1) a)

```
void rhelp (int m)
{
  if (m=1) return;
  else rhelp (m-1);
}
```

$\sum_{i=1}^{m} O(1)$

→ an initial single call of rhelp will cause it to recurse (n-1) times, performing an $\Theta(1)$ operation, thus rhelps runtime is $\Theta(n)$

```
Void rfunc (int n, m)
{
  if (n<1) return;
  else {
    rhelp (n);
    rfunc (n-m, m);
  }
}
```

rhelp (n); $\Theta(n)$

n, sqrt(n)

→ a single call of rfunc (n, m) passes in $(n, \sqrt{n})$ and calls rhelp once taking $\Theta(n)$ time. rfunc then recurses by $(n-\sqrt{n}, \sqrt{n})$ until $(n<1)$. This means rfunc calls itself $\frac{n}{\sqrt{n}}$ number of times, calling rhelp each recurse. $K = \frac{n}{\sqrt{n}} = \sqrt{n}$

The total runtime of rfunc (n, sqrt(n))

$$T(n) = \sum_{i=1}^{n/\sqrt{n}} \sum_{j=1}^{n} \Theta(1) = \sum_{i=1}^{\sqrt{n}} \Theta(n) = \boxed{\Theta(n^{3/2})}$$

b)

```
int f1 (int* A)
{
  if (x == 0) {
    x = n;
    return 1;
  }
  else if ( x % (int)(sqrt(n)) == 0) {
    for (int i = 0; i ≤ n; i++) {
      for (int j = 0; j ≤ i; j++) {
        O(1)
      }
    }
  }
  else {
    O(1)
  }
  x--;
  return 0;
}
```

→ BEST CASE RUNTIME, $n == 0$: $O(1)$, $\Theta(1)$

WORST CASE

→ BEST CASE SCENARIO, $\Theta(1)$

Because the worst case does not happen every time, but rather only every n^{th} time, total runtime of f1(A) is amortized depending on input size (n).

$$\sum_{i=0}^{n} \sum_{j=0}^{i} \Theta(1) = \Theta(n^2)$$

AMORTIZED RUNTIME: $T(n) = \dfrac{\Theta(n^2) + 1 + \cdots + 1}{n}$

$$\boxed{= \Theta(n)}$$