the outer loop.
* This process repeats until the array is fully sorted.

**3. What is the time complexity of Selection Sort?**

**Answer:**

* **Best case:** O(n²)
* **Average case:** O(n²)
* **Worst case:** O(n²)  
  Selection Sort always compares all pairs regardless of the input's order.

**4. What is the space complexity of Selection Sort?**

**Answer:**  
O(1) — Selection Sort is **in-place** (does not use extra memory).

**5. Does Selection Sort work for negative numbers or duplicate elements?**

**Answer:**  
Yes, Selection Sort works for **any integer values**, including negative numbers and duplicates. It just compares values using <.

**6. What is the role of the swap function in your code?**

**Answer:**  
The swap function exchanges the values at the two indices — it places the smallest element found into its correct position in the sorted part of the array.

**7. Why do we check if (minIndex != i) before swapping?**

**Answer:**  
This check avoids unnecessary swapping when the smallest element is already in the correct position, optimizing the code slightly.

**8. What is the purpose of printVector in the code?**

**Answer:**  
printVector is a helper function that prints the array/vector elements in order, used to show the array before and after sorting.

**9. What would happen if the swap() was not done inside the if (minIndex != i) check?**

**Answer:**  
The algorithm would still work, but it would perform **unnecessary swaps**, reducing efficiency.

**10. Is Selection Sort a stable sorting algorithm?**

**Answer:**  
**No.**  
Selection Sort is **not stable** by default, because swapping can change the relative order of equal elements.

**🧠 2. Basic and Conceptual Viva Questions (with Answers)**

**1. What is a stable sorting algorithm?**

**Answer:**  
A sorting algorithm is **stable** if two equal elements retain their original relative order after sorting.

**2. Give some examples of stable and unstable sorting algorithms.**

**Answer:**

* **Stable:** Bubble Sort, Insertion Sort, Merge Sort
* **Unstable:** Selection Sort, Quick Sort, Heap Sort

**3. What are the advantages and disadvantages of Selection Sort?**

**Answer:**

**Advantages:**

* Easy to implement
* Performs well on small datasets
* Does not require extra space

**Disadvantages:**

* Inefficient for large datasets (O(n²))
* Not stable

**4. When should you use Selection Sort?**

**Answer:**  
When the dataset is small and memory usage is a concern. It’s also useful in teaching sorting algorithms because of its simplicity.

**5. What is an in-place sorting algorithm?**

**Answer:**  
An algorithm that uses only a constant amount of extra memory (O(1)) for sorting. Selection Sort is in-place.

**6. Can you sort in descending order using selection sort?**

**Answer:**  
Yes. Just modify the comparison condition from arr[j] < arr[minIndex] to arr[j] > arr[maxIndex].

**7. How many total swaps are performed in Selection Sort?**

**Answer:**  
At most **(n - 1)** swaps — one for each pass, only when a new minimum is found.

**8. Is it possible to optimize Selection Sort?**

**Answer:**  
Only slightly — by avoiding unnecessary swaps using if (minIndex != i), but the time complexity remains O(n²).

**9. What are the real-world applications of Selection Sort?**

**Answer:**  
Due to its inefficiency, it's rarely used in practice for large datasets, but it's useful in embedded systems with low memory or as an introductory algorithm for learning sorting.

**10. What is the difference between Bubble Sort and Selection Sort?**

**Answer:**

| **Feature** | **Bubble Sort** | **Selection Sort** |
| --- | --- | --- |
| Swaps | More frequent | Fewer swaps |
| Stable | Yes | No |
| Time Complexity | O(n²) | O(n²) |
| Logic | Compare adjacent | Select min in unsorted |

Would you like a **PDF or printable viva revision sheet** based on this for quick access before your practical?