Assignment # 1

Time Complexity Analysis

Question 2:

Assuming that each operation takes a single unit of time to execute calculate the time complexity function T(n) and Big-O for the following program fragments:

1:

```
for (i=1;i<=n;++i)
{     cout << i;
      Sum=0;
      for (j=1;j<=i;++j)
      {
            Sum++;
            cout << i;

      }
      cout << Sum;
}
```

Dependent Loops:
Arithmetic series for inner loop:
i= 1, 2, 3, 4……..n
j= 1, 2, 3, 4……..n
Sum = n(n+1)/2

Statement	Number of times executed
i=1	1
i<=n	n+1
++i	n
Cout<<i	n
sum=0	n
j=1	n
j<=i	$n(n+1)/2 +1 =(n^2 + n /2) + 1$
++j	$n^2 + n /2$
sum++	$n^2 + n /2$
Cout<<i	$n^2 + n /2$
Cout<<sum	n
Total	**$6n + 4(n/2) + 4 n^2 +3$**
	$T(n) = 4 n^2 + 8n +3$ **$T(n) = O(n^2)$**

2:

```
for (i=1;i<n;i=i*4)
{
      cout << i;
      for (j=0;j<n;j=j+2)
      {
            cout << j;
            sum++
      }
      cout << sum;
}
```

Independent Loops:
i= 1, 4, 16, ……..n, $n = 4^k$, k= $\log_4(n)$
j= 1, 2, 4, 6……..n , n/2

Statement	Number of times executed
Sum = 0	1
i=1	1
i<n	$\log_4(n)+1$
i=i*2	$\log_4(n)$
Cout<<i	$\log_4(n)$
j=0	$\log_4(n)$
j<n	$\log_4(n)\ (n/2+1) = \log_4(n)\ (n/2) + \log_4(n)$
j=j+2	$\log_4(n)\ (n/2)$
Cout<<j	$\log_4(n)\ (n/2)$
sum++	$\log_4(n)\ (n/2)$
cout<<sum	$\log_4(n)$
Total	**$4\ (n/2) \log_4(n)+ 6 \log_4(n)+3$**
	$T(n) =2n(\log_4(n))+ 6 \log_4(n)+3$ **$T(n) = O(n \log(n))$**

3:

```
sum = 0;
for (i=1;i<=n;i=i*2)
{
      cout << i;
      cout << sum;
      for (j=1;j<=i;++j)
      {
            cout << j;
            cout << "*";
            sum++;
      }
      sum =0;
}
```

Dependent Loops:
Geometric series for inner loop:
i= 1, 2, 4, 8……..n, $n = 2^k$, k= $\log_2(n)$
j= 1, 2, 4, 8……..n,
Sum = $(2^k - 1)$ = (n-1)

Statement	Number of times executed
Sum = 0	1
i=1	1
i<=n	$\log_2(n)+1$
i=i*2	$\log_2(n)$
Cout<<i	$\log_2(n)$
Cout<<sum	$\log_2(n)$
j=1	$\log_2(n)$
j<=i	n
j++	n-1
Cout<<j	n-1
sum++	n-1
Sum = 0	$\log_2(n)$
Total	**4n+ 6$\log_2(n)$**
	T(n) = 4n+ 6$\log_2(n)$ **T(n) = O(n)**

4:

```
for (i=0;i<n;i=i+3)
{
      cout << i;
      for (j=1;j<n;j=j*3)
      {
            cout << j;
            sum++
      }
      cout << sum;
}
```

Independent Loops:
i= 1, 3, 6, 9……..n, n/3
j= 1, 3, 9, 27……..n, $n = 3^k$, k= $\log_3(n)$

Statement	Number of times executed
i=0	1
i<n	n/3+1
i=i+3	n/3
Cout<<i	n/3
j=1	n/3
j<n	$n/3(\log_3(n)+1) = n/3(\log_3(n)) + n/3$
j=j*3	$n/3(\log_3(n))$
Cout<<j	$n/3(\log_3(n))$
sum++	$n/3(\log_3(n))$
Cout<<sum	n/3
Total	**$6(n/3)+ 4(n/3(\log_3(n)) +2$**
	$T(n) = 2n+4/3(n(\log_3(n)))+2$, **T(n) = O(n log(n))**

5:

```
for (i=1;i<=n;++i)
  {       cout << i;
          Sum=0;
          for (j=1;j<=i;++j)
          {
             for (k=1;k<=j;++k)
             {

                  Sum++;
                  cout << i;
             }
         }
         cout << Sum;
  }
```

Statement	Number of times executed
i=0	1
i<n	n+1
++i	n
Cout<<i	n
Sum= 0	n
j=1	n
j<=i	$n(n+1)/2 +1 = n^{2/}/2 +n/2 +1$
++j	$n^{2/}/2 +n/2$
k=1	$n^{2/}/2 +n/2$
k<=j	$((2n^3+ 6n^2 + 4n)/ 12)+ n^2$
++k	$(2n^3+ 6n^2 + 4n)/ 12$
sum++	$(2n^3+ 6n^2 + 4n)/ 12$
Cout<<i	$(2n^3+ 6n^2 + 4n)/ 12$
Cout<<sum	n
Total	**$2/3(n^3)+ 7/2(n^2) + 53/6(n) +3$**
	$T(n) = O(n^3)$

Dependent Loops:
Arithmetic series for inner j loop:
i= 1, 2, 3, 4……..n
j= 1, 2, 3, 4……..n
we can write in formula form

for (i=1 ; i <= n ; i++) 1+ n+1 + n

$$n + \sum_{i=1}^{n}(i+1) + \sum_{i=1}^{n} i$$

for (j=1 ; j <= i ; j++)

Now for each iteration of j against i, k depends on both outer loops

$$\sum_{i=1}^{n} i + \sum_{i=1}^{n}\sum_{j=1}^{i}(j+1) + \sum_{i=1}^{n}\sum_{j=1}^{i} j$$

for (k=1 ; k <= j ; k++)

Anything inside k will run:

$$\sum_{i=1}^{n}\sum_{j=1}^{i} j$$
$$= \sum_{i=1}^{n}\sum_{j=1}^{i} j$$
$$= \sum_{i=1}^{n} \frac{i(i+1)}{2}$$
$$= \frac{1}{2}\left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i\right)$$
$$= \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right)$$
$$= \frac{2n^3+3n^2+n}{12} + \frac{n^2+n}{4}$$
$$= \frac{2n^3+6n^2+4n}{12}$$
$$= O(n^3)$$

6:

```
for (i=1;i<=10;++i)
{     cout << i;
      Sum=0;
}
```

Statement	Number of times executed
i=1	1
i<=10	11
++i	10
cout << i	10
Sum=0	10
Total	**42**
	T(n) = 42 ,T(n) = O(1)

7: Binary Search

```
high = N-1;
low = 0;
index = -1;
while(high >= low)
{
      mid = (high + low)/2;
      if (key == a[mid]) {
            index = mid;
            break;
      }
      else if (key > a[mid])
            low = mid + 1;
      else high = mid - 1;
}
```

Statement	Number of times executed
high = n-1	1
low =0	1
index = -1	1
`high>=low`	$k+1 = \log_2(n) + 1$
`mid=(high+low)/2`	$k = \log_2(n)$
key == a[mid]	$k = \log_2(n)$
Index = mid	$k = \log_2(n)$
break	$k = \log_2(n)$
Total	**$5(\log_2(n)) + 4$**
	$T(n) = 5(\log_2(n)) + 4$ **$T(n) = O(\log(n))$**

Let us assume that N is a power of 2. We can write $N = 2^k$ where k is a non-negative integer. After every iteration, the range is halved as either the low or the high is moved to mid + 1 or mid -1 respectively, effectively reducing the search space to approximately half the original size.

Iterations: 1, 2, 3, ..., k+1.

Search Space: N, N/2, N/4, N/8, …, 1

$$2^k, 2^{k-1}, \ldots, 2^{k-k} = 2^0 = 1$$

$N = 2^k$, , $k = \log_2(n)$

T(n) can vary depending on your assumption of input to algorithm.