a)

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

] runs $\log_2(n)$ times

$$O(n) = \log_2(n) = \log(n)$$

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

inner loop: $O(n^3)$ for worst case

if $= O(1)$, but will run $\sqrt{n}$ times

outermost loop will run n times.

$$\sum_{i=1}^{n} \sum_{i}^{\sqrt{n}} \sum_{i}^{n^3} 1 = \sum_{i=1}^{n} \sum_{i=1}^{\sqrt{n}} n^3 = \sum_{i=1}^{n} n^{3/2} = n^2 = O(n^2)$$

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
      }
    }
  }
}
```

← m doubles each iteration

inner for loop: $O(\log n)$

if statement = $O(1)$

2nd Outermost for loop : $O(n)$

Outermost for loop : $O(n)$

Overall $\sum_{i=1}^{n} \sum_{i=1}^{n} \log n = O(n^2 \log n)$

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;
     }
}
```

$\sum_{j}^{?} 1$ = will probably evaluate to a constant, which we ignore in $O(n)$ calculation

Since everything in the loop seems to run in constant the runtime seems to be

constant $\sum_{i=0}^{n} i$ (for just the outer loop)

$= Cn^2 = O(n^2)$