Lab/Project Assignment Report

Only for course Teacher						
		Needs Improvement	Developing	Sufficient	Above Average	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	5
Content Quality	2					
Clarity	1					
Spelling & Grammar	1					
Organization and Formatting	1					
					Total obtained mark	
Comments						

Semester: Spring 2023

Student Name: Md. Mubtasim Fuad Khan

Student ID:221-35-883

Batch: 37 **Section: B**

Course Code: SE214 **Course Name: Algorithm Design and Analysis**

Course Teacher Name: Aditi Dhali

Designation: Lecturer, Department of Software Engineering

Submission Date: 06/06/2023

Problem 01: Linear Search

array linear search and delete.c | Linear Search.c

Lab Assignment > Codes > Linear Search.c > main()

```c
// linear search an elements and delete that element
#include<stdio.h>
void main()
{
int a[100],i,newindex,n,search;
printf("Enter array Size: ");
scanf("%d",&n);
printf("\nEnter %d array elements \n",n);
for(i=0;i<n;i++)
    scanf("%d",&a[i]);
printf("Array elements are: \n");
for(i=0;i<n;i++)
    printf("%d\t",a[i]);

    printf("\nEnter a number to search: ");
  scanf("%d", &search);

  for (i = 0; i < n; i++)
  {
    if (a[i] == search)    /* If required element is found */
    {
      printf("%d is present at Position %d.\n", search, i+1);
      printf("%d is present at Index no %d.\n", search, i);
      break;
    }
  }

}
```

Output:

```
Enter array Size: 5

Enter 5 array elements
21 12 23 44 94
Array elements are:
21      12      23      44      94
Enter a number to search: 44
44 is present at Position 4.
44 is present at Index no 3.
```

Problem 02: Binary search

Linear Search.c | Binary search.c | Binary Search.c

Lab Assignment > Codes > Binary search.c > main()

```c
#include <stdio.h>
int main() {
    int a[]={1,2,4,5,67,89,100};
    int item=89;

    // Runtime Complexity: O(logN)
    // Space Complexity: O(10)

    int left,right,middle;
    left=0;
    right=6;

    while(left <= right){
        middle=(left+right)/2;
        if(a[middle] == item){
            printf("Item found at index: %d\n",middle);
            return 0;
        }else if(a[middle < item]){
            left = middle + 1;
        }else{
            right = middle - 1;
        }
    }
    printf("Item not found!!!\n");
    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Binary search.exe'
Item found at index: 5
```

Problem 03: Insertion Sort

Insertion Sort.c

Lab Assignment > Codes > Insertion Sort.c > main()

```c
// Insertion sort in C
#include <stdio.h>
int main()
{
    int array[100];
    int i, j, key, size;

    printf("Enter the number of elements in the array: ");
    scanf("%d", &size);

    printf("Enter the array elements: ");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &array[i]);
    }

    for (i = 1; i < size; i++)
    {
        key = array[i];
        j = i - 1;

        while (key < array[j] && j >= 0)
        {
            array[j + 1] = array[j];
            j--;
        }
        array[j + 1] = key;
    }

    printf("Sorted array in ascending order:\n");
    for (i = 0; i < size; i++)
    {
        printf("%d ", array[i]);
    }
    printf("\n");
}
```

Output:

```
Enter the number of elements in the array: 5
Enter the array elements: 13
67
44
98
23
Sorted array in ascending order:
13 23 44 67 98
```

Problem 04: Selection Sort

Insertion Sort.c | Selection Sort.c

Lab Assignment > Codes > Selection Sort.c > main()

```c
#include <stdio.h>
int main()
{
    int a[100], n, i, j, min, swap;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    printf("Enter %d Numbers\n", n);
    for (i = 0; i < n; i++)
        scanf("%d", &a[i]);
    for(i = 0; i < n - 1; i++)
    {
        min=i;
        for(j = i + 1; j < n; j++)
        {
            if(a[min] > a[j])
                min=j;
        }
        if(min != i)
        {
            swap=a[i];
            a[i]=a[min];
            a[min]=swap;
        }
    }
    printf("Sorted Array:  ");
    for(i = 0; i < n; i++)
        printf("%d\t", a[i]);
    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Selection Sort.exe'
Enter number of elements: 6
Enter 6 Numbers
12
76
34
84
9
44
Sorted Array:  9        12      34      44      76      84
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output>
```

Problem 05: Merge Sort

Insertion Sort.c | Selection Sort.c | Merge Sort.c

Lab Assignment > Codes > Merge Sort.c > ...

```c
#include <stdio.h>
void merge(int arr[], int l, int m, int r) {
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;

    // Create temporary arrays
    int L[n1], R[n2];

    // Copy data to temporary arrays L[] and R[]
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    // Merge the temporary arrays back into arr[l..r]
    i = 0; // Initial index of first subarray
    j = 0; // Initial index of second subarray
    k = l; // Initial index of merged subarray

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
```

```c

    // Copy the remaining elements of L[], if there are any
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Copy the remaining elements of R[], if there are any
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

// Main function that sorts arr[l..r] using merge()
void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        // Same as (l+r)/2, but avoids overflow for large l and h
        int m = l + (r - l) / 2;

        // Sort first and second halves
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);

        merge(arr, l, m, r);
    }
}
```

```c

// Utility function to print an array
void printArray(int A[], int size) {
    int i;
    for (i = 0; i < size; i++)
        printf("%d ", A[i]);
    printf("\n");
}

// Test the merge sort function
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);

    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Merge Sort.exe'
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output>
```

Problem 06: Bubble Sort

Bubble Sort.c

Lab Assignment > Codes > Bubble Sort.c > main()

```c
#include <stdio.h>
int main() {
    int arr[]={10, 30, 20, 50, 70, 90};
    int i,j,temp,size=6;

    for(i=0; i<size-1; i++){

        for(j=0; j<size-1-i; j++){
            if(arr[j] > arr[j+1]){
                //swap two numbers
                temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;

            }
        }

        }

    printf("After sorting:\n");
    for(i=0; i < size; i++){
        printf("%d ",arr[i]);
    }
    printf("\n");
    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Bubble Sort.exe'
After sorting:
10 20 30 50 70 90
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output>
```

Problem 07: Quick Sort

Bubble Sort.c | Quick Sort.c

Lab Assignment > Codes > Quick Sort.c > ...

```c
#include <stdio.h>

// Swap two elements
void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

// Partition function to find the correct position of the pivot
int partition(int arr[], int low, int high) {
    int pivot = arr[high];  // Choosing the last element as the pivot
    int i = (low - 1);  // Index of smaller element

    for (int j = low; j <= high - 1; j++) {
        // If current element is smaller than or equal to the pivot
        if (arr[j] <= pivot) {
            i++;  // Increment index of smaller element
            swap(&arr[i], &arr[j]);
        }
    }
    swap(&arr[i + 1], &arr[high]);
    return (i + 1);
}

// Quicksort function
void quickSort(int arr[], int low, int high) {
    if (low < high) {
        // Partition the array into two sub-arrays
        int pi = partition(arr, low, high);

        // Recursive call to quicksort the left and right sub-arrays
        quickSort(arr, low, pi - 1);
        quickSort(arr, pi + 1, high);
    }
}
```

```c

// Utility function to print an array
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

// Test the quicksort function
int main() {
    int arr[] = {10, 7, 8, 9, 1, 5};
    int arr_size = sizeof(arr) / sizeof(arr[0]);

    printf("Given array is \n");
    printArray(arr, arr_size);

    quickSort(arr, 0, arr_size - 1);

    printf("\nSorted array is \n");
    printArray(arr, arr_size);

    return 0;
}

```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Quick Sort.exe'
Given array is
10 7 8 9 1 5

Sorted array is
1 5 7 8 9 10
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output>
```

Problem 08: Coin change

Bubble Sort.c | Quick Sort.c | Coin change.c

Lab Assignment > Codes > Coin change.c > ...

```c
#include <stdio.h>
#include <limits.h>

// Function to find the minimum number of coins required to make a given amount of change
int coinChange(int coins[], int numCoins, int amount) {
    // Create a table to store the minimum number of coins required for each amount
    int dp[amount + 1];
    int i, j;

    // Initialize the table with a maximum value
    for (i = 0; i <= amount; i++)
        dp[i] = INT_MAX;

    // Base case: 0 coins are required to make change for 0 amount
    dp[0] = 0;

    // Compute the minimum number of coins required for each amount
    for (i = 1; i <= amount; i++) {
        // Try each coin denomination
        for (j = 0; j < numCoins; j++) {
            if (coins[j] <= i) {
                int subproblem = dp[i - coins[j]];
                if (subproblem != INT_MAX && subproblem + 1 < dp[i])
                    dp[i] = subproblem + 1;
            }
        }
    }

    // If the final value remains INT_MAX, it means it's not possible to make the given amount
    if (dp[amount] == INT_MAX)
        return -1;

    return dp[amount];
}
```

```c
// Test the coin change function
int main() {
    int coins[] = {1, 2, 5};
    int numCoins = sizeof(coins) / sizeof(coins[0]);
    int amount = 11;

    int minCoins = coinChange(coins, numCoins, amount);

    if (minCoins != -1)
        printf("Minimum number of coins required to make change for amount %d: %d\n", amount, minCoins);
    else
        printf("It is not possible to make change for amount %d.\n", amount);

    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Coin change.exe'

Minimum number of coins required to make change for amount 11: 3
```

Problem 09: Knap Sack

Bubble Sort.c | Quick Sort.c | Coin change.c | Knap Sack.c | BFS.c

Lab Assignment > Codes > Knap Sack.c > main()

```c
#include<stdio.h>

// Function to find the maximum of two integers
int max(int a, int b) {
    return (a > b) ? a : b;
}

// Function to solve the Knapsack problem
int knapSack(int W, int wt[], int val[], int n) {
    int i, w;
    int K[n+1][W+1];

    // Build the K[][] table in bottom-up manner
    for (i = 0; i <= n; i++) {
        for (w = 0; w <= W; w++) {
            if (i == 0 || w == 0)
                K[i][w] = 0;
            else if (wt[i-1] <= w)
                K[i][w] = max(val[i-1] + K[i-1][w-wt[i-1]], K[i-1][w]);
            else
                K[i][w] = K[i-1][w];
        }
    }

    // Return the maximum value that can be put in the knapsack of capacity W
    return K[n][W];
}
```

```c

// Example usage
int main() {
    int val[] = {60, 100, 120};
    int wt[] = {10, 20, 30};
    int W = 50;
    int n = sizeof(val)/sizeof(val[0]);

    int maxValue = knapSack(W, wt, val, n);
    printf("\nMaximum value that can be obtained: %d\n\n", maxValue);

    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'Knap Sack.exe'

Maximum value that can be obtained: 220
```

Problem 10: BFS

Bubble Sort.c | Quick Sort.c | Coin change.c | Knap Sack.c | BFS.c

Lab Assignment > Codes > BFS.c > ...

```c
#include <stdio.h>
#include <stdlib.h>

int a[20][20], q[20] = {0}, visited[20] = {0}, n, i, j, f = 0, r = -1;

void bfs(int v)
{
    for (i = 0; i < n; i++)
    {
        if (a[v][i] == 1 && visited[i] == 0)
        {
            q[++r] = i;
            visited[i] = 1;
        }
    }

    if (f <= r)
    {
        bfs(q[f++]);
    }
}
```

```c
int main()
{
    int v;

    printf("\n Enter The number of vertices: ");
    scanf("%d", &n);

    printf("\n ENTER GRAPH DATA IN MATRIX FORM:\n");
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }

    printf("\n Enter the starting vertex: ");
    scanf("%d", &v);
```

```
41
42      bfs(v);
43
44      int allReachable = 1;
45
46      for (i = 0; i < n; i++)
47      {
48          if (visited[i])
49          {
50              printf("%d\t", i);
51          }
52          else
53          {
54              allReachable = 0;
55              break;
56          }
57      }
58
59      if (!allReachable)
60      {
61          printf("\n BFS is not possible. Not all nodes are reachable.\n");
62      }
63
64      return 0;
65  }
66
```

Output:

```
 Enter The number of vertices: 4

 ENTER GRAPH DATA IN MATRIX FORM:
0 1 0 1
1 0 1 1
0 1 0 1
1 1 1 0

 Enter the starting vertex: 1
0       1       2       3
```

Problem 11: DFS

BFS.c DFS.c

Lab Assignment > Codes > DFS.c > ...

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX_VERTICES 10

int graph[MAX_VERTICES][MAX_VERTICES];
int visited[MAX_VERTICES];
int total;

void DFS(int vertex);

int main()
{
    int i, j;

    printf("Enter the total number of vertices in the graph: ");
    scanf("%d", &total);

    // Adjacency matrix input
    printf("Enter the adjacency matrix:\n");
    for (i = 0; i < total; i++) {
        for (j = 0; j < total; j++) {
            scanf("%d", &graph[i][j]);
        }
    }

    for (i = 0; i < total; i++) {
        visited[i] = 0;
    }

    printf("\nDFS traversal:\n");
    DFS(0);

    return 0;
}

void DFS(int vertex)
{
    int j;
    printf("%d\t", vertex);
    visited[vertex] = 1;

    for (j = 0; j < total; j++) {
        if (!visited[j] && graph[vertex][j] == 1) {
            DFS(j);
        }
    }
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'DFS.exe'
Enter the total number of vertices in the graph: 4
Enter the adjacency matrix:
0 1 0 1
1 0 1 0
0 1 0 1
1 0 1 0

DFS traversal:
0       1       2       3
```

Problem 12: LCS

BFS.c | DFS.c | Prim's Algorithm.c | LCS.c

Lab Assignment > LCS.c > lcs(char *, char *, int, int)

```c
#include <stdio.h>
#include <string.h>

int max(int a, int b) {
    return (a > b) ? a : b;
}

void lcs(char* X, char* Y, int m, int n) {
    int L[m + 1][n + 1];
    int i, j;

    // Build the L[m+1][n+1] table in bottom-up manner
    for (i = 0; i <= m; i++) {
        for (j = 0; j <= n; j++) {
            if (i == 0 || j == 0)
                L[i][j] = 0;
            else if (X[i - 1] == Y[j - 1])
                L[i][j] = L[i - 1][j - 1] + 1;
            else
                L[i][j] = max(L[i - 1][j], L[i][j - 1]);
        }
    }
```

```c
    // Length of LCS is stored in L[m][n]
    int length = L[m][n];

    // Allocate memory to store the LCS
    char lcs[length + 1];
    lcs[length] = '\0';  // Set the null character at the end

    // Traverse the L[m+1][n+1] table to find the LCS
    i = m;
    j = n;
    while (i > 0 && j > 0) {
        // If current characters in X and Y are equal, it is part of the LCS
        if (X[i - 1] == Y[j - 1]) {
            lcs[length - 1] = X[i - 1];
            i--;
            j--;
            length--;
        }
        // If not equal, then find the larger of two and go in the direction of larger value
        else if (L[i - 1][j] > L[i][j - 1])
            i--;
        else
            j--;
    }

    // Print the LCS
    printf("\nLCS: %s\n\n", lcs);
}
```

```c
int main() {
    char X[] = "AGGTAB";
    char Y[] = "GXTXAYB";

    int m = strlen(X);
    int n = strlen(Y);

    lcs(X, Y, m, n);

    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\output> & .\'LCS.exe'

LCS: GTAB
```

Problem 13: LIS

BFS.c | DFS.c | Prim's Algorithm.c | LCS.c | LIS.c

Lab Assignment > Codes > LIS.c > ...

```c
#include <stdio.h>

int lis(int arr[], int n) {
    int lis[n];
    int i, j, max = 0;

    // Initialize LIS values for all indexes
    for (i = 0; i < n; i++)
        lis[i] = 1;

    // Compute optimized LIS values in bottom-up manner
    for (i = 1; i < n; i++) {
        for (j = 0; j < i; j++) {
            if (arr[i] > arr[j] && lis[i] < lis[j] + 1)
                lis[i] = lis[j] + 1;
        }
    }

    // Pick the maximum of all LIS values
    for (i = 0; i < n; i++) {
        if (lis[i] > max)
            max = lis[i];
    }

    return max;
}

```

```c
int main() {
    int arr[] = {10, 22, 9, 33, 21, 50, 41, 60};
    int n = sizeof(arr) / sizeof(arr[0]);

    int length = lis(arr, n);

    printf("\nLength of LIS: %d\n\n", length);

    return 0;
}
```

Output:

```
PS F:\SWE\2nd Year Spring 2023\Algorithm\Lab Assignment\Codes\output> & .\'LIS.exe'

Length of LIS: 5
```