Experiment #2		Student ID	
Date		Student Name	

Experiment Title: Performance analysis of Time and Space Complexity

Aim/Objective: Analysis of Time and Space Complexity of Algorithms

Description: The students will understand and find the Time and Space Complexity of Algorithms

Pre-Requisites

Knowledge: Basics of Data Structures and C Programming, Basic knowledge about algorithms in C and Data Structures.

Tools: Code Blocks / Eclipse IDE

Pre-Lab:

1. During lockdown Mothi gets bored by his daily routine while scrolling YouTube and he found an algorithm that looks different. Mothi is very crazy about algorithms, but as he cannot solve algorithms of multiple loops, he got struck and need your help to find the time complexity of that algorithm

```
Algoritm KLU(int n) {
  int count=0;
  for(int i=0;i<n;i=i*2) {
    for(int j=n;j>0;j=j/3) {
      for(int k=0;k<n;k++) {
        Count++;
      } } } }
```

- **Procedure/Program:**

```
  void KLU(int n){
 int count =0;
For(int i=0; i<n; i=i*2){
For(int j=n; j>0; j=j/3){
for(int k=0; k<n; k++){
  count ++;
  }
  }
  }
```

- **Data and Results:** For (int i=0; i<n; i=i*2) ——— }

  if multiplied by 2 after each iteration, Aussa

  i=

  For (int i=1; i<n; i=i*2) the i is doubles each time which means log-2(n) i to reach (n)

  for (in j<n; j>0; j=j/3) dividing j after each iteration j>0 and the number log 3(n)

  For (int k=0; k<n; k++) k start a to and interments by 1 up to n-1)

- **Analysis and Inferences:**

  , The outer loop runs log-2(n) liy

  , The middle loop runs log 3(n)

  * The inner loop runs n times

  T(n) = (log 2(n)) * (log 3(n)) + n

2. Suresh provided the following recursive algorithm to the students:

recursive algorithm:

```
int custom_recursive_function (int n)
{   if (n <= 1)
        return 1;
else
        return 3 * custom_recursive_function (n-1);
}
```

Determine the time complexity of the custom_recursive_function function.

- **Procedure/Program:**

Experiment #2		Student ID	
Date		Student Name	

```
int custome -mecursive .function(intn)
    {if(n<=1)
          return1;
     else
     return n*custom-recursive- function (n-1);
    }
```

- **Data and Results:**

  ·when n≤1 the function return1

  ·In recursive case n>1 the function a recursive cell to custom- recursive-function(n-1) and multiply]

- **Analysis and Inferences:**

  $T(n)=T(n-1)+O(1)$

  $T(n-1)=T(n-2)+O(1)$

  $T(n-2)=T(n-3)+O(1)$

  $T(n)=T(1)+O(n)=O(n)$

  The recursive cell to custom recursive function (n-1) recursive in relation "—$T(n)=T(n-1)+O(1)$

In-Lab:

1. During the final skill exam, the teacher gave the students a problem to calculate the factorial of a given number n. One student, Ravi, decided to write a recursive algorithm that is intentionaliy inefficient to ensure his approach is unique. Your task is to determine whether Ravi's algorithm for calculating the factorial is correct and to analyze its time complexity.

Here is Ravi's recursive algorithm:

```
int inefficient_factorial(int n) {   if (n == 0)
      return 1;   else if (n == 1)
      return 1;   else
```

Experiment #2		Student ID	
Date		Student Name	

```
    return n * inefficient_factorial(n-1) * inefficient_factorial(n-1);

}
```

Question: Determine if Ravi's algorithm for calculating the factorial is correct and analyz time complexity.

- **Procedure/Program:**

```
int inefficient_factorial(int n){
    if(n==0)
        return 1;
    else if(n==1);
        return 1;
    else
    return* inefficient_factorial(n-1)*
          inefficient_factorial(n-1)
```

- **Data and Results:**

The result of correctens and time complexity analyze

corrections   in correct

Time complexity $O(2^n)$

- **Analysis and Inferences:** Instead of multiplying n by the Factorial of (n-1) once Addiciently the time complexity $o(e^n)$ make the algorithm inefficientfrom moderately large valeces of n

Post-Lab:

1) In the city of KLU, there are numerous street lamps arranged in a straight line. These street lamps are equipped with sensors that can detect their respective positions. Professor Stefan has asked Mothi to find the street lamp that has a specific height using a recursive search algorithm. Mothi has written a recursive algorithm to find the height but needs help in determining the time complexity. Given an array of street lamp heights a, which is sorted in ascending order, write an algorithm to find the position of a lamp with a specific height using a recursive linear search.

```
int recursiveLinearSearch(int a[], int low, int high, int tar)
 {    if (low > high)
                  return -1;
     if (a[low] == tar)
        return low;
     return recursiveLinearSearch(a, low + 1, high, tar);
 }
```

Question: Determine the time complexity of the recursiveLinearSearch function in the best, worst, and average cases.

- **Procedure/Program:**

```
int recursive linear search(int a[],int low,int
                            high,int tar){
  if(low >high){
    return-1;
  }
  if(a[low]==tar)
  {return low;
  }
return recursive linear search(a,low+1,high, tar)
```

Experiment #2		Student ID	
Date		Student Name	

- **Data and Results:**

The resut OF the time complexity of analysis
Bast case O(1)
worst case O(n)
Average case O(n)

- **Analysis and Inferences:** *The reccussive linear search funct has a time complexity of O(n) The function read from best case O(1)

- **Sample VIVA-VOCE Questions (In-Lab):**

1. Define time complexity in the context of algorithms. How does time complexity influ the efficiency of an algorithm?
2. Explain the concept of space complexity. Why is it important to consider space complex along with time complexity when analyzing algorithms?
3. Differentiate between worst-case, best-case, and average-case time complexity. How do each scenario impact the performance of an algorithm?
4. What is Big-O notation? How is it used to express the time complexity of algorithms? Provide an example to illustrate.
5. Explain the process of Merge Sort with a detailed step-by-step example. How does Mer Sort ensure its time complexity of O(nlogn)?

Evaluator Remark (if Any):	**Marks Secured:** 4) **out of 50**
	Signature of the Evaluator with Date

Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Student ID	
Student Name	

e complexity of analysis
o(1)
(n)
o(n)

ive linear search functio
(n) The function reads

hms. How does time complexity influenc

important to consider space complexity
hms?

rage-case time complexity. How does
n?

e time complexity of algorithms?

-by-step example. How does Merge

ed: 4 out of 50

e Evaluator with Date

marks for each experiment.

W-2

① The two amount of time it tak for each statenont to complete

② The function that describes how much memory (spacy)a algorithm recquiires to the quantity.

③ . The Best case the minimum no.of step of an algorithm.

. the function a specrfanes the minimum of an n size of a n.

④ The decribes the worst - case to Provide an upper bound on the algorithm.

⑤ Divide the array int two (nearby) equal hot using merge sort only (rid) using with help of rid.

and

and

e)