



**NED University of Engineering & Technology**  
**Department of Electrical Engineering**

**LAB MANUAL**

**Data Structures and Algorithms**  
**(EE-264)**  
**For**  
**SE Electrical**

**Instructor name:** \_\_\_\_\_

**Student name:** \_\_\_\_\_

**Roll no:** \_\_\_\_\_ **Batch:** \_\_\_\_\_

**Semester:** \_\_\_\_\_ **Year:** \_\_\_\_\_

**To be filled by lab technician**

Attendance: Present out of \_\_\_\_ Lab sessions

Attendance Percentage \_\_\_\_\_

**To be filled by Lab Instructor**

Lab Score Sheet

Roll No.	Rubric based Lab I	Rubric based Lab II	Rubric based Lab III	Rubric based Lab IV	Rubric based Lab V	Rubric based Lab VI	OEL/PBL Rubric Score A	Final LAB Rubric Score B	Attendance Percentage C	Final weighted Score for MIS System [10(A)+10(B)+5(C)]/25 Round to next higher multiple of 5

EE-264 DSA Rubric Based Labs: 3, 4, 5, 8, 9, 10

Note: All Rubric Scores must be in the next higher multiple of 5 for correct entry in MIS system.

# LAB MANUAL

## **Data Structures and Algorithms** **(EE-264)** **For** **SE Electrical**

Content Revision Team: **Dr. Riaz Uddin** and **Dr. Mirza Muhammad Ali Baig**

Last Revision Date: **28<sup>th</sup> December 2020**

Approved By

**The Board of Studies of Department of Electrical Engineering**

---

---

---

---

---

---

## CONTENTS

**CLO (Psychomotor, Level 3):** Students should build various basic algorithms, analyze empirically their growth of computational time and formulate object-oriented programming to develop basic data-structures

**CLO (Affective, Level 2):** Students should build various basic algorithms, analyze empirically their growth of computational time and formulate object-oriented programming to develop basic data-structures

S. No.	Date	Title of Experiment	Total Marks	Signature
1		Introduction to programming with <i>Python</i>		
2		Developing and executing algorithms using <i>Python</i>		
3		To analyze the efficiency of sorting algorithms		
4		To develop and apply the recursive divide and conquer approach in sorting		
5		Extending the divide-and-conquer approach on sorting and searching problems		
6		Apply Asymptotic Notations to the Sorting Algorithms.		
7		Introduction to object oriented programming.		
8		Develop a system which can perform basic banking related tasks		
9		To implement fundamental data structures in Python (using list) a) Stack b) Queue		
10		Accomplish the following open-ended tasks:  Using Node class, develop 1. Stacks 2. Queue		
11		Accomplish the open-ended task:  Using Node class, develop Singly connected linked-list		

## Laboratory Session No. 01

### Objective:

*To get introduced with fundamentals of programming with Python*

### Outcomes:

By the end of this lab, student should be able to

a) Correctly code algorithms in python which may include

- 1) Loops
- 2) Conditions
- 3) Lists
- 4) User defined functions
- 5) Importing libraries to program

### 1) Loops:

In *Python*, *for* and *while* loops follows the following syntax.

**WHILE LOOP:-**

```
In [10]: a,b=0,1
         while b<1000:
         |     print (b)
         |     a,b=b,a+b
```

```
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
```

*while loop in Python*

## ***FOR LOOP:-***

---

```
In [1]: for i in range(0,10):  
        print(i)  
        print('Marwa Ashfaq\n')
```

```
0  
Marwa Ashfaq  
  
1  
Marwa Ashfaq  
  
2  
Marwa Ashfaq  
  
3  
Marwa Ashfaq  
  
4  
Marwa Ashfaq  
  
5  
Marwa Ashfaq  
  
6  
Marwa Ashfaq  
  
7  
Marwa Ashfaq  
  
8  
Marwa Ashfaq  
  
9  
Marwa Ashfaq
```

---

***for loop in Python***

---

## 2) Conditions:

---

```
In [2]: a=int(input('please enter a value:'))
        please enter a value:100
```

```
In [3]: if a<12:
        print('value is less than 12')
        else:
        print('value is greater than 12')
        value is greater than 12
```

---

*if-else condition in Python*

---

## 3) Lists:

A list is created by placing all items in “square brackets []”.  
Elements can be added/appended in a list as well.

---

```
In [1]: #Defining a list
        list=[0,1,2,3]
```

```
In [2]: list
```

```
Out[2]: [0, 1, 2, 3]
```

```
In [3]: #Adding elements in a list
        list=list + [4]
```

```
In [4]: list
```

```
Out[4]: [0, 1, 2, 3, 4]
```

```
In [12]: #Appending a List
        list.append(5)
```

```
In [11]: list
```

```
Out[11]: [0, 1, 2, 3, 4, 5]
```

---

*list example*

---

#### 4) User defined Functions:

Functions in *Python* can be created by using the syntax shown below. A function is a block of code which only runs when it is called. Defining and calling a function are explained as follows:

---

```
In [1]: #Defining Functions
def fib(n):
    a, b = 0,1
    while b<n:
        print(b)
        a,b = b, a+b
```

```
In [2]: def fib2(n):
        result=[]
        a,b = 0,1
        while a<n:
            result.append(a)
            a,b = b, a+b
        return result
```

```
In [3]: fib(100)
```

```
1
1
2
3
5
8
13
21
34
55
89
```

```
In [4]: fib2(100)
```

```
Out[4]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

---

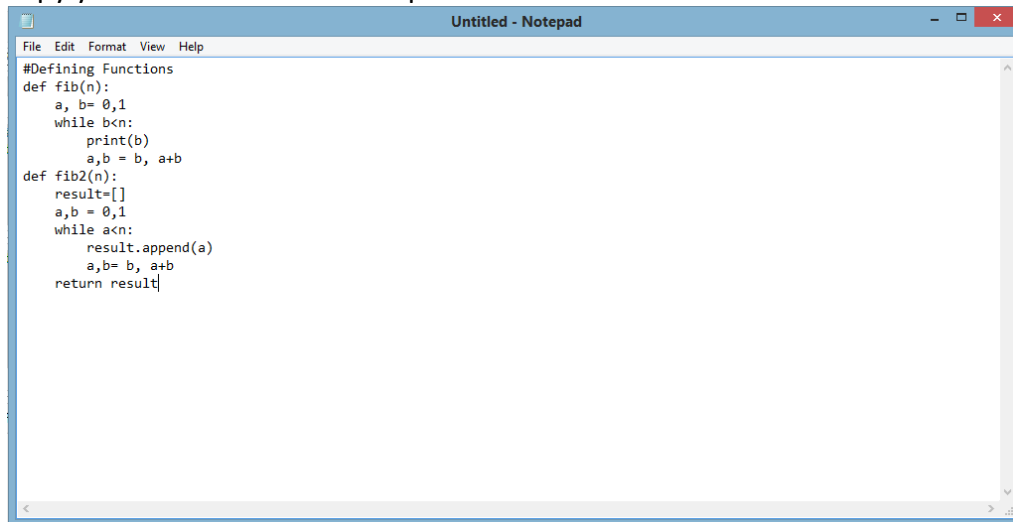
***Working with functions in Python***

---



### ***Saving and Importing user-defined function to a program:***

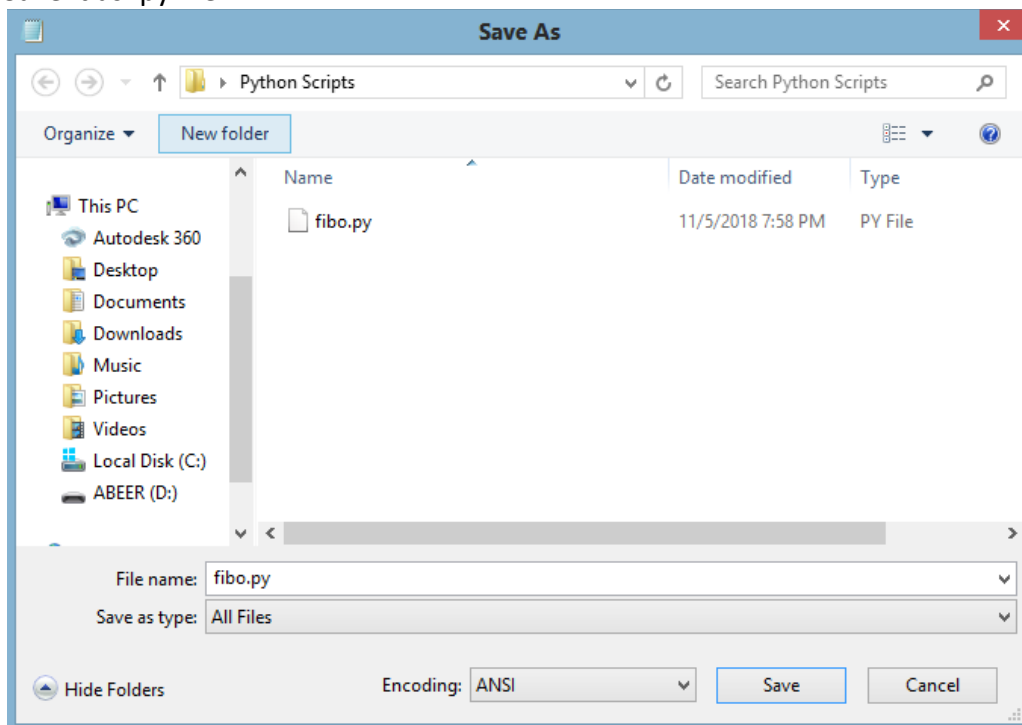
- Copy your desired code in notepad.



The screenshot shows a Notepad window titled "Untitled - Notepad". The menu bar includes File, Edit, Format, View, and Help. The code content is as follows:

```
#Defining Functions
def fib(n):
    a, b = 0,1
    while b<n:
        print(b)
        a,b = b, a+b
def fib2(n):
    result=[]
    a,b = 0,1
    while a<n:
        result.append(a)
        a,b= b, a+b
    return result
```

- Save it as .py file.



- Change its extension from.txt to .py.
- Import as follows:

---

```
In [2]: import fibo
```

```
In [3]: fibo.fib(100)
```

```
1
1
2
3
5
8
13
21
34
55
89
```

```
In [4]: from fibo import fib2
```

```
In [5]: fib2(100)
```

```
Out[5]: [0, 1, 1, 2, 3, 5, 8, 13,
```

```
In [6]: from fibo import fib
```

```
In [7]: fib(100)
```

```
1
1
2
3
5
8
13
21
34
55
89
```

```
In [8]: f=fib2(1000)
```

```
In [9]: f
```

```
Out[9]: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987]
```

---

***Calling user-defined function in Python***

---

## 5) Importing libraries to program:

**Python library** is a collection of functions and methods that allows you to perform lots of actions without writing your own code. For importing libraries, the “import” command is used.

Once the library is imported, its different functions can be called. Following is an example which makes use of a library

---

```
In [1]: import math
```

```
In [2]: math.sqrt(121)
```

```
Out[2]: 11.0
```

```
In [4]: math.factorial(6)
```

```
Out[4]: 720
```

```
In [5]: math.acos(1)
```

```
Out[5]: 0.0
```

```
In [6]: math.asin(1)
```

```
Out[6]: 1.5707963267948966
```

```
In [8]: math.pi
```

```
Out[8]: 3.141592653589793
```

---

***Making use of libraries in Python***

---

## Laboratory Session No. 02

### Objective:

*To developing and execute basic algorithms using Python*

### Outcomes:

By the end of this lab, student should be able to implement following exercises in *Python*

- 1) Write a program which could generate the following pattern. [hint: use 'end' option in print command]

```
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```

- 2) Write a program which can generate the following

*Input a number: 10*  
*10 x 1 = 10*  
*10 x 2 = 20*  
*10 x 3 = 30*  
*10 x 4 = 40*  
*10 x 5 = 50*  
*10 x 6 = 60*  
*10 x 7 = 70*  
*10 x 8 = 80*  
*10 x 9 = 90*  
*10 x 10 = 100*

- 3) Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error message. If the score is between 0.0 and 1.0, print a grade using the following table:

$\geq 0.9$  A  
 $\geq 0.8$  B  
 $\geq 0.7$  C  
 $\geq 0.6$  D  
 $< 0.6$  F

Enter score: 0.95

A

Enter score: perfect

Bad score

Enter score: 10.0

Bad score

Enter score: 0.75

C

Enter score: 0.5

F

- 4) Re-write the above program using functions
- 5) Write a Python function to calculate the factorial of a number. [use recursive approach]
- 6) Write a function which can search for an entry in a list. Also show the entry count in the list.
- 7) Develop code in python for sorting a list using selection sort approach. In selection sort you find the minimum value first and place it at the end of the list.

## Laboratory Session No. 03

Objective:

*To analyze and evaluate experimentally the running time of*

- 1) *Selection Sort*
- 2) *Bubble Sort*
- 3) *Insertion Sort*

### Special Instructions

- 1) You are supposed to translate pseudocodes of the above mentioned codes in *Python*.
- 2) Show in tabulated form, the analytical expressions of computational times for the above algorithms based on RAM model
- 3) Now, evaluate the run time using *time* library functions
- 4) You would need to discuss the average run time of each algorithm for best and worst cases

### 1. Selection Sort:

---

```
for i = 1 to A.length
```

```
    min_pos = i
```

```
    for j = i+1 to length_of_list
```

```
        if list[min_pos] > list[j]
```

```
            min_pos = j
```

```
    temp = list[i]
```

```
    list[i] = list[min_pos]
```

```
    list[min_pos] = temp
```

---

*Pseudocode of Selection Sort*

---

---

---

```
def Selection_Sort(M):  
    for i in range(0,len(M)):  
        min_pos=i  
        for j in range (i+1,len(M)):  
            if M[min_pos]> M[j]:  
                min_pos=j  
        temp=M[i]  
        M[i]=M[min_pos]  
        M[min_pos]=temp  
    return(M)
```

```
Z=[10,12,6,89,43]
```

```
Selection_Sort(Z)
```

```
[6,10,12,43,89]
```

```
Out[2]:
```

```
[6, 10, 12, 43, 89]
```

---

---

*Python Code*

---

---

### Analysis of Selection Sort

Pseudocode		Cost	Time (Worst)	Time (Best)
<b>1</b>	for i=1 to length_of_list	C <sub>1</sub>	n+1	n+1
<b>2</b>	min_pos=i	C <sub>2</sub>	n	n
<b>3</b>	for j=i+1 to length_of_list	C <sub>3</sub>	$\sum_{j=1}^n j = \frac{n(n+1)}{2}$	$\sum_{j=1}^n j = \frac{n(n+1)}{2}$
<b>4</b>	if list[min_pos] > list[j]	C <sub>4</sub>	$\sum_{j=1}^n (j-1) = \frac{n(n-1)}{2}$	$\sum_{j=1}^n (j-1) = \frac{n(n-1)}{2}$
<b>5</b>	min_pos = j	C <sub>5</sub>	$\sum_{j=1}^n (j-1) = \frac{n(n-1)}{2}$	0
<b>6</b>	else	0	n	n
<b>7</b>	temp = list[i]	C <sub>7</sub>	n	n
<b>8</b>	list[i] = list[min_pos]	C <sub>8</sub>	n	n
<b>9</b>	list[min_pos] = temp	C <sub>9</sub>	n	n
<b><i>Analysis of Selection Sort</i></b>				



## Run time of Selection Sort

```
def Selection_Sort(M):
    for i in range(0,len(M)):
        min_pos=i
        for j in range (i+1,len(M)):
            if M[min_pos]> M[j]:
                min_pos=j
        temp=M[i]
        M[i]=M[min_pos]
        M[min_pos]=temp
    return(M)
Z=[10,12,6,89,43]
```

```
Selection_Sort(Z)
```

```
[6, 10, 12, 43, 89]
```

```
import time
a=time.time()
Selection_Sort(list(range(6000,1,-1)))
b=time.time()
c=b-a
print('run time=',c)
```

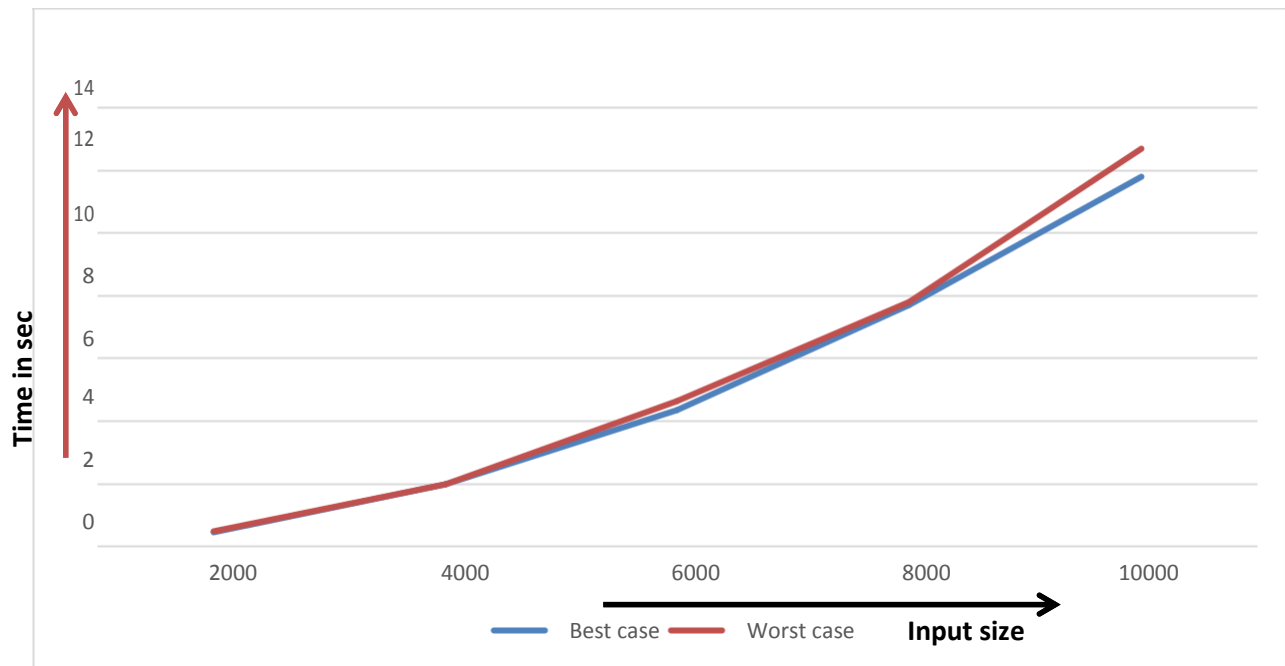
*Python implementation for runtime assessment for a **worst case***

### Tabulated run-time of Selection Sort:

After experimenting with the python code for five different sizes of inputs, following run-times were recorded.

S. No	Number of elements in array	Time of Best case(sec)	Time of worst case(sec)
1	2000	0.4653	0.4973
2	4000	1.9898	1.9856
3	6000	4.3554	4.6329
4	8000	7.7099	7.7937
5	10000	11.792	12.696

### Growth Plot:



### Note:

Student is supposed to repeat similar exercise, for *bubble* and *insertion sort* algorithms.

*\*keep in mind that your reading will depend on your computer's speed. The above tables and graphs are just for the verification of concepts*

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

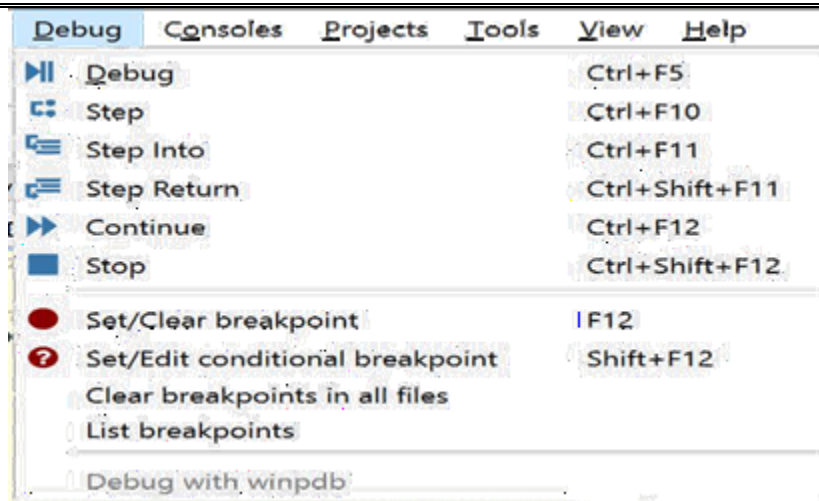
## Laboratory Session No. 04

### Objective:

***To develop and apply the recursive divide and conquer approach in sorting (using debugging tools in Python)***

### Debugging:

Debugging is a process which involves identifying a problem, isolating the source of the problem and then either correcting the problem or determining a way to look around it. In debugging process, we run the program step-by-step and keep a look on the variables. To invoke the option for debugging in *spyder IDE* we take following steps:



***Debugging tools in Spyder***

Here, the *DEBUG* option, starts debugging. The *STEP* option, steps to next line of the code. The *STEP INTO* option, takes you inside the function's body. The *STEP RETURN* option, steps to return the function call. The *CONTINUE* option, continues with debugging mode. The *STOP* option, forces the current debugging to stop.

### Merge-sort Algorithm:

Merge Sort is based on the approach of *Divide and Conquer*. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves.

Following is the python-code for mergesort algorithm :

---

```
def MergeSort(A):
    n=len(A)
    s=list( )
    if n==1:
        s=A
    else:
        a=(n//2)
        s1=MergeSort(A[0:a])
        s2=MergeSort(A[a:n])
        s=merge(s1,s2)
```

```
def merge(A,B):
    n1=len(A)
    n2=len(B)
    A=A+[float('inf')]
    B=B+[float('inf')]
    i=0
    j=0
    l=list( )
    for k in range(0,n1+n2):
        if A[i]<=B[j]:
            l=l+[A[i]]
            i=i+1
        else:
            l=l+[B[j]]
            j=j+1
    return l
```

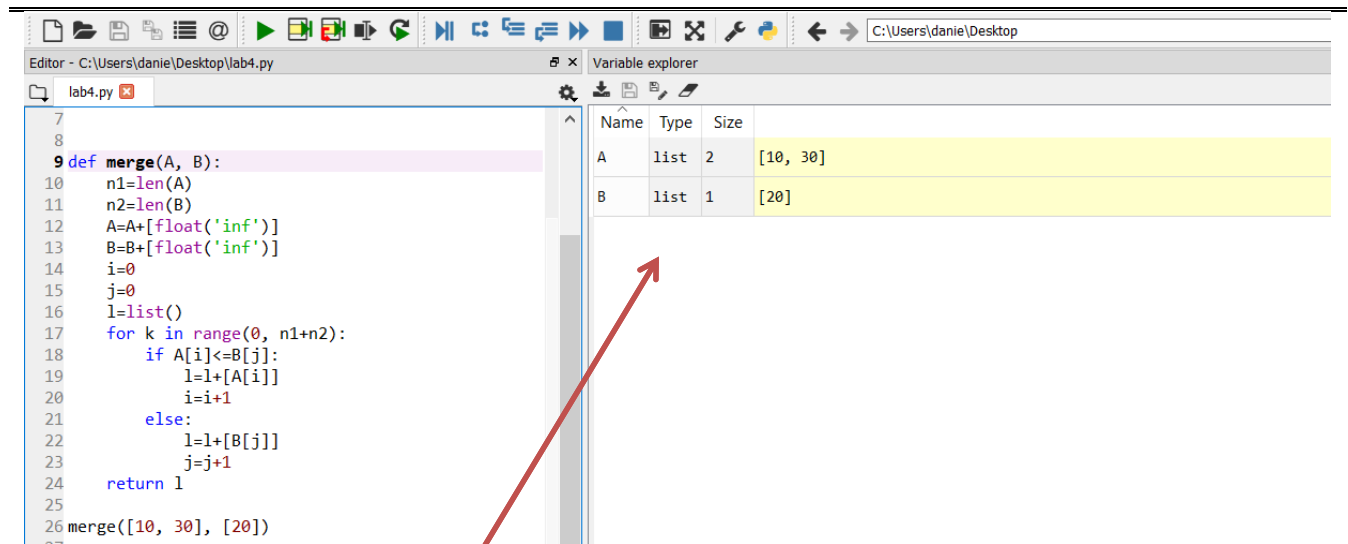
---

***Megesort in Python***

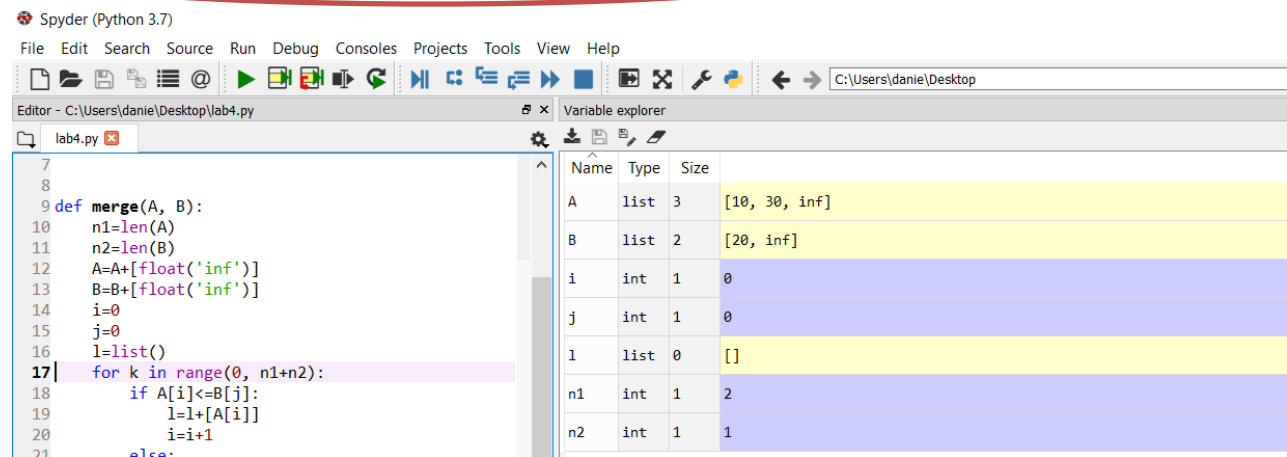
---

In the following section, we see how variables can be watched while running program in debugging mode.

In the following exercise, we see how we can merge two arrays of two and one elements through debugging mode.



Variables are shown here, before the start of the loop execution



*Running merge procedure in debugging mode*

## 1<sup>st</sup> iteration of k:

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\danie\Desktop\lab4.py

```
7
8 def merge(A, B):
9     n1=len(A)
10    n2=len(B)
11    A=A+[float('inf')]
12    B=B+[float('inf')]
13    i=0
14    j=0
15    l=list()
16    for k in range(0, n1+n2):
17        if A[i]<=B[j]:
18            l=l+[A[i]]
19            i=i+1
20        else:
21            l=l+[B[j]]
22            j=j+1
23    return l
```

lab4.py\*

Name	Type	Size	
A	list	3	[10, 30, inf]
B	list	2	[20, inf]
i	int	1	0
j	int	1	0
k	int	1	0
l	list	1	[10]
n1	int	1	2

Variable explorer File explorer Help Profiler

## 2<sup>nd</sup> iteration of k:

Spyder (Python 3.7)

File Edit Search Source Run Debug Consoles Projects Tools View Help

Editor - C:\Users\danie\Desktop\lab4.py

```
8 def merge(A, B):
9     n1=len(A)
10    n2=len(B)
11    A=A+[float('inf')]
12    B=B+[float('inf')]
13    i=0
14    j=0
15    l=list()
16    for k in range(0, n1+n2):
17        if A[i]<=B[j]:
18            l=l+[A[i]]
19            i=i+1
20        else:
21            l=l+[B[j]]
22            j=j+1
23    return l
24
```

lab4.py\*

Name	Type	Size	
A	list	3	[10, 30, inf]
B	list	2	[20, inf]
i	int	1	1
j	int	1	0
k	int	1	1
l	list	2	[10, 20]
n1	int	1	2

Variable explorer File explorer Help Profiler



### 3<sup>rd</sup> iteration of k:

Editor - C:\Users\danie\Desktop\lab4.py

```
6 """
7
8 def merge(A, B):
9     n1=len(A)
10    n2=len(B)
11    A=A+[float('inf')]
12    B=B+[float('inf')]
13    i=0
14    j=0
15    l=list()
16    for k in range(0, n1+n2):
17        if A[i]<=B[j]:
18            l=l+[A[i]]
19            i=i+1
20        else:
```

Variable explorer

Name	Type	Size	Value
A	list	3	[10, 30, inf]
B	list	2	[20, inf]
i	int	1	1
j	int	1	1
k	int	1	2
l	list	3	[10, 20, 30]
n1	int	1	2

### End of for...loop:

Editor - C:\Users\danie\Desktop\lab4.py

```
3 Created on Wed Dec 5 00:13:41 2018
4
5 @author: danie
6 """
7
8 def merge(A, B):
9     n1=len(A)
10    n2=len(B)
11    A=A+[float('inf')]
12    B=B+[float('inf')]
13    i=0
14    j=0
15    l=list()
16    for k in range(0, n1+n2):
17        if A[i]<=B[j]:
18            l=l+[A[i]]
19            i=i+1
20        else:
```

Variable explorer

Name	Type	Size	Value
A	list	3	[10, 30, inf]
B	list	2	[20, inf]
i	int	1	2
j	int	1	1
k	int	1	2
l	list	3	[10, 20, 30]
n1	int	1	2

Variable explorer File explorer Help Profiler

IPython console

### Sorted List returned:

Editor - C:\Users\danie\Desktop\lab4.py

```
10 n2=len(B)
11 A=A+[float('inf')]
12 B=B+[float('inf')]
13 i=0
14 j=0
15 l=list()
16 for k in range(0, n1+n2):
17     if A[i]<=B[j]:
18         l=l+[A[i]]
19         i=i+1
20     else:
21         l=l+[B[j]]
22         j=j+1
23 return l
24
25
26 merge([10, 30], [20])
```

Variable explorer

Name	Type	Size	Value
A	list	3	[10, 30, inf]
B	list	2	[20, inf]
i	int	1	2
j	int	1	1
k	int	1	2
l	list	3	[10, 20, 30]
n1	int	1	2

Variable explorer File explorer Help Profiler

IPython console

Task:

There are going to be recursive calls in the *mergesort* procedure given above. Student is supposed to note the values of different variables during each recursive call and record their observations.

Further, student is supposed to compare the run-time of *mergesort* algorithm, with the sorting algorithms covered in lab session 03.

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

## Laboratory Session No. 05

### Objective:

***Extending the divide-and-conquer approach on sorting and searching problems***

We first start with the analysis and experimental verification of the run-time of linear search algorithm. The linear search algorithm looks for an entry present in the array sequentially. In the second stage, we apply divide and conquer based approach for searching problem and compare the running time of both linear search and binary search analytically as well as empirically.

### The linear search Algorithm

---

```
def linearsearch(x, key):  
    count=0  
    flag=0  
    for i in range(len(x)):  
        count=count+1  
        if x[i]==key:  
            flag=1  
    return flag
```

---

*Code of linear search algorithm in Python*

---

## Analysis of Linear Search (perform for best and worst cases)

Pseudocode		frequency	Time
1	def linearsearch(x, y):		
2	count=0		
3	flag=0		
4	for i in range(len(x)):		
5	count=count+1		
6	if x[i]==y:		
7	flag=1		
8	return flag		
<i>Code of linear search algorithm in Python</i>			

### Best case

### Worst case

## Binary Search

Binary search is done on already sorted array. Program compares the value to be searched from the value present at the mid in the list. If value is lesser than value at mid in the list it looks for the value in the same way in the list on the left of mid. If value is larger than value at mid, it looks in the list on the right of mid. When the value is found it generates an output flag that value is found.

---

```
def bsearch(A, key):
    f_index=0
    l_index=len(A)-1
    flag=0
    while f_index<=l_index and flag==0:
        mid=(f_index+l_index)//2
        if A[mid]==key:
            flag=1
        elif key<A[mid]:
            l_index=mid-1
        else:
            f_index=mid+1
    return flag
```

---

*Code of binary search algorithm in Python*

---

## Analysis of Binary Search

Pseudocode		Frequency	Time
1	def bsearch(A, key):		
2	f_index=0		
3	l_index=len(A)-1		
4	flag=0		
5	while f_index<=l_index and flag==0:		
6	mid=(f_index+l_index)//2		
7	if A[mid]==key:		
8	flag=1		
9	elif key<A[mid]:		
10	l_index=mid-1		
11	else:		
12	f_index=mid+1		
13	return flag		

Best case

Worst case



## Compare the running times of linear search and binary search

### Task:

Student is supposed to implement both searching algorithms and test them for different sizes of inputs. To summaries the observations, growth plots should be made.

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

## Laboratory Session No. 06

### Objective:

*Apply Asymptotic Notations to the Sorting Algorithms.*

### Θ Notation:

The theta notation bounds a function from above and below, so it defines exact asymptotic behavior.

A simple way to get theta notation of an expression is to drop low order terms and ignore leading constants.

For a given function  $g(n)$ , we denote  $\Theta(g(n))$  is following set of functions.

$\Theta(g(n)) = \{f(n): \text{there exist positive constants } c_1, c_2 \text{ and } n_0 \text{ such}$   
that  $0 \leq c_1 * g(n) \leq f(n) \leq c_2 * g(n) \text{ for all } n \geq n_0\}$

The above definition means, if  $f(n)$  is theta of  $g(n)$ , then the value  $f(n)$  is always between  $c_1 * g(n)$  and  $c_2 * g(n)$  for large values of  $n$  ( $n \geq n_0$ ). The definition of theta also requires that  $f(n)$  must be non-negative for values of  $n$  greater than  $n_0$ .

### Θ Notation for Insertion Sort:

In order to apply theta notation for insertion sort we have to bound the time  $T(n)$  graph of insertion sort between two graphs of the same nature that of  $T(n)$  but with different constant terms  $C_1$  and  $C_2$ . The analysis is given by:

S.no	n	T(n)	$T(n)/n^2$	$C_1n^2$	$C_2n^2$
1	1000	0.06	<b><math>C_1=0.000000056</math></b>	0.056	0.064
2	5000	1.57	0.00000006	1.5	1.625
3	10000	6.37	0.00000006	6	6.5
4	15000	14.32	0.00000006	13.5	14.62
5	20000	25.5	<b><math>C_2=0.000000976</math></b>	24.4	26.5

*Tabulation of Upper and lower bound asymptotes*

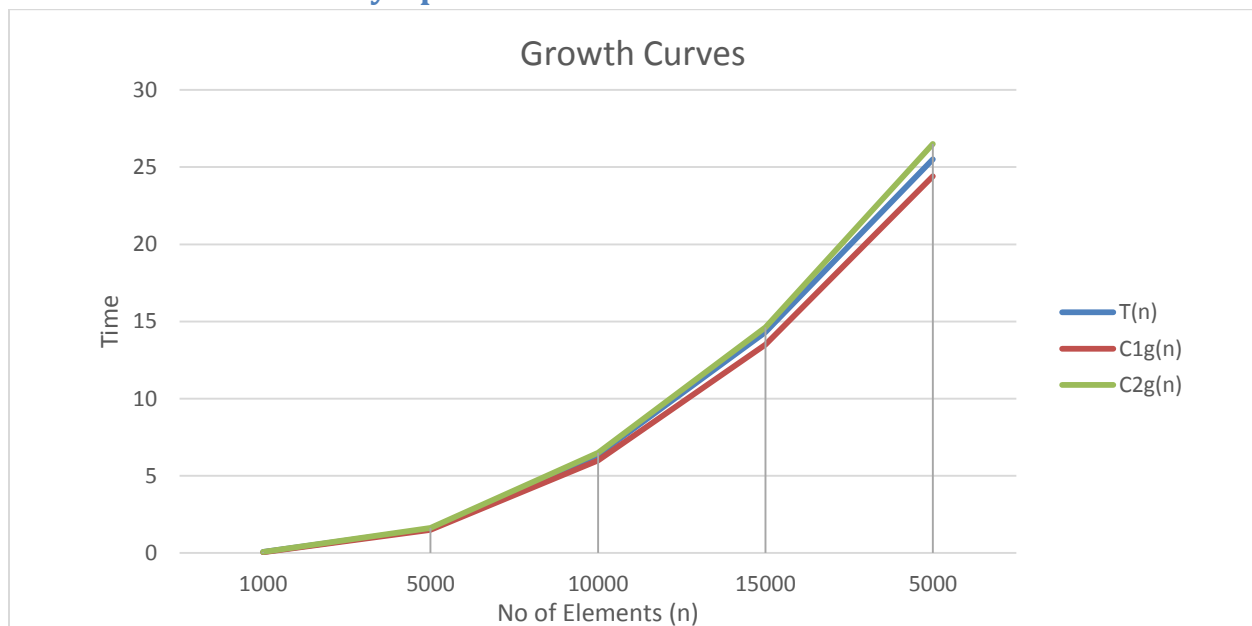
Here  $n$  = no of elements

$T(n)$  = Time Complexity

$C_1n^2$  = Lower Bound

$C_2n^2$  = Upper Bound

### Growth Curves with Asymptotes



## Θ Notation for Merge Sort:

In order to apply theta notation for merge sort we have to bound the time  $T(n)$  graph of merge sort between two graphs of the same nature that of  $T(n)$  but with different constant terms  $C_1$  and  $C_2$ . The analysis is given by:

n	T(n)	$T(n)/n \log n$	$C_1(n \log n)$	$C_2(n \log n)$
1000	0.008	$C_1=7.80E-07$	0.007773	0.008272
5000	0.05	$7.90E-07$	0.048536	0.051608
10000	0.1152	$C_2=8.22E-07$	0.109225	0.118261
15000	0.1761	$8.10E-07$	0.168553	0.181038
20000	0.2477	$8.20E-07$	0.234318	0.25375

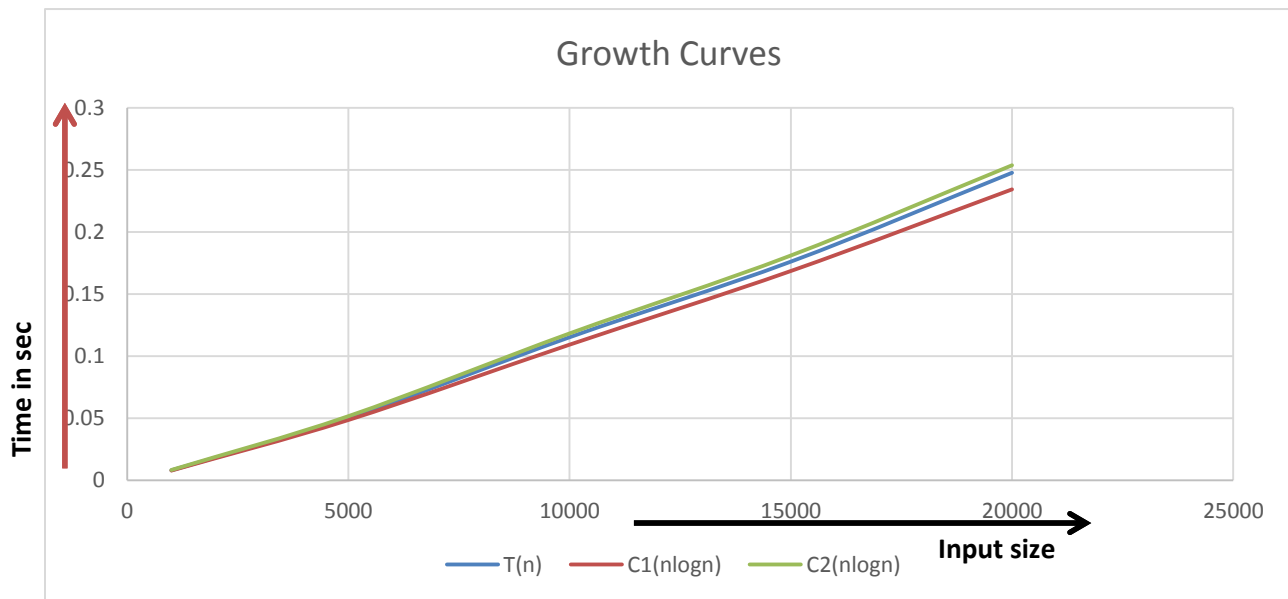
**Tabulation of Upper and lower bound asymptotes**

Here n = no of elements

$T(n)$  = Time Complexity

$C_1 n \log n$  = Lower Bound

$C_2 n \log n$  = Upper Bound



*\*keep in mind that your reading will depend on your computer's speed. The above tables and graphs are just for the verification of concepts*

## Laboratory Session No. 07

### Objective:

***Introduction to object oriented programming (OOP), creating classes and objects***

### SIGNIFICANCE of OOP:

Object-oriented programming is often the most natural approach, once we get the hang of it. OOP languages allow us to break down our software into bite-sized problems that we then can solve — one object at a time. This isn't to say that OOP is the One True Way. However, the advantages of object-oriented programming are many. When you need to solve complex programming challenges and want to add code tools to your skill set, OOP is your friend and has much greater longevity and utility. The concept of data classes allows a programmer to create any new data type that is not already defined in the language itself. The concept of a data class makes it possible to define subclasses of data objects that share some or all of the main class characteristics called inheritance, this property of OOP forces a more thorough data analysis, reduces development time, and ensures more accurate coding.

### CONCEPT OF CLASS AND OBJECT:

A *class* is a template or set of instructions to build a specific type of object. Every object is built from a class. Each class should be designed and programmed to accomplish one, and only one, thing. An object's properties are what it knows and its methods are what it can do.

### 2) CLASSES IN PYTHON:

We can use classes in python in order to save data. We can also access or call the data from different operation when needed.

```
In [4]: class Student():
        '''A student with name, roll number and CGPA'''
        pass

In [7]: t=Student()

In [8]: type (t)
Out[8]: __main__.Student
```

*Class creation in Python*

In the above the making of a general class is shown. Now we are going to use data in the class. The following shows the calling and saving of the data.

---

```
In [9]: t.name='Subhan'

In [10]: t.roll=156

In [11]: t.cgpa=3.0

In [13]: t.name,t.roll,t.cgpa

Out[13]: ('Subhan', 156, 3.0)
```

---

*Assigning attributes*

---

### 3) USE OF `__init__` FUNCTION IN PYTHON:

When a new instance of a [python class](#) is created, it is the `__init__` method which is called and proves to be a very good place where we can modify the object after it has been created. There is no explicit variable declaration in Python. They spring into action on the first assignment. The use of `self` makes it easier to distinguish between instance attributes from local variables. Normal attributes are introduced in the `__init__` method, but some attributes of a class hold for *all* instances in all cases. Following example can be used to understand `__init__` and `self` construct:

---

```
► In [14]: class Student():
            def __init__(Student, nam, cgp, rol):
                Student.name = nam
                Student.cgpa = cgp
                Student.roll = rol
            def name_print(Student):
                '''Print the name of the student.'''
                print("The name assigned is",self.name)
            def cgpa_print(Student):
                '''Print the name of the student.'''
                print("The cgpa of",self.name, 'is', Student.cgpa)
            def all_print(Student):
                '''Print the name of the student.'''
                print(Student.name, 'has roll #',Student.roll,'and cgpa equals to', Student.cgpa)
```

---

*`__init__` function usage*

---



Now after using the constructor the making and calling of the data become easier.

---

```
In [15]: a=Student('Subhan',3.0,156)
         b= Student('Humayun',4.0,150)
         c = Student('Osama',2.5,140)
         d = Student('Bilal',2.8,167)
```

```
In [16]: b.all_print()

         Humayun has roll # 150 and cgpa equals to 4.0
```

```
In [17]: a.all_print()

         Subhan has roll # 156 and cgpa equals to 3.0
```

---

***Creating objects with attributes***

---

## Laboratory Session No. 08

### Objective:

*To implement the following open-ended problem in python*

*Develop a system which can perform following basic banking related tasks*

- a) Customer account could be created with name, NIC, account number and initial balance. All such attributes should be placed in a class*
- b) Balance of any costumer could be updated*
- c) Customer data could be sorted name wise and balance wise(any previously used sorting procedure may be applied)*

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

## Laboratory Session No. 09

### Objective:

*To implement fundamental data structures in Python (using list)*

### Note:

Using *list* in python, implement the following

- a) Stacks(push and pop operations)
- b) Queues(enqueue and dequeuer operations)
- c) A dynamic set 'S' having following functionalities
  - a. Search (S, key)
  - b. Insert (an object)
  - c. Delete (an object)
  - d. Minimum(S)
  - e. Maximum(S)

### a) Stacks (push and pop operations):

Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in First Out) or FILO (First in Last Out). The code of the stack is given below for push and pop operations.

#### Implementation in Python

---

```
class stack():  
  
    def __init__(self):  
        self.stack=list()  
  
    def push(self,data):  
        self.stack.insert(0,data)  
  
    def pop(self):  
        print(self.stack[0])  
        self.stack.remove(self.stack[0])
```

---

Python code for class *STACK*

---

### b) Queues (enqueue and dequeuer operations):

A Queue is a linear structure which follows a particular order in which the operations are performed. The order is First-In-First-Out (FIFO). A good example of a queue is any queue of consumers for a resource where the consumer that came first is served first. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added. The code of queue is given under

#### Implementation in Python

---

```
class queue():  
  
    def __init__(self):  
  
        self.queue=list()  
  
    def enqueue(self,data):  
  
        self.queue.insert(0,data)  
  
    def dequeue(self):  
  
        n=len(self.queue)  
  
        print(self.queue[n-1])  
  
        self.queue.remove(self.queue[n-1])
```

---

Python code for *QUEUE*

---

### c) A dynamic Set

Dynamic set may refer to: A set (abstract data type) that supports insertion and/or deletion of elements. This data structure is frequently used in database access. The code for various performing operation of the dynamic set is provided below

#### Implementation in Python

---

```
class ds():
    l=list()
    def add(self,data):
        ds.l.append(data)
        print(ds.l)
    def delete(self,data):
        ds.l.remove(data)
        print(ds.l)
    def search(self,key):
        flag=0
        for i in range(len(ds.l)):
            if ds.l[i]==key:
                flag=1
        return flag
    def min(self):
        for j in range(1,len(ds.l)):
            key=ds.l[j]
            i=j-1
            while i>-1 and ds.l[i]>key:
                ds.l[i+1]=ds.l[i]
                i=i-1
            ds.l[i+1]=key
        print(ds.l[0])
    def max(self):
        n=len(ds.l)
        for j in range(1,len(ds.l)):
            key=ds.l[j]
            i=j-1
```

---



---

```
while i>-1 and ds.l[i]>key:
    ds.l[i+1]=ds.l[i]
    i=i-1
    ds.l[i+1]=key
print(ds.l[n-1])
```

---

Python code for *DYNAMIC SET*

---

Note:

Student is now supposed to create objects and perform relevant tasks using those objects for the above classes.

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

## Laboratory Session No. 10

### Objective:

*Accomplish the following open ended tasks*

*Using Node class, develop*

1. Stack
2. Queues
3. Singly connected linked-list with following features
  - a. Add nodes
  - b. Traverse all nodes starting from top node
  - c. Search any key value in all nodes
  - d. Insert node between any two nodes

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

## Laboratory Session No. 11

### Objective:

*Accomplish the open-ended task:*

*Using Node class, develop singly connected linked-list*

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60



Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	

**NED University of Engineering & Technology**  
**Department of Electrical Engineering**



Course Code: **EE-264**

Course Title: **Data Structures and Algorithms**

Laboratory Session No.: \_\_\_\_\_

Date: \_\_\_\_\_

<b>Psychomotor Domain Assessment Rubric for Laboratory (Level P3)</b>					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Initialisation and Configuration:</b> Set up and <u>recognise</u> software initialisation and configuration steps 10%	Completely unable to recognise initialisation and configuration 0	Able to recognise initialisation but could not configure 10	Able to recognise initialisation but configuration is erroneous 20	Able to recognise initialisation and configuration with minimal errors 30	Able to recognise initialisation and configuration with complete success 40
<b>Input/Output Variable Recognition, Definition and Initialisation:</b> <u>Recognise</u> and <u>perceive</u> correct input/output variables along with data types for testing a specific algorithm/data structure 15%	Incorrect perception for both Input/Output variables and data types 0	Correctly perceives the required Input/Output variables and data types but fails to initialise variables altogether 15	Correctly perceives the required Input/Output variables and data types and only initialises them partially 30	Correctly perceives the required Input/Output variables and data types and initialises them completely but with errors 45	Correctly perceives the required Input/Output variables and data types and initialises them with complete success 60
<b>Procedural Programming of given Algorithm:</b> <u>Practice</u> procedural programming techniques including recursion, in order to code specific algorithms from their pseudo code 15%	Little to no understanding of procedural programming techniques 0	Slight ability to use procedural programming techniques for coding given algorithm 15	Mostly correct recognition and application of procedural programming techniques but makes crucial errors for the given algorithm 30	Correctly recognises and uses procedural programming techniques with no errors but unable to run algorithm successfully 45	Correctly recognises and uses procedural programming techniques with no errors and runs algorithm successfully 60
<b>Object Oriented Programming for given Algorithm and Data Structure Implementation:</b> <u>Imitate</u> and <u>practice</u> given OOP instructions for making specific data structure/algorithm 15%	Incorrect selection and use of programming constructs and instructions 0	Correct selection of programming constructs and instructions but their use is incorrect 15	Correct selection and use of programming constructs and instructions with many syntax/semantic errors 30	Correct selection and use of programming constructs and instructions with little to no syntax/semantic errors 45	Correct selection and use of programming constructs and instructions with no syntax/semantic errors 60

Psychomotor Domain Assessment Rubric for Laboratory (Level P3)					
Skill(s) to be assessed	Extent of Achievement				
	0	1	2	3	4
<b>Software Menu Identification and Usage:</b> Ability to <u>operate</u> software environment <u>under supervision</u> , using menus, shortcuts, instructions etc. 15%	Unable to understand and use software menu 0	Little ability and understanding of software menu operation, makes many mistake 15	Moderate ability and understanding of software menu operation, makes lesser mistakes 30	Reasonable understanding of software menu operation, makes no major mistakes 45	Demonstrates command over software menu usage with frequent use of advance menu options 60
<b>Detecting and Removing Errors/Exceptions:</b> <u>Detect</u> Errors/Exceptions and <u>manipulate</u> , under supervision, to rectify the Code 10%	Unable to check and detect error messages in software 0	Able to find error messages in software but no understanding of detecting those errors and their types 10	Able to find error messages in software as well as understanding of detecting some of those errors and their types 20	Able to find error messages in software as well as understanding of detecting all of those errors and their types 30	Able to find error messages in software along with the understanding to detect and rectify them 40
<b>Debugging and Troubleshooting:</b> <u>Recognise and Practice</u> Debugging and Troubleshooting steps through line-by-line code execution 10%	Unable to recognise and use debugging options in software 0	Little ability to recognise and use debugging and troubleshooting options in software 10	Ability to recognise and use debugging and troubleshooting options with little ability to rectify code 20	Ability to recognise and use debugging and troubleshooting options with ability to rectify and step-through code 30	Ability to recognise, describe, and use debugging and troubleshooting with ability to rectify and step-through code 40
<b>Graphical visualisation and comparison of time complexity of algorithms:</b> <u>Manipulate</u> given Code/Instructions under supervision, in order to produce graphs for comparing time complexity of algorithms 10%	Unable to understand and utilise visualisation or plotting instructions 0	Ability to understand and utilise visualisation and plotting instructions with errors 10	Ability to understand and utilise visualisation and plotting instructions successfully but unable to draw results from them 20	Ability to understand and utilise visualisation and plotting instructions successfully, partially able to draw results from them 30	Ability to understand and utilise visualisation and plotting instructions successfully, also able to draw complete results from them 40

Total Points (out of 400)	
Weighted CLO (Psychomotor Score)	(Points/4)
Remarks	
Instructor's Signature with Date	