Assignment #1 - Part [A]

Task 1: Recurrence Equations **[10 Marks]**

Solve the following recurrence equations showing all the steps.

Give asymptotic upper and lower bounds for T(n) in each of the following recurrences.Assume that T(n) is constant for n <= 2. Make your bounds as tight as possible and justify your answer.

(a) $T(n) = 4T(n/4) + 5n$

(b) $T(n) = 4T(n/5) + 5n$

(c) $T(n) = 5T(n/4) + 4n$

(d) $T(n) = 25T(n/5) + n^2$

(e) $T(n) = 4T(n/5) + \lg n$

(f) $T(n) = 4T(n/5) + \lg^5 n\sqrt{n}$

(g) $T(n) = 4T(\sqrt{n}) + \lg^5 n$

(h) $T(n) = 4T(\sqrt{n}) + \lg^2 n$

(i) $T(n) = T(\sqrt{n}) + 5$

(j) $T(n) = T(n/2) + 2T(n/5) + T(n/10) + 4n$

Task 2: Code Analysis **[40 Marks]**

For each of the code snippets provided below, perform a detailed analysis. Specifically identify the key operations and write the time complexity using Big O notation.

1.

```
int secondLargest(int arr[], int n) {
    int largest = INT_MIN,
secondLargest = INT_MIN;
    for (int i = 0; i < n; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] >
secondLargest && arr[i] != largest) {
            secondLargest = arr[i];
        }
    }
    return secondLargest;
}
```

2.

```
int findMissingNumber(int arr[], int n)
{
    int totalSum = n * (n + 1) / 2;
    int arrSum = 0;
    for (int i = 0; i < n - 1; i++) {
        arrSum += arr[i];
    }
    return totalSum - arrSum;
}
```

3.

```
void dijkstra(int graph[V][V], int src)
{
    int dist[V];
    bool visited[V] = { false };

    for (int i = 0; i < V; i++)
            dist[i] = 99999;
    dist[src] = 0;

    for (int count = 0; count < V - 1;
count++) {
        int u = minDistance(dist,
visited);
        visited[u] = true;

        for (int v = 0; v < V; v++) {
            if (!visited[v] && graph[u]
[v] && dist[u] != 99999 && dist[u] +
graph[u][v] < dist[v]) {
                dist[v] = dist[u] +
graph[u][v];
            }
        }
    }
}
```

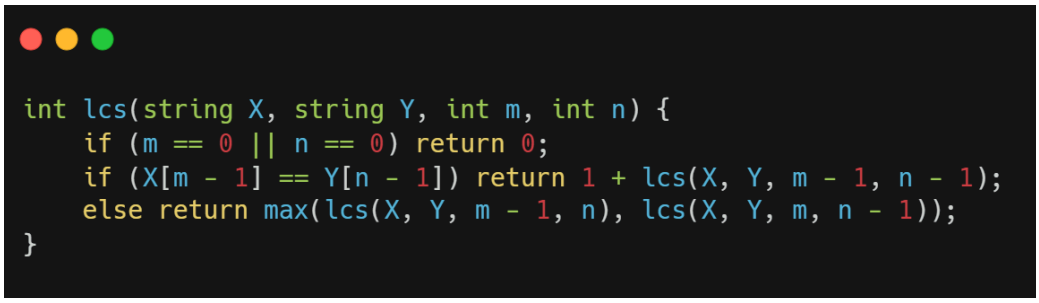
4.

```
void multiplyMatrices(int A[3][3], int
B[3][3], int C[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            C[i][j] = 0;
            for (int k = 0; k < 3; k++)
            {
          C[i][j] += A[i][k] * B[k][j];
            }
        }
    }
}
```

5.

```
int countSetBits(int n) {
    int count = 0;
    while (n > 0) {
        count += n & 1;
        n >>= 1;
    }
    return count;
}
```

6.



```
int lcs(string X, string Y, int m, int n) {
    if (m == 0 || n == 0) return 0;
    if (X[m - 1] == Y[n - 1]) return 1 + lcs(X, Y, m - 1, n - 1);
    else return max(lcs(X, Y, m - 1, n), lcs(X, Y, m, n - 1));
}
```

7.

```cpp
#define ll long long
#define vl vector<ll>
ll n, j;
vl weights;
ll knapsack(ll cap, ll i, vl &selected)
{
    if (i == 0 || cap == 0)
        return 0;
    if (weights[i - 1] > cap)
        return knapsack(cap, i - 1, selected);
    vl picked = selected;
    ll pick = knapsack(cap - weights[i - 1], i - 1, picked) + weights[i - 1];
    ll leave = knapsack(cap, i - 1, selected);
    if (pick > leave)
    {
        picked.push_back(weights[i - 1]);
        selected = picked;
        return pick;
    }
    return leave;
}
```

8.

```cpp
void permute(string s, int l, int r) {
    if (l == r) {
        cout << s << endl;
        return;
    }

    for (int i = l; i <= r; i++) {
        swap(s[l], s[i]);
        permute(s, l + 1, r);
        swap(s[l], s[i]);
    }
}
```

9.

```c
int ternarySearch(int arr[], int l, int r, int x) {
    if (r >= l) {
        int mid1 = l + (r - l) / 3;
        int mid2 = r - (r - l) / 3;

        if (arr[mid1] == x) return mid1;
        if (arr[mid2] == x) return mid2;

        if (x < arr[mid1])
            return ternarySearch(arr, l, mid1 - 1, x);
        else if (x > arr[mid2])
            return ternarySearch(arr, mid2 + 1, r, x);
        else
            return ternarySearch(arr, mid1 + 1, mid2 - 1, x);
    }

    return -1;
}
```

10.

```c
int max(int a, int b) {
    return (a > b) ? a : b;
}
int max(int a, int b, int c) {
    return max(max(a, b), c);
}
int maxCrossingSum(int arr[], int l, int m, int h) {
    int sum = 0;
    int left_sum = INT_MIN;
    for (int i = m; i >= l; i--) {
        sum = sum + arr[i];
        if (sum > left_sum)
            left_sum = sum;
    }
    sum = 0;
    int right_sum = INT_MIN;
    for (int i = m; i <= h; i++) {
        sum = sum + arr[i];
        if (sum > right_sum)
            right_sum = sum;
    }
    return max(left_sum + right_sum - arr[m], left_sum, right_sum);
}
int maxSubArraySum(int arr[], int l, int h) {
    if (l > h)
        return INT_MIN;
    if (l == h)
        return arr[l];
    int m = (l + h) / 2;
    return max(maxSubArraySum(arr, l, m - 1), maxSubArraySum(arr, m + 1, h),
               maxCrossingSum(arr, l, m, h));
}
```

11.

```cpp
struct suffix {
    int index;
    string suffix;
};

bool compareSuffix(suffix a, suffix b) {
    return a.suffix < b.suffix;
}

void buildSuffixArray(string s) {
    suffix suffixes[s.length()];
    for (int i = 0; i < s.length(); i++) {
        suffixes[i].index = i;
        suffixes[i].suffix = s.substr(i);
    }
    sort(suffixes, suffixes + s.length(), compareSuffix);
    for (int i = 0; i < s.length(); i++) {
        cout << suffixes[i].index << " ";
    }
}
```

12.

```cpp
void bellmanFord(int graph[][3], int V, int E, int src) {
    int dist[V];
    for (int i = 0; i < V; i++) dist[i] = INT_MAX;
    dist[src] = 0;

    for (int i = 1; i <= V - 1; i++) {
        for (int j = 0; j < E; j++) {
            int u = graph[j][0], v = graph[j][1], weight = graph[j][2];
            if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
                dist[v] = dist[u] + weight;
            }
        }
    }

    for (int j = 0; j < E; j++) {
        int u = graph[j][0], v = graph[j][1], weight = graph[j][2];
        if (dist[u] != INT_MAX && dist[u] + weight < dist[v]) {
            cout << "Negative weight cycle detected!";
            return;
        }
    }

    for (int i = 0; i < V; i++) cout << i << " " << dist[i] << endl;
}
```

13.

```cpp
void countFrequency(int arr[], int n) {
    cout << "Element Frequency" << endl;
    for (int i = 0; i < n; i++) {
        int count = 1;
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                count++;
                j++;
            }
        }
        cout << arr[i] << " " << count << endl;
    }
}
```

14.

```cpp
bool isPalindrome(int arr[], int size) {
    for (int i = 0; i < size / 2; i++) {
        if (arr[i] != arr[size - i - 1]) {
            return false;
        }
    }
    return true;
}
```

15.

```cpp
void findDuplicates(int arr[], int n) {
    cout << "Duplicates: ";
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                cout << arr[i] << " ";
                break;
            }
        }
    }
    cout << endl;
}
```

16-

```cpp
int getsum(int n,int m){
    int sum = 0;
    for(int i=0;i<n;i++){
        for(int j=m;j>=1;j/=2){
            sum += i+j;
        }
    }
    return sum;
}
```

17-

```
int count(int arr[],int n){
    const int fixed[] = {15,90,48,64,5};
    const int k = 5;
    int count = 0;
    for(int i =0;i<n;i++){
        for(int j=0;j<k;j++){
            if(arr[i]==fixed[j]){
                count++;
                 break;
            }
        }
    }
    return count;
}
```

18-

```
void findPairsWithSum(int arr[], int n, int sum) {
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] + arr[j] == sum) {
                cout << "Pair with sum " << sum << " is: (" << arr[i] << ", " << arr[j] << ")" << endl;
            }
        }
    }
}
```

19.

```
long long power(long long base, int exp){
    if(exp == 1)
        return base;
    if(exp == 0)
        return 1;
    else{
        long long temp = base * base;
        long long ans = power(temp, exp/2);
        if(exp % 2 == 1)
            return base * ans;
        return ans;
    }
}
```

20-

```
void prefix_sum(int arr[], int prefix[], int n) {
    prefix[0] = arr[0];
    for (int i = 1; i < n; i++) {
        prefix[i] = prefix[i - 1] + arr[i];
    }
}
```

Submission Guidelines:

- You should work individually in those tasks
- No late submissions are allowed.
- Cheating is NOT tolerated by any means.
- TAs will grade the assignment out of 60, but this score may be adjusted later for scaling purposes.

Deliverables:

- Submit a scanned file of your **handwritten work** under the name: ***<G#_YourName_YourID>***

Deadline:

- Date: Thursday, 24th of October
- Time: 10:00 PM