NAME:SHESHANK
ROLL NO: CH.SC.U4CSE24125

1. BUBBLE SORT:-
   CODE:

```c
#include <stdio.h>
int main() {
int a[50], n, i, j, temp;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter elements:\n");
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for (i = 0; i < n - 1; i++) {
for (j = 0; j < n - i - 1; j++) {
if (a[j] > a[j + 1]) {
temp = a[j];
a[j] = a[j + 1];
a[j + 1] = temp;}}}
printf("Sorted array:\n");
for (i = 0; i < n; i++)
printf("%d ", a[i]);
return 0;
}
```

   OUTPUT:

```
amma@amma22:~/Documents/125$ gcc -o bubblesort bubblesort.c
amma@amma22:~/Documents/125$ ./bubblesort
Enter number of elements: 4
Enter elements:
3 9 1 7
Sorted array:
1 3 7 9 amma@amma22:~/Documents/125$ gcc -o insertionsort in
```

2. INSERTION SORT:

CODE:

```c
#include <stdio.h>
int main() {
int a[50], n, i, j, key;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter elements:\n");
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for (i = 1; i < n; i++) {
key = a[i];
j = i - 1;
while (j >= 0 && a[j] > key) {
a[j + 1] = a[j];
j--;}
a[j + 1] = key;}
printf("Sorted array:\n");
for (i = 0; i < n; i++)
printf("%d ", a[i]);
return 0;
}
```

OUTPUT:

```
amma@amma22:~/Documents/125$ gcc -o insertionsort insertionsort.c
amma@amma22:~/Documents/125$ ./insertionsort
Enter number of elements: 5
Enter elements:
1 9 7 3 6
Sorted array:
1 3 6 7 9 amma@amma22:~/Documents/125$
```

3. SELECTION SORT:

CODE:

```c
#include <stdio.h>
int main() {
int a[50], n, i, j, min, temp;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter elements:\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[j] < a[min])
min = j;}
temp = a[i];
a[i] = a[min];
a[min] = temp;}
printf("Sorted array:\n");
for (i = 0; i < n; i++)
printf("%d ", a[i]);
return 0;
}
```

OUTPUT:

```
amma@amma22:~/Documents/125$ gcc -o selectionsort selectionsort.c
amma@amma22:~/Documents/125$ ./selectionsort
Enter number of elements: 4
Enter elements:
8 4 1 7
Sorted array:
1 4 7 8 amma@amma22:~/Documents/125$
```

4. HEAP SORT:

CODE:

```c
#include <stdio.h>
void heapify(int a[], int n, int i) {
int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
int temp;
if (left < n && a[left] > a[largest])
largest = left;
if (right < n && a[right] > a[largest])
largest = right;
if (largest != i) {
temp = a[i];
a[i] = a[largest];
a[largest] = temp;
heapify(a, n, largest);}}
int main() {
int a[50], n, i, temp;
printf("Enter number of elements: ");
scanf("%d", &n);
printf("Enter elements:\n");
for (i = 0; i < n; i++)
scanf("%d", &a[i]);
for (i = n / 2 - 1; i >= 0; i--)
heapify(a, n, i);
for (i = n - 1; i > 0; i--) {
temp = a[0];
a[0] = a[i];
a[i] = temp;
heapify(a, i, 0);}
printf("Sorted array:\n");
for (i = 0; i < n; i++)
printf("%d ", a[i]);
return 0;}
```

OUTPUT:

```
amma@amma22:~/Documents/125$ gcc -o HEAPSORT HEAPSORT.c
amma@amma22:~/Documents/125$ ./HEAPSORT
Enter number of elements: 6
Enter elements:
4 2 9 1 7 5
Sorted array:
1 2 4 5 7 9 amma@amma22:~/Documents/125$
```

5. BUCKET SORT:
CODE:

```c
#include <stdio.h>
#include <stdlib.h>
void bucketSort(float arr[], int n)
{
    int i, j;
    int bucketCount = n;
    float **buckets = (float **)malloc(bucketCount * sizeof(float *));
    int *count = (int *)calloc(bucketCount, sizeof(int));
    for (i = 0; i < bucketCount; i++)
        buckets[i] = (float *)malloc(n * sizeof(float));
    for (i = 0; i < n; i++)
    {
        int index = bucketCount * arr[i];
        buckets[index][count[index]++] = arr[i];
    }
    for (i = 0; i < bucketCount; i++)
    {
        for (j = 1; j < count[i]; j++)
        {
            float key = buckets[i][j];
            int k = j - 1;
            while (k >= 0 && buckets[i][k] > key)
            {
                buckets[i][k + 1] = buckets[i][k];
                k--;
            }
            buckets[i][k + 1] = key;
        }
    }
    int idx = 0;
    for (i = 0; i < bucketCount; i++)
    {
```

```
    {
        for (j = 0; j < count[i]; j++)
        {
            arr[idx++] = buckets[i][j];
        }
    }
    for (i = 0; i < bucketCount; i++)
    free(buckets[i]);
    free(buckets);
    free(count);
}
int main()
{
    float arr[] = {0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51};
    int n = sizeof(arr) / sizeof(arr[0]);
    bucketSort(arr, n);
    printf("Sorted array:\n");
    for (int i = 0; i < n; i++)
        printf("%.2f ", arr[i]);
    return 0;
}
```

OUTPUT:

```
Sorted array:
0.23 0.25 0.32 0.42 0.47 0.51 0.52

=== Code Execution Successful ===
```

6. BFS:

CODE:

```c
#include <stdio.h>
int queue[20], front = -1, rear = -1;
int visited[20];
int n;
int graph[20][20];
void enqueue(int v)
{
    if (rear == n - 1)
        return;
    if (front == -1)
        front = 0;
    queue[++rear] = v;
}
int dequeue()
{
    return queue[front++];
}
int isEmpty()
{
    return front > rear || front == -1;
}
void bfs(int start)
{
    int i, v;
    enqueue(start);
    visited[start] = 1;
    while (!isEmpty())
    {
        v = dequeue();
        printf("%d ", v);
        for (i = 0; i < n; i++)
        {
```

```c
            if (graph[v][i] == 1 && visited[i] == 0)
            {
                enqueue(i);
                visited[i] = 1;
            }
        }
    }
}
int main()
{
    int i, j, start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    for (i = 0; i < n; i++)
        visited[i] = 0;
    printf("Enter starting vertex: ");
    scanf("%d", &start);
    printf("BFS Traversal: ");
    bfs(start);
    return 0;
}
```

OUTPUT:

```
Enter number of vertices: 4
Enter adjacency matrix:
1 2 3 4
2 3 3 4
1 2 3 4
1 2 3 6
Enter starting vertex: 3
BFS Traversal: 3 0

=== Code Execution Successful ===
```

7. DFS:

CODE:

```c
#include <stdio.h>
int visited[20];
int n;
int graph[20][20];
void dfs(int v)
{
    int i;
    visited[v] = 1;
    printf("%d ", v);

    for (i = 0; i < n; i++)
    {
        if (graph[v][i] == 1 && visited[i] == 0)
        {
            dfs(i);
        }
    }
}
int main()
{
    int i, j, start;
    printf("Enter number of vertices: ");
    scanf("%d", &n);
    printf("Enter adjacency matrix:\n");
    for (i = 0; i < n; i++)
        for (j = 0; j < n; j++)
            scanf("%d", &graph[i][j]);
    for (i = 0; i < n; i++)
        visited[i] = 0;
    printf("Enter starting vertex: ");
    scanf("%d", &start);
    printf("DFS Traversal: ");
    dfs(start);
    return 0;
}
```

OUTPUT:

```
Enter number of vertices: 3
Enter adjacency matrix:
1 2 3
3 4 5
6 7 8
Enter starting vertex: 3
DFS Traversal: 3

=== Code Execution Successful ===
```