94/100

# Name: Ghazi Najeeb Al-Abbar

# Student ID: 2181148914

# Assignment #1

**Problem 1.1:** 34

| Expression | Dominating Term(s) | Big O notation |
|---|---|---|
| $5 + 0.001n^3 + 0.025n$ | $0.001n^3$ | $O(n^3)$ |
| $10^{1000} + 2^{1000} + 4^{200} + 1$ | $10^{1000} + 2^{1000} + 4^{200} + 1$ which one? | $O(1)$ |
| $5n + n^{1.5} + 3n \log n$ | $n^{1.5}$ | $O(n^{1.5})$ |
| $\sqrt[3]{n^9} + 10^{10^{10^{10^{10}}}}$ | $\sqrt[3]{n^9} = n^3$ | $O(n^3)$ |
| $n! + 2^n + n \log n$ | $n!$ | $0(n!)$ |
| $n! + n^n + n \log n$ | $n^n$ | $O(n^n)$ |
| $2^{3^n} + 3^{2^n} + 5^n$ | $3^{2^n} = 9^n$ | $O(9^n)$ |
| $\sqrt{n} + log\, n$ | $\sqrt{n}$ | $O(\sqrt{n})$ |
| $0.003 \log n + \log(\log n)$ | $0.003 \log n$ | $O(\log n)$ -1 |
| $log_2 n + log_3 n + log_5$ | $log_2 n$ | $O(log_2 n)$ |
| $\sum_{i=1}^{10} i * n^i$ constant | $10n^{10}$ | $O(n^{10})$ |
| $\prod_{i=1}^{4} n^i$ | $n^{10}$ n is constant | $O(n^{10})$ -1 |

| | | |
|---|---|---|
| $\sum_{i=2}^{10} 30log(i)$ | $\cancel{\sum_{i=2}^{10} 30log(i)}$ 1/10 | O(1) |

**Problem 1.2:**

O(1), O(1), O(logn), O($log_2 n$), O($\sqrt{n}$), O($n^{1.5}$), O($n^3$), O($n^{10}$), O($n^{10}$), O($9^n$), O(n!), O($n^n$)

**Problem 2:** 32

**Function A:**

```
void A(){
    cout<<"Hello World"<<endl; ──────────→ t = 1
}
```

T(n) = 1⇒Big O notation: O(1)

**Function B:**

```
void B(int n){
    int i = 0; ──────────→ t = 1
    int sum = 0; ──────────→ t = 1
    while (i < n){ ──────────→ t = n + 1
        sum += i; ──────────→ t = n
        i += 1; ──────────→ t = n
    }
}
```

T(n) = 1 + 1 + n + 1 + n + n = 3 + 3n ⇒ Big O notation: O(n)

**Function C:**

assume n = m

```
void C(int n, int m){
    int i = 0; ──────────→ t = 1
    int sum = 0; ──────────→ t = 1
    while (i < n){ ──────────→ t = n + 1
        int j = 0; ──────────→ t = n
        while (j < m){ ──────────→ t = n*(m+1)
            sum += j; ──────────→ t = nm
            j += 1; ──────────→ t = nm
        }
        i += 1; ──────────→ t = n
    }
}
```

$O(n^2)$

T(n,m) = 1 + 1 + n + 1 + n + nm + n + nm + nm + n = 3 + 3m + 3nm ⇒Big O notation: O(nm)

**Function D:**

a sum n = m = l

```
void D(int n, int m, int l){         
    int i = 0;                      → t = 1
    int sum = 0;                    → t = 1
    while (i < n){                  → t = n + 1
        int j = 0;                  → t = n
        while (j < m){              → t = n*(m+1)
            int k = 0;              → t = nm
            while( k < l ){         → t = nm*(l+1)
                sum += k;           → t = nml
                k += 1              → t = nml
            }
            j += 1;                 → t = nm
        }
        i += 1;                     → t = n
    }
}
```

T(n,m,l) = 1 + 1 + n + 1 + n + nm + n + nm + nml + nm + nml + nml + nm + n

= 3 + 4n + 4nm + 3nml ⇒Big O notation: O(nml)

$O(n^3)$

**Function E:**

```
int E(){
    int n;                                  → t = 1
    cout<<"Enter␣a␣positive␣number:";       → t = 1
    cin>>n;                                 → t = 1
    int i = 0;                              → t = 1
    int sum = 0;                            → t = 1
    while( i < n ){                         → t = n + 1
        sum += i;                           → t = n
        i += 1;                             → t = n
    }
    return sum;                             → t = 1
}
```

T(n) = 1 + 1 + 1 + 1 + 1 + n + 1 + n + n + 1 = 7 + 3n ⇒Big O notation: O(n)

**Function F:**

```
void F(int n){
    int sum = 0;                → t = 1
    for(int i = 0; i < n; i++){     — Int i = 0: t = 1
                                    — i < n: t = n + 1
                                    — i++: t = n
        for(int j = i; j < n; j++){     — Int j = i: t = n
                                        — j < n: t = (n-1)n/2 + n
                                        — i++: t = (n-1)n/2
            sum += 1;           → t = (n-1)n/2
        }
    }
}
```

T(n) = 1 + 1 + n + 1 + n + $\frac{1}{2}n^2$ - $\frac{1}{2}$n + n + $\frac{1}{2}n^2$ - $\frac{1}{2}$n + $\frac{1}{2}n^2$ - $\frac{1}{2}$n = 3 + $\frac{1}{2}$n + $\frac{3}{2}n^2$

⇒Big O notation: O($n^2$)

**Function G:**

```
void G(int n){
    int sum = 0;  → t = 1
    for(int i = 1; i < n; i = i*2){  ← int i = 1: t = 1 | i < n: t = log(n) + 1 | i = i*2: t = log(n)
        sum += 1;  → t = log(n)
    }
}
```

T(n) = 1 + 1 + log(n) + 1 + log(n) + log(n) = 3 + 3log(n) ⇒Big O notation: O(log(n))

**Function H:**

```
void H(int n){
    int sum = 0;  → t = 1
    for(int i = n; i > 0; i = i/2){  ← int i = n: t = 1 | i > 0: t = log(n) + 1 | i = i/2: t = log(n)
        sum += 1;  → t = log(n)
    }
}
```

T(n) = 1 + 1 + log(n) + 1 + log(n) + log(n) = 3 + 3log(n) ⇒Big O notation: O(log(n))

**Problem 3:** 28

**Function A1:**

```
void S(int n){
    for(int i = 1; i < n; i = i*2){  ← int i = 1: t = 1 | i < n: t = log(n) + 1 | i = i*2: t = log(n)
        cout<<"i␣=␣"<<i;  → t = log(n)
    }
}
void L(int n){
    for(int i = 0; i < n; i = i + 2){  ← int i = 0: t = 1 | i < n: t = ((n + 2)/2) + 1 | i = i + 2: t = ((n + 2)/2)
        cout<<"L.␣i␣=␣"<<i;  → t = ((n + 2)/2)
    }
}
void M(int n){
    while(n > 0){  → t = log5 n + 1 (log base 5 of n)
        cout<<"M.␣i␣=␣"<<i;  → t = log5 n
        n = n/5;  → t = log5 n
    }
}
void A1(int n){
    S(n);
    L(n);
    M(n);
}
```

S(n): T1(n) = 1 + log(n) + 1 + log(n) + log(n) = 2 + 3log(n)

L(n): T2(n) = 1 + $\frac{1}{2}$n + 1 + 1 + $\frac{1}{2}$n + 1 + $\frac{1}{2}$n + 1 = 5 + $\frac{3}{2}$n

M(n): T3(n) = $log\ _5 n$ + 1 + $log\ _5 n$ + $log\ _5 n$ = 1 + 3$log\ _5 n$

∴T(n) = T1 + T2 + T(3) = 2 + 3log(n) + 5 + $\frac{3}{2}$n + 1 + 3$log\ _5 n$ = 8 + 3log(n) + $\frac{3}{2}$n + 3$log\ _5 n$

⇒Big O notation: O(n)

**Function B1:**

```
void B1(int n){
    int sum = 0;  → t = 1
    for(int i = 1; i < n; i = i*2){  → int i = 0: t = 1 | i < n: t = 6 | i++: t = 6
        sum += 100;  → t = 6
        if( sum > 500 ){  → t = 6
            break;  → t = 1
        }
    }
}
```

T(n) = 1 + 1 + 6 + 6 + 6 + 6 + 1 = 27 ⇒Big O notation: O(1) ✓

**Function C1:**

```
int C1(int n){
    if( n == 0)  → t = 1 + n
        return 0;  → t = 1
    return C1(n-1) + n;  → t = n
}
```

show steps for recursion!

T(n) = 1 + n + 1 + n = 2 + 2n ⇒Big O notation: O(n) ✓

**Function D1:**

```
int D1(int n){
    if(n == 0)  → t = 1
        return 0;
    return 2*C1(n-1) + n;  → t = 2 + 2n
}
```

T(n) = 1 + 2 + 2n = 3 + 2n ⇒Big O notation: O(n) ✓

**Function E1:**

```
int E1(int n){
    if(n == 0)  → t = 1
        return 0;
    return C1(n-1) + C1(n-1) + n;  → t = 2 + 2n + 2 + 2n + n
}
```

✗ -1

T(n) = 1 + 2 + 2n + 2 + 2n + n = 5 + 5n ⇒Big O notation: O(n) ✗

**Function F1:**

```
void F1(int n){
    if(n == 0)
        return 0;
    for(int i = 0; i < n; i++){
        cout<<n<<","<<i;
    }
    F1(n-1);
}
```

if(n == 0) → t = 1 + n

return 0; → t = 1

for(int i = 0; i < n; i++){

- int i = 0: t = 1 + n
- i < n: t = n + 1 + n*(n-1)/2 + n
- i++: t = n + n*(n-1)/2

cout<<n<<","<<i; t = n + n(n-1)/2

F1(n-1); → t = n

slow steps for recursion

$$T(n) = 1 + n + 1 + 1 + n + n + 1 + \frac{1}{2}n^2 - \frac{1}{2}n + n + n + \frac{1}{2}n^2 - \frac{1}{2}n + n + \frac{1}{2}n^2 - \frac{1}{2}n + n$$

$$= 4 + \frac{11}{2}n + \frac{3}{2}n^2 \Rightarrow \text{Big O notation: } O(n^2)$$ ✓

**Function G1:**

```
void G1(int n){
    if(n == 0)
        return 0;
    return G1(n/2) + n;
}
```

if(n == 0) → t = 1 + log(n)

return 0; → t = 1

return G1(n/2) + n; → t = log(n)

slow steps for recursion

T(n) = 1 + log(n) + 1 + log(n) = 2 + 2log(n) ⇒Big O notation: O(log(n)) ✓