Part (A)

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

Part 1

Part2

Part 1: $\Theta(1)$

Part2: $\Theta(\log n)$, as i doubles in size each loop, so it will take log n loops for i to be $\geq n$

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

```
Part (B)
void f2(int n)
{                                        Part 3
    for(int i=1; i <= n; i++){                           Part 2
        if( (i % (int)sqrt(n)) == 0){                  Part 1
            for(int k=0; k < pow(i,3); k++) {
              /* do something that takes O(1)   Θ(1)
time */
            }
        }
    }
}
```

Part 1: $\Theta(i^3)$, as k must be $\geq$ pow(i,3) before the loop ceases

Part 2: $\Theta(\sqrt{n})$, as the if condition will be true only $\sqrt{n}$ times, as i must be a multiple of $\sqrt{n}$, and $\sqrt{n} \cdot \sqrt{n} = n$, so the final loop $i = n$.

Part 3: $\Theta(n)$, as i must be $\geq n$ to stop the for loop

Part 1 → Part 2: $n\sqrt{n}\sum_{i=1}^{\sqrt{n}} i^3 \approx n\sqrt{n}\left((\sqrt{n})^3 + (\sqrt{n}-1)^3 \ldots + 1\right)$

Part 2 → Part:

$\approx n^{3/2} n^2 = n^{7/2}$ $\quad n\sqrt{n} - \cancel{3n} + \cancel{3\sqrt{n}} \cancel{1}$ $\Theta(n^{7/2})$

$\Theta(n + n^{7/2}) = \boxed{\Theta(n^{7/2})}$

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
      }
    }
  }
}
```

Part 4

Part 3

Part 2

Part 1

$\Theta(1)$

Part 1: $\Theta(\log n)$, as m doubles in size each loop

Part 2: $\Theta(n)$, as worst case senario every element in A is equal to i, which means the if block is called n times

Part 3: $\Theta(n)$, as k must reach n

Part 4: $\Theta(n)$, as i must reach n

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

$$\text{Runtime} = \frac{\#\text{ of actions}}{\#\text{ of steps}}$$

Steps	Actions
1-9	9
10	10
11-14	4
15	15
16-21	7
....	....

Part (D)

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

The # of steps is increases alongside the # of actions, therefore the runtime inside the for loop is essentially $\Theta(1)$ for large n

$$\underbrace{10 \quad 15}_{5} \quad \underbrace{15 \quad 22}_{7} \quad \underbrace{22 \quad 33}_{11} \quad \underbrace{33 \quad 49}_{17} \quad \underbrace{49 \quad 73}_{\cdots\cdots} \quad 109 \quad 163 \quad 244$$

$$\Theta(n+1) = \boxed{\Theta(n)}$$