This Paper Contains:
7 Pages
10 Questions

Time allowed: Forty five minutes

Marks for this paper total 10.
Students should answer ALL Questions.
Q1. The time complexity of the Insertion Sort algorithm is \((n\) is the size of the input): 

(A) \(O(n\log n)\)
(B) \(O(n^3)\)
(C) \(O(\log n)\)
(D) None of the above.

Q2. The time complexity of the Merge Sort algorithm is \((n\) is the size of the input): 

(A) \(O(\log n)\)
(B) \(O(n)\)
(C) \(O(n\log n)\)
(D) None of the above.

Q3. The time complexity of the Partition method in the Quick Sort algorithm is:

(A) \(O(n)\)
(B) \(O(n^2)\)
(C) \(O(\log n)\)
(D) Constant time.
Q4. The following is the code for the `dequeue()` method for the recursive or linked implementation of a Queue:

```java
public Object dequeue () throws Underflow{
    if (!isEmpty()){
        Object o = first.item;
        <missing line 1.>
        if (isEmpty())
            <missing line 2.>
            return o;
    }
    else throw new Underflow("dequeuing from empty queue");
}
```

The missing lines are:

(A) 1. first = first.successor; 2. last = last.successor;
(B) 1. first = first.successor; 2. last = null;
(C) 1. first = null; 2. last = null;
(D) 1. first.successor = first; 2. last = null;
Q5. The following is the code for the `enqueue` method for an array implementation of a queue.

```java
public void enqueue (Object a) throws Overflow {
    if (!isFull()) {
        // missing line 1.
        // missing line 2.
    }
    else throw new Overflow("enqueuing to full queue");
}
```

The missing lines are:

(A) 1. `items[last]=a;` 2. `last++;`
(B) 1. `last++;` 2. `items[last]=a;`
(C) 1. `last--;` 2. `items[last]=a;`
(D) none of the above.
Q6. The following is the code for the Insertion Sort algorithm:

```java
public void insertionSort(long[] a)
{
    long key;
    int i;
    for (int j = 1; j < a.length; j++)
    {
        key = a[j];
        i = j-1;
        while ((i>-1)&&(a[i]>key))
        {
            <missing line 1.>
            i=i-1;
        }
        <missing line 2.>
    }
}
```

The missing lines are:

(A) 1. a[i]=key; 2. a[i+1]=key;
(B) 1. a[i]=a[i+1]; 2. a[i+1]=key;
(C) 1. a[i+1]=a[i]; 2. a[i]=key;
(D) 1. a[i+1]=a[i]; 2. a[i+1]=key;
Q7. The following is an iterative code for computing the $n$th Fibonacci number:

```java
static int fib(int n)
{
    int f2;
    int f1 = 1;
    int f0 = 1;
    for (int i = 1; i < n; i++) {
        f2 = f1 + f0;
        // <missing line 1.>
        // <missing line 2.>
    }
    return f0;
}
```

The missing lines are:

(A) 1.  f2=f1;  2.  f1=f0;
(B) 1.  f1=f2;  2.  f2=f0;
(C) 1.  f0=f1;  2.  f1=f2;
(D) 1.  f2=f0;  2.  f1=f2;
Q8. Which of the following statements is true?

(A) The worst case complexity of quicksort is $O(n \log n)$ and the average case complexity is $O(n^2)$.
(B) Both the worst case and the average case complexities of quicksort are $O(n^2)$.
(C) The average case complexity of quicksort is $O(n \log n)$ and the worst case complexity is $O(n \log n)$.
(D) The average case complexity of quicksort is $O(n \log n)$ and the worst case complexity is $O(n^2)$.

Q9. The following is the code for the delete operation of the linked implementation of the stack data structure:

```java
public void delete () throws Underflow {
    if (<missing statement>) first = first.successor;
    else throw new Underflow("deleting from empty list");
}
```

The missing statement is (the isEmpty() method checks whether the stack is empty, and the isFull() method checks whether the stack is full):

(A) isEmpty()
(B) isFull()
(C) !isEmpty()
(D) !isFull()

Q10. If there are $n$ objects in a queue, the time complexity to delete the last object is:

(A) $O(n \log n)$
(B) $O(n)$
(C) $O(n^2)$
(D) none of the above.