problem 1

(a) Divide and conquer Pseudocode

Binary Divide and Conquer (m=2)

```
function divide_binary (v, left, right):
    if left == right:
        if v[left] == 1:
            return left
        else:
            return -1
    mid = (left + right) // 2
    pos = divide_binary (v, left, mid)
    if pos != -1:
        return pos
    return divide_binary (v, mid + 1, right).
```

This algorithm performs a binary divide and conquer without copying the vector.
only index boundaries are passed, and the recursion stops when the subvector size is 1.

m-ary Divide and conquer ($m \geq 2$)

```
function divide_m(v, left, right, m):
  if left == right:
    if v[left] == 1:
      return left
    else:
      return -1
  length = (right - left + 1)//m
  for k = 0 to m-1:
    sub_left = left + k * length
    sub_right = sub_left + length - 1
    pos = divide_m(v, sub_left, sub_right, m)
    if pos != -1:
      return pos
  return -1
```

The vector is divided into m equal parts recursively until each part has size 1.

(b) complexity Analysis of Division

Binary Division ($m=2$)

The recurrence relation is:

$$T(n) = 2T(n/2) + O(1)$$

By the Master Theorem:

$$T(n) = O(n)$$

m-ary Division ($m \geq 2$)

The recurrence relation is:

$$T(n) = mT(n/m) + O(1)$$

This also gives:

$$T(n) = O(n)$$

Therefore, pure divide-and-conquer decomposition has linear complexity.

(c) Collecting the Unique 1

After the vector is fully decomposed into size 1 segments each element is checked once.

Cost of checking leaf nodes:

$$f(n) = O(n)$$

Recurrence relation (binary case):

$$T(n) = 2T(n/2) + O(1) \Rightarrow T(n) = O(n)$$

The final complexity is $O(n)$, not $O(\log n)$, because every element must be examined problem 2. solution

(1) School Multiplication in Base 10

Digits are stored in arrays with the least significant digit first.

```
funtion multiply_ base10(X, Y):
  n = len(X)
  m = len(Y)
  Z = array of size n+m initialized to 0
  for i = 0 to n-1:
  carry = 0
  for j = 0 to m-1:
  temp = Z[i+j] + X[i] * Y[j] + carry
  Z[i+j] = temp mod 10
  carry = temp div 10
  Z[i+m] += carry
  reture Z
```

This method suppots arbitrarily large integers since all operations are digit-based.

(2) Divide and Conquer Multiplication

let: $x = x_1 \cdot 10^{n/2} + x_0$, $y = y_1 \cdot 10^{n/2} + y_0$

then: $x \cdot y = z_2 \cdot 10^n + z_1 \cdot 10^{n/2} + z_0$

where $z_2 = x_1 y_1$; $z_1 = x_1 y_0 + x_0 y_1$; $z_0 = x_0 y_0$

the recurrence relation is:

$$T(n) = 4T(n/2) + O(n)$$

By the Master Theorem:

$$T(n) = O(n^2)$$

(3) karatsuba Algorithm

karatsuba reduces the number of recursive multiplications:

$z_2 = x_1 y_1$; $z_0 = x_0 y_0$; $z_1 = (x_1 + x_0)(y_1 + y_0) - z_2 - z_0$

The recurrence relation is:

$$T(n) = 3T(n/2) + O(n)$$

Thus: $T(n) = O(n^{\log_2 3}) \approx O(n^{1.585})$

(4) computing $\sum_{i=1}^{n} i$ using one multiplication

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

let: $v = [n]$
$w = [n+1]$

Then: $$\sum_{i=1}^{n} i = \frac{1}{2} \cdot \text{mult}(v, w)$$