Problem 3: Runtime Analysis

Saman Nejad

a)

```
void f1 (int n)
{
    int i=2;  ← constant, base case
    while (i<n) {  ← loops n log n times as long as n is greater than i
        // O(1)
        i = i*i;  ← i^2
    }
}
```

Answer $= O(\log n)$

Ex: set n=8   End result

iteration 1: $i=2$, $n=8$
iteration 2: $i=4$, $n=8$
iteration 3: $i=16$, $n=8$

Looped 3 times

$2^n = 8 \Rightarrow 2^3 = 8 \Rightarrow \log_2 8 = 3$

$\therefore$ the complexity is $O(\log n)$

B)

```
void f2(int n)
{
    for (int i=1; i<=n; i++) {  ← loop n times
        if ( i % (int) sqrt(n)) == 0 {  ← perform nested loop if i's remainder is 0 by square root of n
            for(int k=0; k < pow(i,3); k++) {  ← k < i^3, loop k times till k > i
                // O(1)
            }
        }
    }
}
```

$n \cdot n^3 = n^4$

Answer $= O(n^4)$

Ex: set $n=8$ will make the program iterate 512 times, This is because the nested for loop will iterate $i^3 - 1$ till conditions are met. Furthermore, because the if condition is dependent on both $i + n$ makes iterations of it dependent on that fxn, Since outer loop has complexity of n + inner to n^3, we combine to get $O(n^4)$.

C)
```
for (int i=1; i<=n; i++) {          loop n times  = n^2 because n.n, in other words
                                    loop n times    a nested for loop.
  for (int k=1; K <=n ; k++ {       loop log(n) times because the m variable increases
                                    by a base condition of 2 per iteration.
    if (A[k] == 1) {   <- constant
      for (int m=1; m<=n; m=m*m) {    m=2·m
        // O(1)
        // A[] elements stay the same
      }
    }
  }
}
```

Answer => $O(n^2\log(n))$

$\therefore\ n^2 \cdot \log n = n^2 \log n \Rightarrow O(n^2\log(n))$

Starting from the most inner loop, we see that the function loops log n times because the m variable increases by a factor of 2 via each iteration, which essentially is n halving in value. Going to the next 2 for loops, we can see each for loop loops n times. Combining the 2 gives us n^2 + combining the entire program gives us iteration of $n^2 \log(n)$, thus an $O(n^2\log(n))$.

D)
```
int f(int n) {
  int *a = new int [10];   <- constant
  int size = 10 ;  <- constant
  for(int i=0; i<n; i++) {          loop n times
    if (i== size) {   constant
      int newsize = 3*size/2;  constant
      int *b = new int[newsize];   constant
      for (int j=0 ; j< size; j++) b[j] = a[j];     loop size times, so 10 times
      delete [ ] a;  <- constant
      a=b;  <- constant
      size = newsize;  <- constant
    }
    a[i]= i;  <- constant
  }
}
```

Answer => $O(\text{size}*n)$

Looking at the innermost for loop, assuming j < size, we will iterate size (10) times, this is because for loops condition statement to lookup at the outermost for loop, we iterate n times. Therefore this program iterates size* n times, which is $O(\text{size}*n)$ assuming size as a variable & not a constant this is due to the nature of nested for loops as conditions are satisfied. To confirm, we know that insert array complexity is $O(n)$.