Problem 3

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

$O(\log n)$; The while loop makes the problem exponentially smaller.

$O(\log n) + O(1) = \boxed{O(\log n)}$

b)
```
void f2 (int n)
  {
    for (int i=1; i<=n; i++) {               } O(n)
        if (i % (int) sqrt(n)) == 0) {
            for (int k=0; k<pow(i,3), k++){  } O(n^3)
                /* do something that takes O(1) time */
            }
        }
    }
  }
```

$(O(n)) \cdot (O(n^3)) = \boxed{O(n^4)}$

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

$[O(\log n)][O(n)][O(n)] = \boxed{O(n^2 \log n)}$

d)
```
int f (int n)
  {
   int *a = new int[10];          } O(1)
   int size = 10;                 } O(1)
   for (int i=0; i<n; i++)        } O(n)
     {
       if (i== size)
       {
         int newsize = 3*size/2;
         int *b = new int[newsize];
O(10) {  for (int j=0; j<size; j++) b[j]=a[j];
         delete [] a;
         a = b;
         size = newsize;
       }
        a[i]=i*i;
```

$[O(10)][O(n)] = \boxed{O(10n)}$