Question 1 [10 Marks]

a) Provide a worst-case asymptotic time complexity of the following algorithms by using a suitable asymptotic notation considering a nearest function. Assume that there are no errors/ bugs in the algorithms. Show the meaningful working behind your answer.

Code	Time Complexity

int sum=0;
for (int i = 0; i < n ; i+=2)
{ if (i % 10 == 0)
{
for (int j = 0; j < i; j++)
sum++
}
}
	O(n3)

for (int i = 0; i < n * n; ++i)
{
for (int k = 0; k < i; ++k)
cout<<k;
for (int j = n; j > 0; j--)
cout<<j;
}
	$O(n^4)$

i = n;
while(i > 1)
{
j = i; //this does not start at 0
while (j < n)
{ k = 0;
while (k < n)
k = k + 2;
j = j * 2;
}
i = i / 2;
}
	$O(n\log^2 n)$

Question 2 [12 Marks]

a) Sort all the functions below in increasing order of asymptotic (big-O) growth. If some have the same asymptotic growth, then be sure to indicate that. As usual, lg means base 2. [2 marks]

$$10^n \ , n^{1/3}, 2^{2n}, n^{20}, lgn, n!, \ 2^{2^n}, \sqrt{n}$$

Log ^{n}, $N^{1/3}$, $n^{½}$, n^{20}, 10^{n}, 2^{2n},$2^{2^{n}}$,n!

b) Show that $3n^2 + n + 1$ is $\Theta(n^2)$ by directly finding the constants k, C1, and C2 [5 marks]

If $n>4$ then we have $3n_2 + n+1 < n_2$. So, $n_2 + 4n+17$is $\Omega(n_2)$, taking witnesses $C1 = 1$ and $n_0 = 4$.
If $n>2$ then we have $3n_2 + n+1 > 4n_2$. So, $3n_2 + n+1$ is $O((n_2)$, taking witnesses $C2 = 4$ and $n_0 = 2$.
There 3n2 + n+1 is Θ(n2)

c) Derive the recurrence relation that describes processing time T(n) of the recursive method given in the table below: [5 marks]

```
function finalExam ( n )
        if (n > 1)
             print 'A'
            finalExam( n /3 )
                  for i = 1 to n
                       print 'B'
                 end for
            finalExam ( n /3 )
```

Provide Recurrence Relation of the above pseudocode including base case: [2 marks]

T(n) = 2T(n/3) + O(n),
T(1)= c

What is the runtime of the above function? Express your answer using the big-O notation [3 marks]

SOLUTION: The runtime recurrence satisfies T(n) = 2T(n/3) + O(n), since each for loop takes O(n) time to print all the 'B's. This satisfies the conditions of master theorem with a = 2, b = 3, and d = 1. Since log32 < 1, we get T(n) = O(n)

Question 3 [13 Marks]

Counting Sort is known as a stable sorting algorithm. **table sort** is described as a sorting algorithm that *maintains the position of two equals elements relative to one another.* That is, a sorting algorithm is stable if whenever there are two records R and S with the same key and with R appearing before S in the original list, R will appear before S in the sorted list.

Provide an array of at least 7 elements to demonstrate that the Counting Sort is a stable sort. Also provide complete dry run of counting sort on your selected array. [8 marks]

```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k
 8      C[i] = C[i] + C[i − 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

b) What is the best scenario to use Counting sort? Is Quick Sort a stable sort? Why/Why not? Explain with the help of an example. [3 marks]

When maximum range of number is small. No quick sort is not a stable sort due to the selection of pivot value.

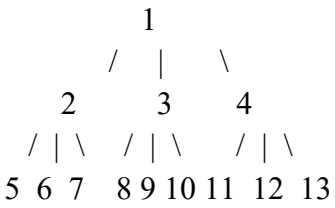
c) For sorting an input list of size *32* by QuickSort recursive algorithm, how many QuickSort calls will be made? Presume the Pivot choice policy is perfect, and QuickPartition always divides the list in half. Explain in one or two lines. [2 marks]

T(32) = 1+2+4+ … +2^log(32) = 2^0 +2^1 +2^2 + … +2^5 = 63, the first call is the driver.

Question 4 [14 Marks]

A d-ary heap is like a binary heap, but (with one possible exception) non-leaf nodes have d children instead of 2 children.

1. Draw d-ary min heap with minimum 2 levels by specifying any value of 'd' other than 2? [5 marks]



2. How would you represent a d-ary heap in an array? [4 marks]

1	2	3	4	5	6	7	8	9	…

3. What will be the formulas for finding the parent and children for a given index? [4 marks]

We can represent a d-ary heap in a 1-dimensional array as follows. The root resides in $A[1]$, its d children reside in order in $A[2]$ through $A[d+1$, their children reside in order in $A[d+2]$ through $A[d^2+d+1]$, and so on. The following two procedures map a node with index i to its parent and to its j-th child (for $1 \le j \le d$), respectively.

```
D-ARY-PARENT( i )
        return  ⌊(i − 2)/d + 1⌋

D-ARY-CHILD( i , j )
        return  d(i + 1) + j + 1
```

To convince yourself that these procedures really work, verify that

```
D-ARY-PARENET(D-ARY-CHILD( i , j )) = i ,
```

for any $1 \le j \le d$. Notice taht the binary heap procedures are a special case of the above procedures when $d = 2$.

4. What is the height of a d-ary heap of n elements in terms of n and d? [1 marks]

Since each node has d children, the height of a d-ary heap with n nodes is $\Theta(\log_d n) = \Theta(\lg d / \lg n)$.

Question 5 [18 Marks]

The following information is based on a piece of text using a set of five different symbols. The frequencies of the symbols in the text are given below:

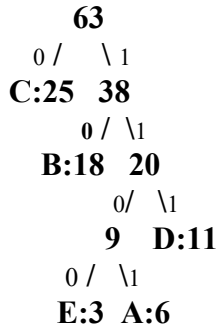
Symbol	Frequency
A	6
B	18
C	25
D	11
E	03

a) What is the minimum number of bits required to store the text using a fixed-length coding scheme? Justify your answer. [3 Marks]

3 bits for each character, total length of the msg is 63 characters,
Fixed length code : 63*3=189

b) What is the minimum number of bits required to store the text using a variable-length coding scheme? You are required to use the Huffman's algorithm learnt in the class. Justify your answer by showing all steps. [10 Marks]

c) Show the final Huffman's tree. [2 Marks]



d) Fill the following table to show Huffman's code of each symbol generated by the Huffman's tree. [3 Marks]

Fill this table	
Symbol	**Variable length code**
A	1101
B	10
C	0
D	111
E	1100