problem 3: Runtime Analysis

Part (a)

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

```
ovtput
Influt
          2
                                    2<sup>k-1</sup>
           16
          254
         65236
         46ill ....
                          Puttern = 22k-1
```

$$2^{2^{k-1}} = N$$

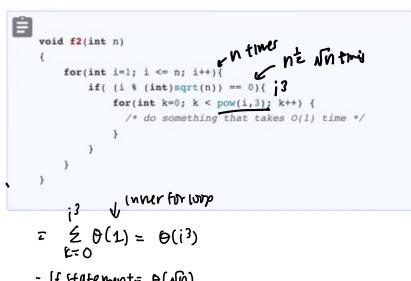
$$\log (2^{2^{k-2}}) = \log(n) \qquad \log(\log(n))$$

$$2^{k-1} = \log(n) \qquad \qquad \leq 0$$

$$2^{k-1} = \log(n) \qquad \qquad \leq 0$$

$$\log(2^{k-1}) = \log(n) \qquad \qquad = \boxed{\Theta(\log(n))}$$

Part (b)



= [f statement= o(Nn)

= Ovter for loop =
$$\stackrel{\circ}{\stackrel{\sim}{=}} (O(1) + O(i^3)) = O(h) + \stackrel{\circ}{\stackrel{\sim}{=}} O(i^3)$$
I from the stidishow

K 3 1 Story whenk= In arbituary k ŊΝ 3N/h 250 [=KNU Stop when i = 12

= 0(N)+O(N=)

∂(N[₹])

```
for(int i=1; i <= n; i++){ > YWW n times
H wolf but // do something that takes O(1) time

The de children // Assume the contents of the
           7 for (int m=1; m <= n; m=m+m) ( -) 30 Nus logn+imes
                // Assume the contents of the A[] array are not changed
              1 M= 1.2,4,8,10,32,2k
                                           Assum A satisfic when i=150 k
                                      LOOPS through like this
                      k=logn
                                           A[t]=i =) A(1]=1 A(2]=1
                                                        A(3]=1 A(...)=1
       inher for 100p executes when k=n times
         dire to trefact we assume that the curter is of the ACI array clocs not change
               E 2 0(1)+ 2 0(1)
             2 0(n) + 2 0 (10gm)
             =\theta(n^2)+\theta(n\log n)
             = (4N)
```

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.

```
A MIN IN times
              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];</pre>
                             delete [] a;
                             a = b;
                             size = newsize;
                                                                                                       5= luq 3 10
                a[i] = i*i; 6 CONHant
                  = \sum_{i=0}^{i=0} \left( \theta(i) + 0 \left( \sum_{j=1}^{i=1} \theta(1) \right) \right)
                   = \sum_{i=0}^{N} 6(1) + \sum_{i=1}^{N} \sum_{j=1}^{N} 6(4)
                   = \theta(N) + |\theta|^{\frac{2}{2}} + |\theta|^{\frac{1}{2}} + |\theta|^{\frac{1}{2}}
                    = \theta(n) + \frac{100^{\frac{2}{10}}}{100^{\frac{2}{10}}} \theta(10^{\frac{2}{10}})
                   =6(N)+10\left(\frac{3}{2}\right)^{10}9_{\frac{3}{2}}^{\frac{1}{10}}
\sum_{i=0}^{n} c^{i} = \frac{c_{n+1-1}}{c_{-1}} = \Theta(c_{n}) = \Theta(n) + \Theta(0) \left(\frac{c_{n}}{c_{n}}\right)
                                             = 6(2n)
N= 1043 19
                                              = 8(11)
```