RUNTIME ANALYSIS : FORD CULLEN

Problem 1:

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

While loop interact i < n where iteration 0 is 2 < N, iteration 1 is 4 < N, iteration 2 is 16 < N , iteration 3 is 16*16= 256 < N

RUN TIME = O(log(log n))

Problem 2 (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 */
            }
        }
    }
}
```

I % sqrt(n) == 0 means I / (int)sqrt(N) = factor = sqrt(n)
O(?) = Sum(i to N) * if (i / sqrt(n)) * Sum (0 to (sqrt(n)^3)
Sum (0 to n) * sum (1 to n)
n * integral of i (1 * sqrt(n)) evaluated at n^3 == n^3 * sqrt (n)
RUNTIME T(n) = n ^ 7/2

= O(n) * (n)^3 = O(n^4)
N = 0; O(1)
N = 2;

PROBLEM 3 (C)

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

```
      for(int m=1; m <= n; m=m+m){ // + n log (n)
        // do something that takes O(1) time
        // Assume the contents of the A[] array are not changed
       }
    }
  }
}
RUN TIME =  n^2
Problem 4 (D)

int f (int n)
{
  int *a = new int [10];
  int size = 10;
  for (int i = 0; i < n; i ++)
     {
       if (i == size) // + additionstatement
         {
            int newsize = 3*size/2; //15, 45/2,
            int *b = new int [newsize];
            for (int j = 0; j < size; j ++)
                //O(size) //size is less than N
             b[j] = a[j];
            delete [] a; O(N)
            a = b;
            size = newsize;
         }
       a[i] = i*i; a[10] = 100; 0(1)
     }
}
Size < N
Inner if statement runs for O(size)
OUter for Loop runs for less than < O(n)
Runtime = O(n)
```

PART 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: What linked list is returned if llrec is called with the input linked lists in1 = 1,2,3,4 and in2 = 5,6?

LEVEL 0
Function call llrec(in1, in2) ; In1 != nullptrl; in2 != nullptr;
then in1->next = llrec (in2 (5,6) , in1->next (2,3,4) ==
in1->next = llrec ([5,6] , [2,3,4]; return in1;
LEVEL 1
llrec ([5,6] , [2,3,4]); In1 == 5,6 != nullptrl; in2 == 2,3,4 != nullptr;
then in1->next = llrec (in2 (2,3,4) , in1->next (5,6)) ==
in1->next = llrec ([2,3,4] , 6) ; return in1
LEVEL 2
LIrec ([2,3,4] , [6]) ; In1 ([2,3,4]) != nullptrl; in2 ([6]) != nullptr;
Then in1-> next = llrec (in2 (6) , in1->next (3,4))
in1 -> next = llrec (6, (3,4)) ; return in1
LEVEL 3
LIrec (6 , (3,4)); In1 (6) != nullptrl; in2 (3,4) != nullptr;
Then in1-> next = llrec (in2 (3,4) , in1->next (6))
LEVEL 4
in1->next = llrec ([3,4] , nullptr)); return in1;
llrec ([3,4] , nullptr)); l2 == nullptr; return [3,4] to LEVEL 3

in1(6); in1->next = [3,4]; In1 = (6,3,4)
Return In1 (6,3,4) to LEVEL 2

In1-> next = (6,3,4); In1 (2) = (2,3,4); In1 = (2,6,3,4)
Return in1 to LEVEL 1

in1 = (5,6);in1 (5)->next = (2,6,3,4); in1 =(5,2,6,3,4)
Return in1 to LEVEL 0

In1 = (1,2,3,4); in1 (1) ->next = (5,2,6,3,4); in1 = (1,5,2,6,3,4)
Return In1 ([1,5,2,6,3,4])
152634

Returns [1,4,5,6,2,3,6,3,4]

Question b: What linked list is return if llrec is called with the input linked lists in1 = nullptr and in2 = 2?

```
Llrec (nullptr, 2); in1 == nullprt; return in2 == 2

Returns 2.
```