Part a

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

Because i is being multiplied by itself within while loop, it will run a total of log(n) times since i is being ~~sta~~ multiplied into itself.

$\Theta(\log n)$

(ex) $n = 64$ When n is 64, the function $\log_2 64 = 5$. runs $\log_2(64)$ times. (5 times)

Part 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++){
        /* O(1) */
      }
    }
  }
}
```

$\Theta(n^{3/2})$

be entered.

This if statement will ~~run~~ $\sqrt{n}$ times.

Inner for loop will run from $k = 0$ to i^3.

i is every multiple of $\sqrt{n}$. (ex) 9, 18, 27, 36...

Part c

```
for (int i = 1; i <= n; i++) {
    for (int k = 1; k <= n; k++) {
        if (A[k] == i) {
            for (int m = 1; m <= n; m = m + m) {
                // O(1) */
            }
        }
    }
}
```

This for loop runs some constant multiplied by n.

This if statement only enters n times @ worst (every int in A[] is 1 & runs n times)

$$\Theta(n(n)) = \boxed{\Theta(n^2)}$$

(n: constant.)

Part d)

Because size has no direct relation with n, I ignore the inner if (i == size) b/c it will not have a significant effect on the runtime. Therefore, when considering the outer loop, we can see runtime is

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