i)

CSCI 104 Written HW #1

9)

```
void f1(int n)
{
  int i=2;
  while (i<n){
    /* something that take O(1) time*/
    i=i*i;
  }
}
```

Iteration 0	1	2	3	
$i=2$	4	16	$16^2 = 256$	$i_k = 2^{2^k}$

- loop ends when $i_k \geq n$

$$2^{2^k} \geq n$$

$$2^k \geq \log_2 n$$

$$k \geq \log_2(\log_2 n)$$

$$k \geq \log\log n$$

The function runs in $O(\log\log n)$ runtime complexity.

b.

```
void f2(int n)
{
  for(int i=1 ; i<=n ; i++){ ← O(n) complexity
    if((i%(int) sqrt(n)) == 0){ ← happens √n times
      for (int k=0; k < pow(i,3); k++){ ← when true, runs i^3 times
        /* something that takes O(k) time */
      }
    }
  }
}
```

if condition passes for $\sqrt{n}$ multiples of $\sqrt{n}$

$i = j\sqrt{n}$ for $j = 1, 2, \ldots, k$

$\text{pow}(j\sqrt{n}, 3) = (j\sqrt{n})^3 = j^3 n^{3/2}$

$O(n^{3/2})$

complexity: $O(n^{5/2})$

c.

```
for(int i=1; i<=n; i++){ ← O(n) complexity
  for(int k=1; k<=n; k++){ ← O(n) complexity
    if(A[k] == i){ ← worst case check every value, O(n) complexity
      for(int m=1; m<=n; m=m+m){
        // something takes O(1) time //
        // contents of A[] are not changed //
      }
    }
  }
}
```

$m = 1, 2, 4, 8, \ldots$

$m = 2^k$ where $k = 0, 1, 2 \ldots$

$2^k > n$

$k > \log_2 n \rightarrow k > \log n$

complexity: $O(n) \cdot O(n) \cdot O(n) \cdot O(\log n)$

$= O(n^2 \log n)$

d) int f (int n)
{

```
int *a = new int[10];          ← O(1) complexity
int size = 10;                 ← O(1) complexity
for (int i = 0; i < n; i++){   ← O(n) complexity
  if (i == size)               ← takes O(1). worst case: i=size everytime
  {
    int newsize = 3*size/2;
    int *b = new int[newsize];
    for (int j=0; j < size; j++) b[j] = a[j];
    delete [a];
    a = b;
    size = newsize;
  }
  a[i] = i*i;
}
}
```

resizes array to 3/2 of original size

size = 10 → 15 → 22 → 33 → ...

size = $10 * (3/2)^k$ ⇒ O(n)

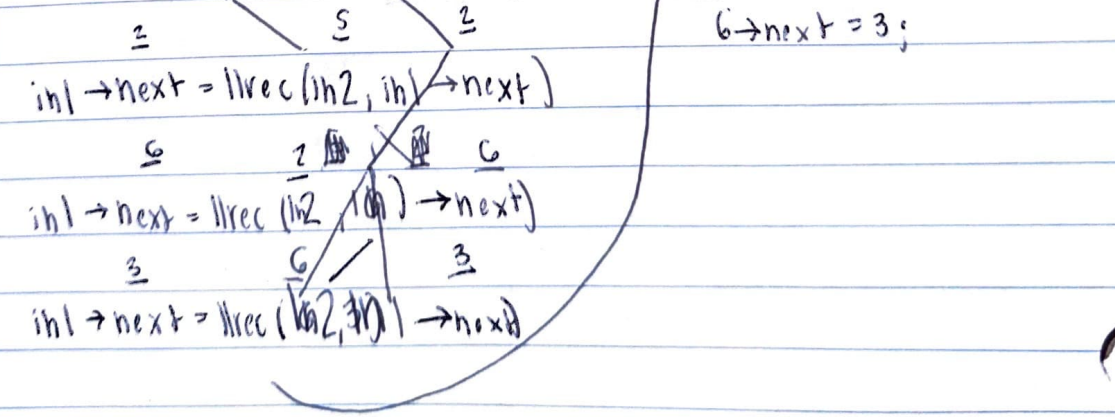
loop total complexity: O(n)

Problem 2

```
struct Node {
  int val;
  Node* next;
};

Node* llrec(Node* in1, Node* in2)
  {
    if (in1==nullptr) {
      return in2;
    }
    else if (in2==nullptr){
      return in1;
    }
    else {
      in1->next = llrec(in2, in1->next);
      return in1;
    }
  }
```

a) in1 = 1, 2, 3, 4 in2 = 5, 6



a. in1 = 1, 2, 3, 4 in2 = 5, 6

1 → 2 = (5 → 6, 2 → 3 → 4);

6 → nullptr = (2 → 3 → 4, 6 → nullptr)

3 → 4 = (6 → nullptr, 3 → 4)

nullptr = (3 → 4, nullptr)

nullptr = 3 → 4;

↓

6 → 3 → 4

↓

2 → 6 → 3 → 4

↓

5 → 2 → 6 → 3 → 4

↓

output: 1 → 5 → 2 → 6 → 3 → 4

b) input:

in1 : null ptr in2 = 2

output: 2