### Consider the following code snippet and answer the questions 1-7:

```cpp

1) int arr[] = {3, 9, 15, 21, 27};

2) int \*ptr = &arr[3];

3) cout << \*ptr << endl;

4) cout << \*(ptr - 2) << endl;

5) cout << \*(++ptr) << endl;

6) cout << \*(ptr--) << endl;

7) cout << --(\*ptr) << endl;

```

\*\*What is the output for line no. 3?\*\*

(A) 3

(B) 9

(C) 21

(D) 27

\*\*What is the output for line no. 4?\*\*

(A) 3

(B) 9

(C) 15

(D) 21

\*\*What is the output for line no. 5?\*\*

(A) 3

(B) 27

(C) 21

(D) Undefined

\*\*What is the output for line no. 6?\*\*

(A) 3

(B) 27

(C) 21

(D) 15

\*\*What is the output for line no. 7?\*\*

(A) 2

(B) 8

(C) 14

(D) 26

\*\*Which of the following is NOT CORRECT for accessing the element in index 2?\*\*

(A) arr[2]

(B) \*(arr + 2)

(C) arr + 2

(D) \*(&arr[2])

\*\*Which of the following is NOT CORRECT about sorting algorithms?\*\*

(A) Quick sort is faster than Bubble sort on average.

(B) Bubble sort requires more swaps than Selection sort.

(C) Merge sort is stable, whereas Quick sort is not always stable.

(D) Bubble sort is the most efficient sorting algorithm for large datasets.

---

### Additional Questions

\*\*Which of the following is NOT CORRECT about binary search?\*\*

(A) Binary search works only on sorted arrays.

(B) The time complexity is O(log n).

(C) Binary search works for both ascending and descending sorted arrays.

(D) Binary search can be used on unsorted arrays.

\*\*Which of the following is CORRECT about recursion?\*\*

(A) Every recursive function must have a base case.

(B) Recursive algorithms are always faster than iterative ones.

(C) Recursion is preferred for tasks requiring memoization.

(D) Recursion cannot be implemented using a stack.

---

Here are the questions with their answers:

\*\*What is the output for line no. 3?\*\*

\*\*Answer:\*\* (C) 21

Explanation: `ptr` points to `arr[3]` which holds the value 21.

\*\*What is the output for line no. 4?\*\*

\*\*Answer:\*\* (B) 9

Explanation: `(ptr - 2)` points to `arr[1]`, which holds the value 9.

\*\*What is the output for line no. 5?\*\*

\*\*Answer:\*\* (D) Undefined

Explanation: After `++ptr`, `ptr` moves to `arr[4]` which is valid. However, subsequent undefined behavior could arise based on memory layout or actions not shown here.

\*\*What is the output for line no. 6?\*\*

\*\*Answer:\*\* (B) 27

Explanation: `ptr--` returns the current value before decrementing. At this point, `ptr` is pointing to `arr[4]`, which is 27.

\*\*What is the output for line no. 7?\*\*

\*\*Answer:\*\* (D) 26

Explanation: After `--(\*ptr)`, the value at `arr[3]` (which is 21 after decrementing `ptr` back) becomes 26.

---

\*\*Which of the following is NOT CORRECT for accessing the element in index 2?\*\*

\*\*Answer:\*\* (C) `arr + 2`

Explanation: `arr + 2` points to the address of `arr[2]`, but does not directly access the value. You would need to dereference it as `\*(arr + 2)`.

\*\*Which of the following is NOT CORRECT about sorting algorithms?\*\*

\*\*Answer:\*\* (D) Bubble sort is the most efficient sorting algorithm for large datasets.

Explanation: Bubble sort is highly inefficient for large datasets compared to algorithms like Quick sort, Merge sort, or Heap sort.

---

### Additional Questions

\*\*Which of the following is NOT CORRECT about binary search?\*\*

\*\*Answer:\*\* (D) Binary search can be used on unsorted arrays.

Explanation: Binary search requires the array to be sorted, either in ascending or descending order.

\*\*Which of the following is CORRECT about recursion?\*\*

\*\*Answer:\*\* (A) Every recursive function must have a base case.

Explanation: Without a base case, recursion would lead to infinite function calls and eventually a stack overflow error.

Here are more questions and answers in a similar format:

---

### Code Snippet

```cpp

1) int arr[] = {12, 23, 34, 45, 56};

2) int \*ptr = arr;

3) cout << \*(ptr + 2) << endl;

4) cout << \*ptr++ << endl;

5) cout << ++(\*ptr) << endl;

6) cout << \*(ptr + 3) << endl;

7) cout << \*(ptr - 1) << endl;

8) cout << --(\*ptr) << endl;

```

\*\*What is the output for line no. 3?\*\*

(A) 12

(B) 23

(C) 34

(D) 45

\*\*Answer:\*\* (C) 34

Explanation: `\*(ptr + 2)` accesses the value at index 2 of the array, which is 34.

\*\*What is the output for line no. 4?\*\*

(A) 12

(B) 23

(C) 34

(D) 45

\*\*Answer:\*\* (A) 12

Explanation: `ptr++` increments the pointer after accessing its current value, so the output is the value at `arr[0]`, which is 12.

\*\*What is the output for line no. 5?\*\*

(A) 24

(B) 23

(C) 22

(D) 34

\*\*Answer:\*\* (A) 24

Explanation: `++(\*ptr)` increments the value pointed to by `ptr` (currently `arr[1]`), changing 23 to 24.

\*\*What is the output for line no. 6?\*\*

(A) 45

(B) 56

(C) Undefined

(D) 34

\*\*Answer:\*\* (B) 56

Explanation: `(ptr + 3)` points to `arr[4]`, which holds the value 56.

\*\*What is the output for line no. 7?\*\*

(A) 12

(B) 23

(C) 34

(D) 45

\*\*Answer:\*\* (B) 23

Explanation: `\*(ptr - 1)` points to the previous element, which is `arr[0]`, holding the value 23.

\*\*What is the output for line no. 8?\*\*

(A) 11

(B) 12

(C) 22

(D) 24

\*\*Answer:\*\* (C) 22

Explanation: `--(\*ptr)` decrements the value pointed to by `ptr` (currently `arr[1]` after previous operations), changing it to 22.

---

### Conceptual Questions

\*\*Which of the following is NOT CORRECT for pointer arithmetic?\*\*

(A) Adding an integer to a pointer moves it to the next memory block.

(B) Subtracting two pointers gives the number of elements between them.

(C) A pointer can point to any arbitrary memory location.

(D) Pointer arithmetic depends on the size of the data type.

\*\*Answer:\*\* (C) A pointer can point to any arbitrary memory location.

Explanation: Pointers must point to valid memory locations, or accessing them may result in undefined behavior.

\*\*Which of the following is CORRECT about array and pointer relationships?\*\*

(A) The array name stores the first element’s value.

(B) The array name acts as a constant pointer to the first element of the array.

(C) The array name can be reassigned to another memory address.

(D) The array elements can be accessed only through the array name.

\*\*Answer:\*\* (B) The array name acts as a constant pointer to the first element of the array.

Explanation: The array name represents the address of the first element of the array, and it cannot be reassigned.

\*\*Which of the following is NOT CORRECT about sorting algorithms?\*\*

(A) Merge sort has O(n log n) worst-case time complexity.

(B) Quick sort’s average time complexity is O(n log n).

(C) Bubble sort always performs O(n log n) comparisons.

(D) Selection sort has O(n²) time complexity in all cases.

\*\*Answer:\*\* (C) Bubble sort always performs O(n log n) comparisons.

Explanation: Bubble sort has O(n²) time complexity in both average and worst cases.

\*\*Which of the following is CORRECT about linear search?\*\*

(A) Linear search is faster than binary search for all datasets.

(B) Linear search can work on both sorted and unsorted data.

(C) Linear search requires a sorted array to function.

(D) Linear search has O(log n) time complexity.

\*\*Answer:\*\* (B) Linear search can work on both sorted and unsorted data.

Explanation: Linear search scans each element one by one, regardless of whether the data is sorted or not.

Here’s another set of questions with answers:

---

### Code Snippet

```cpp

1) int array[] = {10, 20, 30, 40, 50};

2) int \*ptr = array + 1;

3) cout << \*ptr << endl;

4) cout << \*(ptr++) << endl;

5) cout << \*(--ptr + 2) << endl;

6) cout << \*ptr-- << endl;

7) cout << \*(++ptr) << endl;

```

\*\*What is the output for line no. 3?\*\*

(A) 10

(B) 20

(C) 30

(D) 40

\*\*Answer:\*\* (B) 20

Explanation: `ptr` points to `array[1]` after initialization, which holds the value 20.

\*\*What is the output for line no. 4?\*\*

(A) 10

(B) 20

(C) 30

(D) 40

\*\*Answer:\*\* (B) 20

Explanation: `\*(ptr++)` dereferences `ptr` before incrementing it, so it outputs the value of `array[1]`, which is 20.

\*\*What is the output for line no. 5?\*\*

(A) 10

(B) 20

(C) 40

(D) 50

\*\*Answer:\*\* (C) 40

Explanation: `--ptr` moves the pointer back to `array[1]`, and adding 2 moves it to `array[3]`, which holds 40.

\*\*What is the output for line no. 6?\*\*

(A) 10

(B) 20

(C) 30

(D) Undefined

\*\*Answer:\*\* (B) 20

Explanation: `\*ptr--` dereferences the pointer at its current location (`array[1]`) before decrementing it.

\*\*What is the output for line no. 7?\*\*

(A) 10

(B) 20

(C) 30

(D) 40

\*\*Answer:\*\* (B) 20

Explanation: After `++ptr`, the pointer moves back to `array[1]`, which still holds 20.

---

### Conceptual Questions

\*\*Which of the following is NOT CORRECT for a pointer initialized with `int \*ptr = arr;`?\*\*

(A) `ptr + 1` moves to the next integer in the array.

(B) `\*(ptr + 2)` gives the value of the third element.

(C) `ptr++` changes the pointer to the next element.

(D) `ptr = ptr + 2` gives the value of the second element.

\*\*Answer:\*\* (D) `ptr = ptr + 2` gives the value of the second element.

Explanation: `ptr = ptr + 2` changes the pointer's address but doesn’t directly give the value. You would need to dereference it with `\*`.

\*\*Which of the following is CORRECT about memory allocation?\*\*

(A) `malloc` initializes allocated memory to zero.

(B) `calloc` allocates uninitialized memory.

(C) `free` deallocates memory allocated using `malloc` or `calloc`.

(D) `new` can be used in C programming.

\*\*Answer:\*\* (C) `free` deallocates memory allocated using `malloc` or `calloc`.

Explanation: `free` is used to deallocate memory allocated by `malloc` or `calloc`, but neither initializes memory to zero.

\*\*Which of the following is NOT CORRECT about arrays?\*\*

(A) Arrays are stored in contiguous memory locations.

(B) The array size must be defined at compile time in standard C.

(C) Arrays can be resized dynamically in standard C.

(D) Array indexing starts at 0.

\*\*Answer:\*\* (C) Arrays can be resized dynamically in standard C.

Explanation: Arrays in standard C have fixed sizes and cannot be resized dynamically. You would need pointers and dynamic memory allocation for resizing.

\*\*Which of the following is CORRECT about searching algorithms?\*\*

(A) Binary search requires a sorted array.

(B) Linear search requires an array sorted in ascending order.

(C) Binary search is faster than linear search in all cases.

(D) Both binary and linear search have O(log n) time complexity.

\*\*Answer:\*\* (A) Binary search requires a sorted array.

Explanation: Binary search only works with sorted data. Linear search works on unsorted data but has a slower time complexity of O(n).

\*\*Which of the following is CORRECT about sorting?\*\*

(A) Selection sort always performs fewer comparisons than Bubble sort.

(B) Merge sort requires auxiliary space, while Quick sort works in-place.

(C) Bubble sort and Quick sort are stable algorithms.

(D) Insertion sort is inefficient for small datasets.

\*\*Answer:\*\* (B) Merge sort requires auxiliary space, while Quick sort works in-place.

Explanation: Merge sort needs additional memory for temporary arrays, while Quick sort sorts in-place.

---

These questions cover a mix of code-based problems and theoretical concepts in programming, pointers, and sorting/searching algorithms.