SCHOOL OF COMPUTING

DESIGN AND ANALYSIS OF ALGORITHMS

LAB WORKBOOK

WEEK - 2

NAME : SANTHOSH A

ROLL NUMBER : CH.SC.U4CSE24142

CLASS : CSE-B

Question 1: Write a C program that reads N integers from the user and sorts them in ascending order using Bubble Sort.

CODE:

```c
//Write a C program that reads N integers from the user and sorts them in ascending order using Bubble Sort
// CH.SC.U4CSE24142 - SANTHOSH A
#include <stdio.h>

void bubbleSort(int arr[], int n) {
    int i, j, temp;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int num, i;
    printf("Enter the number of integers: ");
    scanf("%d", &num);
    int arr[num];
    printf("Enter %d integers:\n", num);
    for (i = 0; i < num; i++) {
        scanf("%d", &arr[i]);
    }
    bubbleSort(arr, num);
    printf("Sorted integers in ascending order:\n");
    for (i = 0; i < num; i++) {
        printf("%d ", arr[i]);
```

```
    }
    return 0;
}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Bubble_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter the number of integers: 5
Enter 5 integers:
42 20 10 12 11
Sorted integers in ascending order:
10 11 12 20 42
```

Space Complexity: O(1)

Time Complexity: $O(N^2)$

Justification:

- Uses two nested loops to repeatedly compare adjacent elements. Hence the Space Complexity is O(1).
- Uses two nested loops to repeatedly compare adjacent elements. Hence its Time Complexity is $O(N^2)$.

Question 2: Create a C program that implements Selection Sort to sort an array of integers.

CODE:

```
//Create a C program that implements Selection Sort to sort an array of integers.
// CH.SC.U4CSE24142 - SANTHOSH A
#include <stdio.h>
#include <stdlib.h>
void selectionSort(int arr[], int n) {
    int i, j, min_idx, temp;
    for (i = 0; i < n - 1; i++) {
        min_idx = i;
        for (j = i + 1; j < n; j++) {
            if (arr[j] < arr[min_idx]) {
                min_idx = j;
            }
        }
```

```c
        temp = arr[min_idx];
        arr[min_idx] = arr[i];
        arr[i] = temp;
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int n;
    printf("Enter the number of integers: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    selectionSort(arr, n);
    printf("Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Selection_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter the number of integers: 6
Enter 6 integers:
42 41 49 45 57 11
Sorted array:
11 41 42 45 49 57
```

Space Complexity: O(1)

Time Complexity: $O(N^2)$

Justification:

- No additional memory is used apart from a temporary variable. Hence the Space Complexity is O(1).
- The outer loop runs n-1 times and the inner loop scans the remaining elements to find the minimum value, regardless of input order. Hence the Time Complexity is $O(N^2)$.

Question 3: Write a C program that performs Insertion Sort on an array of integers.

CODE:

```c
//Write a C program that performs Insertion Sort on an array of integers.
// CH.SC.U4CSE24142 - SANTHOSH A
#include <stdio.h>
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int n, i;
    printf("Enter the number of integers: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
```

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

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Insertion_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter the number of integers: 5
Enter 5 integers:
38 14 7 25 3
Sorted array:
3 7 14 25 38
```

Space Complexity: O(1)

Time Complexity: $O(N^2)$

Justification:

- Uses only a single variable to store the current element.Hence the Space Complexity is O(1).
- The while loop may compare the current element with all previous elements in the worst case. Hence the Time Complexity is $O(N^2)$.

Question 4: Write a C program to sort a given set of integers using Heap Sort based on Max Heap.

CODE:

```
//Write a C program to sort a given set of integers using Heap Sort based
on Max Heap
//CH.SC.U4CSE24142 - SANTHOSH A
#include <stdio.h>
void heapify(int arr[], int n, int i) {
    int largest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] > arr[largest])
```

```
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;
    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }
    for (int i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    heapSort(arr, n);
    printf("Sorted array:\n");
```

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

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Max_Heap_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter number of elements: 7
Enter 7 integers:
20 5 17 9 30 42 3
Sorted array:
3 5 9 17 20 30 42
```

Space Complexity: O(1)

Time Complexity: O(N log N)

Justification:

- The sorting is done in-place using the same array with only a few temporary variables. Hence the Space Complexity is O(1).
- Building the max heap takes O(n) time and heapify is called n times, each taking O(log n). Hence the Time Complexity is O(N log N).

Question 5: Write a C program to sort a given set of integers using Heap Sort based on Min Heap.

CODE:

```
//Write a C program to sort a given set of integers using Heap Sort based
on Min Heap.
// CH.SC.U4CSE24142 - SANTHOSH A
#include <stdio.h>
void heapify(int arr[], int n, int i) {
    int smallest = i;
    int left = 2 * i + 1;
    int right = 2 * i + 2;
    if (left < n && arr[left] < arr[smallest])
        smallest = left;
    if (right < n && arr[right] < arr[smallest])
```

```c
        smallest = right;
    if (smallest != i) {
        int temp = arr[i];
        arr[i] = arr[smallest];
        arr[smallest] = temp;
        heapify(arr, n, smallest);
    }
}
void heapSort(int arr[], int n) {
    for (int i = n / 2 - 1; i >= 0; i--) {
        heapify(arr, n, i);
    }
    for (int i = n - 1; i > 0; i--) {
        int temp = arr[0];
        arr[0] = arr[i];
        arr[i] = temp;
        heapify(arr, i, 0);
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int n;
    printf("Enter number of elements: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d integers:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    heapSort(arr, n);
    printf("Sorted array (Descending Order):\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
```

```
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Min_Heap_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter number of elements: 6
Enter 6 integers:
40 15 30 10 25 5
Sorted array (Descending Order):
40 30 25 15 10 5
```

Space Complexity: O(1)

Time Complexity: O(N log N)

Justification:

- The sorting is done in-place using the same array with only a few temporary variables. Hence the Space Complexity is O(1).
- Building the min heap takes O(n) time and heapify is called n times, each taking O(log n). Hence the Time Complexity is O(N log N).

Question 6: Write a C program that implements Bucket Sort to sort an array of floating-point numbers

CODE:

```
//Write a C program that implements Bucket Sort to sort an array of
floating-point numbers
#include <stdio.h>
#include <stdlib.h>
void insertionSort(float arr[], int n) {
    for (int i = 1; i < n; i++) {
        float key = arr[i];
        int j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
```

```
            j--;
        }
        arr[j + 1] = key;
    }
}
void bucketSort(float arr[], int n) {
    float buckets[n][n];
    int bucketCount[n];
    for (int i = 0; i < n; i++) {
        bucketCount[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        int bucketIndex = arr[i] * n;
        buckets[bucketIndex][bucketCount[bucketIndex]] = arr[i];
        bucketCount[bucketIndex]++;
    }
    for (int i = 0; i < n; i++) {
        if (bucketCount[i] > 0) {
            insertionSort(buckets[i], bucketCount[i]);
        }
    }
    int index = 0;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < bucketCount[i]; j++) {
            arr[index++] = buckets[i][j];
        }
    }
}
int main() {
    int n;
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    printf("Enter number of floating-point values: ");
    scanf("%d", &n);
```

```c
    float arr[n];
    printf("Enter %d numbers (between 0 and 1):\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%f", &arr[i]);
    }
    bucketSort(arr, n);
    printf("\nSorted array:\n");
    for (int i = 0; i < n; i++) {
        printf("%0.2f ", arr[i]);
    }
    printf("\n");
    return 0;
}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc Bucket_Sort.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter number of floating-point values: 6
Enter 6 numbers (between 0 and 1):
0.42 0.32 0.20 0.52 0.25 0.47

Sorted array:
0.20 0.25 0.32 0.42 0.47 0.52
```

Space Complexity: O(N^2)

Time Complexity: O(N^2)

Justification:

- This method uses a two dimensional Bucket Array to sort the floating point values. Hence the Space Complexity is O(N^2).
- In the worst case, all elements may fall into a single bucket and are sorted using insertion sort. Hence the Time Complexity is O(N^2).

Question 7: Write a C program to perform Breadth First Search (BFS) on a graph using an adjacency matrix.

CODE:

```c
//Write a C program to perform Breadth First Search (BFS) on a graph using
an adjacency matrix.

// CH.SC.U4CSE24142 - SANTHOSH A
```

```c
#include <stdio.h>
int graph[10][10], visited[10];
int n;
void bfs(int start) {
    int queue[10], front = 0, rear = 0;
    queue[rear++] = start;
    visited[start] = 1;
    printf("BFS Traversal: ");
    while (front < rear) {
        int v = queue[front++];
        printf("%d ", v);
        for (int i = 0; i < n; i++) {
            if (graph[v][i] == 1 && visited[i] == 0) {
                queue[rear++] = i;
                visited[i] = 1;
            }
        }
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++) {
        visited[i] = 0;
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    printf("Enter starting vertex: ");
    scanf("%d", &start);
```

```
    bfs(start);

    return 0;

}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc BFS.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>a
CH.SC.U4CSE24142 - SANTHOSH A
Enter number of vertices: 5
Enter adjacency matrix:
0 1 1 0 0
1 0 0 1 0
1 0 0 1 1
0 1 1 0 1
0 0 1 1 0
Enter starting vertex: 0
BFS Traversal: 0 1 2 3 4
```

Space Complexity: O(V)

Time Complexity: O(V+E)

Justification:

- The queue can store upto V vertices in the worst case. Hence the Space Complexity is O(V).
- BFS visits every vertex once and checks every edge once while exploring adjacency lists. Hence the Time Complexity is O(V+E).

Question 8: Write a C program to perform Depth First Search (DFS) on a graph using an adjacency matrix.

CODE:

```
// Write a C program to perform Depth First Search (DFS) on a graph using
an adjacency matrix.

// CH.SC.U4CSE24142 - SANTHOSH A

#include <stdio.h>

int graph[10][10], visited[10];

int n;

void dfs(int v) {

    visited[v] = 1;

    printf("%d ", v);

    for (int i = 0; i < n; i++) {

        if (graph[v][i] == 1 && visited[i] == 0) {

            dfs(i);

        }
```

```
    }
}
int main() {
    printf("CH.SC.U4CSE24142 - SANTHOSH A\n");
    int start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (int i = 0; i < n; i++) {
        visited[i] = 0;
        for (int j = 0; j < n; j++) {
            scanf("%d", &graph[i][j]);
        }
    }
    printf("Enter starting vertex: ");
    scanf("%d", &start);
    printf("DFS Traversal: ");
    dfs(start);
    return 0;
}
```

OUTPUT:

```
D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>gcc DFS.c

D:\AVV CHENNAI\Semester 4\Design and Analysis of Algorithms\Lab Activities\Week 2>A
CH.SC.U4CSE24142 - SANTHOSH A
Enter number of vertices: 4
Enter adjacency matrix:
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter starting vertex: 0
DFS Traversal: 0 1 3 2
```

Space Complexity: O(V)

Time Complexity: O(V+E)

Justification:

- The recursion stack can grow upto V levels and the visited [ ] array stores V entries. Hence the Space Complexity is O(V).
- DFS recursively explores every vertex and inspects all edges exactly once through the adjacency list. Hence the Time Complexity is O(V+E).