Q1. Consider a linked list of n elements, what is the time taken to insert an element after an element pointed by some pointer

O(1)

O(n)

O(n log n)

O(n^2)

Answer: a

Q2. A linear list in which elements are added or deleted at either end but not in the middle is called

Tree

Queue

Stack

dequeue

Answer: d

Q3. Single linked lists are not suitable for

Insertion sort

Binary search

Polynomial operation

Stack implementation

Answer: b

Q4. In a typical implementation the address field of a linked-list node

contains address of next node

contains address of next pointer

may contain NULL value

both (a) and (c)

Answer: d

Q5. Which of the following is false about a circular linked list?

Every node has a successor

Time complexity of inserting a new node at the head of the list is O(1)

Time complexity for deleting the last node is O(n)

None of the mentioned

Answer: d

Q6. Assuming the size of data type int is 4 bytes, what is the size of `int arr[15];`?

15

19

60

11

Answer: c

Q7. Number of push and pop operations required to access the $n/2^{th}$ element of a stack with n elements so that original stack remains same (using another stack)

2×n

4×n

2×n (n−1)

4×n (n−1)

Answer: b

Q8. The best data structure to check whether an arithmetic expression has balanced parentheses is

Queue

Stack

Tree

List

Answer: b

Q9. In doubly linked list, for deletion of a node, the minimum number of pointer modifications required in some cases is

2 pointers

1 pointer

4 pointers

3 pointers

Answer: a

Q10. Convert the given infix expression `A + B * C ^ K` to postfix

ABCK^*+

ABC*K^+

ABCKA*+

AB+CKA.

Answer: a

Q11. Suppose a circular queue of capacity (n–1) elements is implemented with an array of n elements. Initially, REAR = FRONT = 0. The conditions to detect queue full and queue empty are

full: `(REAR+1) mod n == FRONT`; empty: `REAR == FRONT`

full: `(REAR+1) mod n == FRONT`; empty: `(FRONT+1) mod n == REAR`

full: `REAR == FRONT`; empty: `(REAR+1) mod n == FRONT`

full: `(FRONT+1) mod n == REAR`; empty: `REAR == FRONT`

Answer: a

Q12. If the address of `A[1][1]` and `A[2][1]` are 1000 and 1010 respectively and each element occupies 2 bytes, then the array is stored in

row-major order

column-major order

compiler dependent

none of these

Answer: a

Q13. Minimum possible height of a binary tree with 18 nodes is

3

5

4

18

Answer: b

Q14. With respect to graph

i) Adjacency list can represent parallel edges in the graph
ii) Adjacency matrix cannot represent parallel edges
iii) Adjacency matrix representation uses less space compared to Adjacency list for a given graph

Which of the following are correct?

I & II

II & III

I & III

I, II & III

Answer: a

Q15. If space occupied by a null-terminated string "S1" and "S2" in C are m and n respectively, then the space occupied by the string obtained by concatenating "S1" and "S2" is always

less than m+n

greater than m+n

equal to m+n

none of these

Answer: c

Q16. Evaluate the postfix expression `AB^DE*+A-` **for A=3, B=2, D=2 and E=3**

15

12

6

9

Answer: b

Q17. Why do we need to ensure height balancing for a binary search tree with n elements?

To ensure the worst-case search is O(n log n)

To ensure the best-case search is O(1)

To ensure the best-case search is O(n)

To ensure the worst-case search is O(log n)

Answer: d

Q18. Preorder traversal of a binary tree is J I G A B F C H E D, then the root node is

J

A

F

I

Answer: a

Q19. Breadth First Search implementation of a graph uses

Stack

Queue

Linked list

Tree

Answer: b

Q20. A list node representing an arc of a graph requires how many fields?

3

4

1

2

Answer: a

Q21. A sorted (non-descending) singly linked list currently has n nodes. The worst-case time complexity to insert a new node in sorted order is:

O(n)

O(log n)

O(1)

O(n log n)

Answer: a

Q22. Consider the following pseudo-code. What is its average-case complexity (how many times the print statement executes in terms of parameters m and n)?

```
for (i = 0; i < n; i++) {
    for (k = 0; k < m; k++) {
        print(...);
    }
}
```

$\Theta(m \cdot n)$

$\Theta(n)$

$\Theta(m + n)$

$\Theta(\log n)$

Answer: a

Q23. Given `a[6] = {10, 90, 70, 60, 50, 20}`**, after two passes of selection sort the array becomes**

10, 20, 90, 70, 60, 50

10, 20, 70, 60, 50, 90

10, 20, 50, 60, 70, 90

10, 20, 60, 50, 70, 90

Answer: a

Q24. Given `a[6] = {10, 90, 70, 60, 50, 20}`**, after two passes of insertion sort the array becomes**

10, 70, 90, 60, 50, 20

10, 20, 90, 70, 60, 50

10, 20, 70, 90, 60, 50

10, 70, 20, 90, 60, 50

Answer: a

Q25. Determine the sorting algorithm represented by this snippet:

```
void sort() {
  for (i = 1; i < 6; i++) {
    temp = e[i];
    for (j = i; j > 0 && e[j-1] > temp; j--) {
      e[j] = e[j-1];
    }
    e[j] = temp;
  }
}
```

Selection sort

Bubble sort

Heap sort

Insertion sort

Answer: d

Q26. Determine the value returned by this function:

```
int computeSome(int jaadu) {
  if (jaadu > 9)
    return ((jaadu%10)*(jaadu%10)) + computeSome(jaadu/10);
  else
    return jaadu*jaadu;
}
```

generates magic number from "jaadu" and returns it

squares "jaadu" and returns the total sum of its digits

squares only the numbers greater than 9 and returns the value squares the digits of "jaadu" and returns their total sum

Answer: d

Q27. Determine the output when called as

`determine("kitkat","katkit",6,6);`**:**

```
void determine(char* ch1, char* ch2, int len1, int len2) {
  int count1 = 0;
  for (int i = 0; i < len1; i++)
    for (int j = 0; j < len2; j++)
      if (ch1[i] == ch2[j]) count1++;
  cout << len1 << "," << count1;
}
```

6, 2

6, 0

6, 1

6, 3

Answer: d

Q29. Which sorting algorithm is most sensitive to the data being sorted?

Heap Sort

Merge Sort

Quick Sort

Radix Sort

Answer: c

Q30. Number of swaps required to sort n elements using Selection sort (worst case)?

$\Theta(n)$

$\Theta(n \log n)$

$\Theta(n^2)$

$\Theta(n^2 \log n)$

Answer: a

Q31. Which sorting algorithm has the lowest worst-case time complexity?

Quick sort

Bubble sort

Merge sort

Selection sort

Answer: c

Q32. In a binary max-heap containing n numbers, the smallest element can be found in time

O(n)

O(log n)

O(log log n)

O(1)

Answer: a

Q33. After this loop finishes, `j` is Θ(...)

```
for (i = n, j = 0; i > 0; i /= 2)
  j += i;
```

Θ(log n)

Θ(n)

Θ(n log n)

Θ(1)

Answer: b

Q34. An element in array A is a "legend" if it's greater than all elements to its right. The best algorithm to find all legends is

linear time left-to-right pass

linear time right-to-left pass

divide-and-conquer in Θ(n log n)

Θ(n^2) brute force

Answer: b

Q35. Given inputs (4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199) and hash x mod 10, which are true?

(i) 9679, 1989, 4199 hash to same value
(ii) 1471, 6171 hash to same value
(iii) all elements hash to same value
(iv) each element hashes to a different value

1 only

2 only

1 and 2

3 or 4

Answer: c

Q36. Keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted (in that order) into an empty hash table of length 10 using h(k)=k mod 10 and linear probing. The final table is:

```
Index: 0 1  2   3  4  5   6  7   8   9
Value: _ _ 12  13 2  3  23 5  18  15
```

Answer: as shown above

Q37. Hash table of size 7, h(x)=(3x+4)%7, insert 1, 3, 8, 10 via closed hashing. The table becomes:

```
Index: 0 1 2 3 4 5 6
Value: 1 8 _ _ _ _ 3   (after inserting 10 at index 7→wrap to next free slot)
```

Actually final positions:

- 1 → 0
- 3 → 6
- 8 → (0→1)
- 10 → (6→0→1→2) at index 2

Table:

```
0: 1
1: 8
2: 10
3: _
4: _
5: _
6: 3
```

Answer: as shown above

Q38. Determine output of this recursive function when called with 8:

```
int nothing(int something) {
  if (something == 0 || something == 1) return something;
  else return 2*nothing(something-1) + 3*nothing(something-2);
}
cout << nothing(8);
```

1540

1740

1840

1640

Answer: b

Q40. How many distinct binary search trees can be created from 4 distinct keys?

5

14

24

42

Answer: b