CSCI 104 HW1

Theodora Heredia

PROBLEM 1

part a)

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

i is initialized to 2 and is squared each iteration of the while loop.

$2 \longrightarrow 4 \longrightarrow 16 \longrightarrow 256 \longrightarrow \ldots$

$ 2^2 \quad 2^{2^2} \quad 2^{2^{2^2}}$

so, $i = 2^{2^k}$, where k is the number of while loop iterations.

the loop runs until $i \geq n$, so until $2^{2^k} \geq n$.

to get rid of the exponent, take log base 2

$\log_2(2^{2^k}) \geq \log_2(n)$ becomes $2^k \geq \log_2(n)$.

do this again to get rid of the second exponent.

$\log_2(2^k) \geq \log_2(n)$ becomes $k \geq \log_2(\log_2(n))$

therefore the runtime becomes $O(\log_2(\log_2(n)))$ because the loop will stop when $k = \log_2(\log_2(n))$

part b)

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

the outer for loop will run from 1 to n, making its runtime $O(n)$.

~~then,~~

then, the conditional check runs if $i \% \sqrt{n} = 0$. so i must be a multiple of $\sqrt{n}$. the check will run n amount of times, the iterations of the outer for loop. so, $\frac{n}{\sqrt{n}} = \sqrt{n}$

this makes the conditional check's runtime $O(\sqrt{n})$.

if the conditional check passes, the inner for loop will run i^3 times, so $O(i^3)$.

part c)

```
for(int i=1; i <= n; i++){
  for(int k=1; k <= n; k++){
    if( A[k] == 1){
      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
      }
    }
  }
}
```

the outer and second for loops will run n amount of times, as indicated by their initialization and condition.

the if statement will only execute once, so its in $O(1)$

the third for loop increments m by m each iteration.

ex, $m = 1, 2, 4, 8, 16, \ldots$

until $m \leq n$

this pattern can be represented by: $2^k \leq n$,

where k is the iteration.

take the log to simplify

$k = \log_2(n)$

therefore, it runs in $O(\log n)$

we multiply the runtimes because everything's nested

$O(n) \times O(n) \times O(1) \times O(\log n)$

which equals $O(n^2 \log n)$, the final runtime

CSCI 104 HW1

Theodora Heredia

part d)

```
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;
      }
}
```

the first for loop runs from 0 to n, so it's $O(n)$

then, if i equals size, the statement including a for loop will execute.

this resizing can be expressed by $10 \times (\frac{3}{2})^k \geq n$, where k is the number of resizings.

ex. $(\frac{3}{2})^1 = 1.5$, $(\frac{3}{2})^2 = 2.25$, $(\frac{3}{2})^3 = 3.375$, $(\frac{3}{2})^4 = 5.06$, etc.

we can see that this grows logarithmically, so it's $O(\log n)$

each resizing allocates an array, which is $O(1)$.

it also copies all elements, so it takes array size amount of time.

ex. 10 elements $\longrightarrow$ 15 $\longrightarrow$ 22 $\longrightarrow$ ... n

this ends up being a geometric series in $O(n)$

$O(n) + O(n) = O(n)$, the total runtime

PROBLEM 2

```
struct Node {
    int val;
    Node*  next;
};

Node* llrec(Node* in1, Node* in2)
{
    if(in1 == nullptr) {
        return in2;
    }
    else if(in2 == nullptr) {
        return in1;
    }
    else {
        in1->next = llrec(in2, in1->next);
        return in1;
    }
}
```

question a)

recursive calls

1st call: llrec(1,5) → returns 1,5,2,6,3,4

2nd call: llrec(5,2) → returns 5,2,6,3,4

3rd call: llrec(2,6) → returns 2,6,3,4

4th call: llrec(6,3) → returns 6,3,4

5th call: llrec(3,4) → returns 3,4

6th call: llrec(4, null) → returns 4

question b)

this will execute the first if statement because in1 = nullptr. so the function will simply return the linked list 2, nullptr