Aaron Shields

a) void f1 (int n)

int i = 2

while (i<n) {

i = i * i;

}

}

$i \cdot 2 = O(\log_2 n)$

$i \cdot 3 = O(\log_3 n)$

$k \cdot k = k^2$

$k^2 \cdot k^2 = k^4$

$k^{\sqrt{n}} = n$

$\sqrt{n}$

$k^2 = \boxed{O(sqrt(n))}$

---

b) void f2 (int n)

for (int i = 1; i <= n; i++) {

if (i % (int) sqrt (n)) == 0)

for (int k = 0 k < pow(i,3) k++) {

}

$1^3 = 1$

2

}

}

}

O(n) - Outerloop

$O(\sqrt{n})$ - times

$O(\sqrt{n}^3)$ -

$O(n) + O\sqrt{n} + O\sqrt{n}^3$

$\boxed{\text{time complexity} = O(n\sqrt{n})}$

---

c) for (int i = 1; i <= n ; i++) O(n) n+n+n+n

for (int k = 1; k <= n; k++) O(n) k+k+k-

if (A[k] == i)

for (int m = 1 ; m <= n; m = m+m) { O(log n)

// O(1)

}

$n^2 \log n$ = time complexity $\boxed{n^2 \log(n)}$

}

}

---

d) int * a = new int [10]; O(n)

int size = 10;

for (int i = 0; i < n ; i++) { O(n)

O(1)

{

if (i == size)

{

int newsize = 3 · size/2

int b = new int [newsize]

for (int j = 0 ; j < size ; j++) b[j] = a[j];

a = b

a[i] = i * i;

O(size)

O(n-1) + O( )

O(n - 1)

$\boxed{O(n)}$

$\boxed{\text{time complexity} = O(n)}$