1. **Tail Recursion**
   fun sum(nil) = 0 | sum(h::t) = h + sum t;

   No. The function is not Tail Recursive as the sum t is called recursively and hence the head of the list has to be placed in the activation record and when the function returns 0 and in turn returns to caller recursively. We are not storing the result of the intermediate state here. It is maintained in the Activation Record. Hence, this is not tail recursive.

   First call → sum([10]::[20,30]) = 10 + sum ([20,30])

   Second call → sum([20]::[30]) = 20 + sum([30])

   Third call → sum([30]::[]) = 30 + sum([])

   Fourth call → sum(nil) = 0

   Return to Third call -> 0

   Return to Second call -> 30 + 0 = 30

   Return to First call -> 20 + 30 = 50

   Return to caller -> 50 + 10 = 60

2. **More Tail Recursion**

   //INPUT: Integers x, y such that x >= y and y > 0

   int gcd(int x, int y) { if (y == 0) return x; else return gcd(y, x % y); }

```
int gcd(int x, int y) {
   int r;
   while(y !=  0) {
      r = x % y;
      x = y;
      y = r;
   }
   return x;
}
```

4. **Memory Allocation**

   Free List: 50, 20, 100, 50, 30, 60

   1. **First Fit:**
      20 → 30, 20, 100, 50, 30, 60
      20 → 10, 20, 100, 50, 30, 60
      30 → 10, 20, 70, 50, 30, 60
      50 → 10, 20, 20, 50, 30, 60
      50 → 10, 20, 20, 30, 60
      45 → 10, 20, 20, 30, 15

2. **Best Fit:**
    20 → 50, 100, 50, 30, 60
    20 → 50, 100, 50, 10, 60
    30 → 20, 100, 50, 10, 60
    50 → 20, 100, 10, 60
    50 → 20, 100, 10, 10
    45 → 20, 55, 10, 10

3. **Worst Fit:**
    20 → 50, 20, 80, 50, 30, 60
    20 → 50, 20, 60, 50, 30, 60
    30 → 50, 20, 30, 50, 30, 60
    50 → 50, 20, 30, 50, 30, 10
    50 → 20, 30, 50, 30, 10
    45 → 20, 30, 5, 30, 10

4. **New Request 30:**
    a. All of the allocation strategies will be able to handle the request.
    First Fit: 10,20,20,15
    Best Fit: 20, 25,10,10
    Worst Fit: 20, 5, 30, 10