Part (a)

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

while loop is same as saying for (int $i = 2$; $i < n$; 2^{2^i}). Runtime will be $i \cdot \theta(1)$.

Thus, we have $2^{2^i} = n$.

$2^i = \log(n)$.

$i = \log(\log(n))$

Thus, runtime is $\log(\log(n)) \cdot \Theta(1) = \boxed{\Theta(\log(\log(n)))}$

Part (b)

$\cancel{n} + \sqrt{n} \cdot n^3 = n^{7/2}$

$\geq$ max when $i = n$

```
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 */
            }
        }
    }
}
```

n Times $\sqrt{n}$ $O(i^3)$ $i \cdot \sqrt{n}$ $k < i^3$

$\sqrt{n}$ $2\sqrt{n}$ $3\sqrt{n}$ ... n

i i_1 i_2 $i_{\sqrt{n}}$

Outer for loop runs n Times. If statement runs $n/\sqrt{n} = \sqrt{n}$ Times. Inner For loop runs i^3 Times, i has a max value of n, Thus This for loop has a runtime of n^3.

Thus, runtime equals $n + \sqrt{n} \cdot n^3$ but we can drop initial n. Thus, runtime equals $\sqrt{n} \cdot n^3 = \boxed{\Theta(n^{7/2})}$

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
      }
    }
  }
}
```

n $\quad$ $\frac{1}{2}n$

$O(n)$

$O(n)$

n

$n \cdot n + n \cdot \frac{1}{2}n$

Outer Two for loops have a runtime $\theta(n)$. If statement has complexity $\theta(n)$. Inner for loop has runtime $\frac{1}{2}n$.

Thus, complexity is:

$$\theta\left(n \cdot n + n \cdot \tfrac{1}{2}n\right) = \theta\left(\tfrac{3}{2}n^2\right) = \boxed{\theta(n^2)}$$

Part (d)

Notice that this code is very similar to what will happen if you keep inserting into an ArrayList (e.g. vector). Notice that this is NOT an example of amortized analysis because you are only analyzing 1 call to the function f(). If you have discussed amortized analysis, realize that does NOT apply here since amortized analysis applies to multiple calls to a function. But you may use similar ideas/approaches as amortized analysis to analyze this runtime. If you have NOT discussed amortized analysis, simply ignore it's mention.

```
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;
    }
}
```

$\Theta(n)$

$i = \left(\frac{3}{2}\right)^{\log(n)-3} \cdot 10$

Outer for loop has complexity $\Theta(n)$. Inner for loop has complexity $\Theta\left(\left(\frac{3}{2}\right)^{\log(n)-3} \cdot 10\right)$. Thus, complexity is:

$$\Theta\left(n + \left(\left(\frac{3}{2}\right)^{\log(n)-3} \cdot 10\right)\right) = \Theta\left(n + \left(\frac{3}{2}\right)^{\log(n)}\right) = \Theta\left(n + \cancel{\frac{3^{\log(n)}}{n}}\right) = \boxed{\Theta(n)}$$