Part (a)

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

Let k be the kth iteration in while loop

k =	0	1	2	3	...	
i =	2	4	16	256	...	n
	2^1	2^2	2^4	2^8		
log i =	1	2	4	8	...	log n

when $\log i = 1$, $k = 0$
$\log i = 2$, $k = 1$
$\log i = 4$, $k = 2$
$\vdots$ $\vdots$
$2^k = \log i$ $\therefore \log\log i = k$

$\therefore$ The answer is $\Theta(\log\log n)$

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++) {
              /* do something that takes O(1) time */
            }
        }
    }
}
```

— $O(n)$ | $\Theta(i^3)$

In the iteration i runtime is $\Theta(i^3)$

i could equal to $\sqrt{n}, 2\sqrt{n}, \ldots, n$

so $(\sqrt{n})^3 + (2\sqrt{n})^3 + \ldots + n^3$
$= (\sqrt{n})^3 \cdot (1 + 2^3 + 3^3 + \ldots + \sqrt{n}^3)$
$= \sqrt{n}^3 \cdot n^2$
$= \Theta(n^{\frac{7}{2}})$

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){
        // do something that takes O(1) time
        // Assume the contents of the A[] array are not changed
      }
    }
  }
}
```

$\sum_1^n \sum_1^n (\Theta(1) + O(\log n))$
$= \sum_1^n \sum_1^n \Theta(1) + \sum_1^n O(\log n)$
$= n^2 + n\log n$
$= \Theta(n^2)$

Part (d)

```
int f (int n)
{
  int *a = new int [10];
  int size = 10;
  for (int i = 0; i < n; i ++)
     {
        if (i == size)
          {
             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;
     }
}
```

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