

Spring 14

Subject : \_\_\_\_\_

Date : \_\_\_\_\_

Q1/a

```
public static void insert (Object[] a,  
int n, obj elem, int idx)
```

```
{  
    if (n == a.length)
```

```
{  
    a = resize (a); // new change  
}
```

```
    a[idx] = elem;
```

```
    n++; // doesn't matter  
}
```

```
public static Objectint[] resize (int a[])
```

```
{  
    int b[] = new int [a.length * 2];  
    int newSize = a.length * 2;
```

```
    Object int b[] = new Object [newSize];
```

```
    for (int i = 0; i < a.length; i++)
```

```
    {  
        b[i] = a[i];
```

```
    }  
    return b;
```

```
}
```

Subject : \_\_\_\_\_

0	1	2		
1	2	3		

$$n = 3 - 1$$

Date : \_\_\_\_\_

Q1

61

```
public boolean remove (Object [] a, int n,
                        Object elem)
{
    int idx = 0;    boolean find = false;
    for (int i = 0; i < n; i++)
    {
        if ((Integer) a[i] == (Integer) elem)
        {
            idx = i;    find = true;
            break;
        }
    }

    if (find == true)
    {
        int end = n - 1;
        while (idx < end)
        {
            a[idx + 1] = a[end];
            end idx++;
        }
        a[n - 1] = null;    return true;
    }
    else return false;
}
```



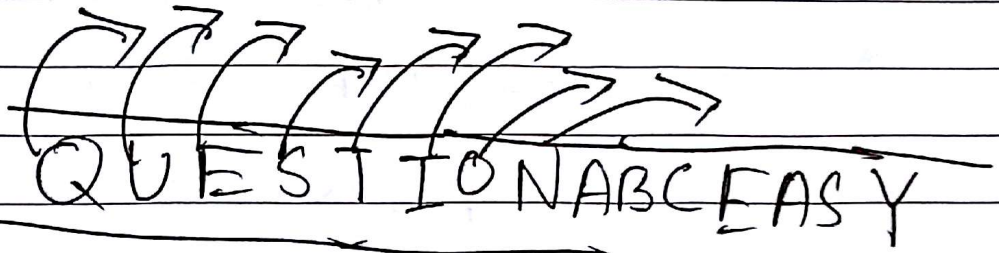
Q1C1

public static int countHowMany (Node head,  
Obj elem)

```
{  
    if (head == null)  
    {  
        return 0;  
    }  
    if ( (Integer)head.data == (Integer)elem )  
    {  
        return 1 + countHM(head.next, elem);  
    }  
    else  
    {  
        return countHM(head.next, elem);  
    }  
}
```

Q2/b

QUE \* S T \* \* IO \* N \* ABC \* EASY



QUESTIONABCEASY

Q2c

```
public void enqueue (Object e) throws Queue
```

```
{ if (Object (size == Object data.length))
    { throw new QueueFullEx();
    }
```

```
if (size == 0) { front = 0; else
    { data[front] = e;
      size++;
    }
    else { rear = (start + size) % data.length;
          data[rear] = e;
          size++;
        }
    }
```

```
public Object dequeue () EmptyQueue
```

```
{ if (size == 0) { throw new EmptyQueue(); }
  else { Object temp = data[front]; data[front] = null;
        front = (front + 1) % data.length; size--;
        return return temp;
      }
}
```



i)

```
public static int[] setup setup(int[] A)
```

```
{
    int min = A[0]; int max = A[0];
    for (int i = 1; i < A.length; i++)
    {
        if (A[i] > max)
        {
            max = A[i];
        }
        if (A[i] < min)
        {
            min = A[i];
        }
    }
}
```

~~int size =~~

~~if min~~

~~size =~~ ~~max - (min) + 1~~

~~int[]~~ B = new int[max];

```
for (int i = 0; i < A.length; i++)
```

```
{
    B[A[i]]++;
}
```

q5

}

Subject : \_\_\_\_\_

Date : \_\_\_\_\_

```
public static boolean search (int B[], int key)
{
    if ( B[key] > 0 )
    {
        return true;
    }
    else return false;
}
```

Subject

Date :

Q5

```
public static int countHowMany(Node root)
```

b1

```
{ if (root == null)
```

```
{ return -1;
```

```
else int L = countHowMany(root.left);
```

```
int R = countHowMany(root.right);
```

```
return L + R;
```

```
}
```



Subject : \_\_\_\_\_

Date : \_\_\_\_\_

0	1	<u>2</u>	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	7	5	3	6	4	8			9	14			12	<u>1</u>

