Problem 4

Angel Flump

Part a

```
for(int i=n-1 ; i>=0 ; i--){
  for(int k=0; k<i*n ; k++){
    // something O(1)
  }
}
```

$n=1$ $i=0$; $i>=0$ → does not run

for ($k=0$; $k<0$; $k++$)

$k=0$ $k<0$

$n=2$ $i=1$ $i>=0$ $i--$ runs twice

2 4 times

$k=0$; $k<2$; $k++$ → runs 2 times

$n=3$ $i=2$ $i>=0$ $i--$ runs 3 times

6 18 times $k=0$ $k<6$ $k++$ runs 6 times

$i=3$ $i>=0$ $i--$ runs 4 times

12 $n=4$ 48 times $k=0$ $k<12$ $k++$ runs 12 times

$n=5$ $i=4$ $i>=0$ $i--$ runs 5 times

100 times $k=0$; $k<80$; $k++$ runs 20 times

$n=6$ $i=5$ → 6 times

180 times $k=0$ $k<30$ → 30 times

inner loop times it will run

$$\sum_{i=0}^{n-1} \sum_{k=0}^{i*n-1} c$$

1

$$= \sum_{i=0}^{n-1} cin = cn \sum_{i=0}^{n-1} i$$

$$= cn \frac{(n-1)(n)}{2} = \theta(n^2)$$

loop does not start at 0 but will always reach it b/c it is decrementing from n-1

Part (b)

```
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*n){
        stmt
      }
    }
  }
}
```

$n=2$ $i=1$ $i<2$ → 1 time

$k=1$ $k<=2$ → 2 times

$n=3$ $i=1$ $i<3$

$k=1$ $k<=3$

∴ worst case scenario: $A[i]=k$: $A[1]=1$, $A[2]=2$, $A[3]=3$

when $i=k$, it will cause the if statement to always happen once (since i increments) therefore happens n times

outer loop $n-1$

middle loop n

if statement: n

inner loop: $\log n$

∴ the inner for loop only runs $n\log n$ times

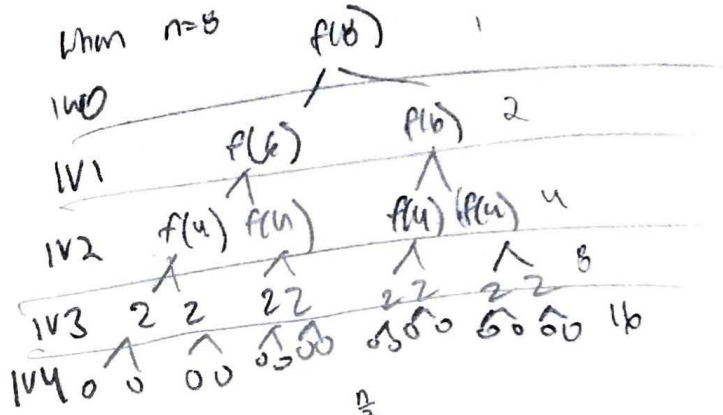
We know that the 2 outer loops will always run. The inner outer loop will run $n-1$ times and the middle outer loop will run at most $n-1$ times since the for loop will not run all n^2 times.

∴ runtime will be $n^2 + n\log n$ → $\theta(n^2)$

Part (c)

```
void f3(int*A, int n){ T(n)
    if (n<=1) return; → Base case only ran once
    else{
        f3(A, n-2) - happens n/2 times
          // O(1)
        f3(A, n-2) - happens n/2 times
    }
}
```

n is not modified inside the if statement, therefore it calls the same recursion tree with the same value of (n-2).



We are summing the work it takes per each level. At most, there can $n/2$ levels each level costs $2^{\frac{n}{2}}$

$$\therefore \sum_{i=0}^{n/2} 2^i = 2^{n/2} \longrightarrow \text{at runtime analysis} \rightarrow O(2^{n/2}) = \Theta(2^n)$$

Problem 4d

Angel Flores

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

The most times we will need to resize is $\log_{3/2} n$ since every time you reach the max size, it makes a new array that is 3/2 times bigger

original size

# of times it would need to be resized

$$\sum_{i=0}^{\log_{3/2} n} 10\left(\frac{3}{2}\right)^i = 10 \sum_{i=0}^{\log_{3/2} n} \left(\frac{3}{2}\right)^i$$

$$= 10\left(\frac{3}{2}\right)^{\log_{3/2} n} = 10n$$ = how long it would take if the if statement ran.

$(n - \log_{3/2} n)$ time it would take if the if statement was not triggered (total time - # of resize required)

$\therefore$ Total runtime $= 10n + n - \log_{3/2} n = 11n + \log_{3/2} n = \theta(n)$.