Course Code	**:**	**BCSL-045**
Course Title	**:**	**Introduction to Algorithm design Lab**
Assignment Number	**:**	**BCA(IV)/L-045/Assignment/2026-27**
Maximum Marks	**:**	**50**
Weightage	**:**	**25%**
Last date of Submission	**:**	**31st October, 2026 (For July 2026 Session)**
	:	**30th April, 2027 (For January 2027 Session)**

Note: Answer all the questions which carry 40 marks. All questions are of equal marks. The rest 10 marks are for viva voce. You are required to write programs in C-language for all the problems, execute and show the results. You may use illustrations and diagrams to enhance the explanations. Please go through the guidelines regarding assignments given in the Programme Guide for the format of presentation. Make suitable assumption if necessary.

Q1: Implement *Insertion Sort* and *Selection Sort in C* for randomly generated integer arrays of sizes *10* and *15*. Perform the following:

- Count the number of comparisons and swaps.
- Measure the execution time for both algorithms.
- Present the observations in a tabular form.
- Compare their best-case and worst-case time complexities. **(8 Marks)**

Q2: Write a C program to solve the *Tower of Hanoi* problem using recursion. Your program should:

- Accept the number of disks from the user.
- Display all the moves required.
- Calculate the total number of moves.
- Verify that the number of moves satisfies the formula 2^n-1.
- Discuss the time complexity of the recursive solution. **(8 Marks)**

Q3: Implement Dijkstra's algorithm for a graph with 5 nodes using adjacency list representation. Display the shortest path tree and explain how the greedy choice is applied at each step. **(8 Marks)**

Q4: Write a C program to perform and visualize DFS and BFS traversal of a user-defined undirected graph. Record and report the order of traversal and number of operations. **(8 Marks)**

Q5: Develop a C program to implement *Linear Search* and *Binary Search.*

For an array of user-defined integers:

- Search for a given element using both algorithms.
- Display the number of comparisons made.
- Compare their performance for sorted and unsorted data.
- Explain why Binary Search is more efficient under appropriate conditions. **(8 Marks)**

Q1: Implement Insertion Sort and Selection Sort in C for randomly generated integer arrays of sizes 10 and 15. Perform the following:

- **Count the number of comparisons and swaps.**
- **Measure the execution time for both algorithms.**
- **Present the observations in a tabular form.**
- **Compare their best-case and worst-case time complexities. (8 Marks)**

Answer

Both algorithms use identical arrays at each input size for a fair comparison. The program generates integer values with a fixed pseudorandom seed (202627), then copies each array before sorting. Insertion Sort is implemented by adjacent exchanges, so its swap count is meaningful. A comparison means comparing two array elements; a swap means exchanging two different positions. Timing uses clock() CPU time averaged over 100,000 complete sorts because one tiny sort is too fast for a stable single-run reading.

C program

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define REPEATS 100000

typedef struct { long long comparisons, swaps; } Stats;

void insertionSort(int a[], int n, Stats *s) {
    s->comparisons = s->swaps = 0;
    for (int i = 1; i < n; i++) {
        int j = i;
        while (j > 0) {
            s->comparisons++;
            if (a[j-1] <= a[j]) break;
            int temp = a[j-1]; a[j-1] = a[j]; a[j] = temp;
            s->swaps++; j--;
        }
    }
}

void selectionSort(int a[], int n, Stats *s) {
    s->comparisons = s->swaps = 0;
    for (int i = 0; i < n-1; i++) {
        int min = i;
        for (int j = i+1; j < n; j++) {
            s->comparisons++;
            if (a[j] < a[min]) min = j;
        }
        if (min != i) {
```