

NAME:SHESHANK
ROLL NO: CH.SC.U4CSE24125

1. BUBBLE SORT:-

CODE:

```
1 #include <stdio.h>
2 int main() {
3     int a[50], n, i, j, temp;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     printf("Enter elements:\n");
7     for (i = 0; i < n; i++)
8         scanf("%d", &a[i]);
9     for (i = 0; i < n - 1; i++) {
10        for (j = 0; j < n - i - 1; j++) {
11            if (a[j] > a[j + 1]) {
12                temp = a[j];
13                a[j] = a[j + 1];
14                a[j + 1] = temp;}}
15        printf("Sorted array:\n");
16        for (i = 0; i < n; i++)
17            printf("%d ", a[i]);
18    return 0;
19 }
```

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:

```
1 #include <stdio.h>
2 int main() {
3     int a[50], n, i, j, key;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     printf("Enter elements:\n");
7     for (i = 0; i < n; i++)
8         scanf("%d", &a[i]);
9     for (i = 1; i < n; i++) {
10        key = a[i];
11        j = i - 1;
12        while (j >= 0 && a[j] > key) {
13            a[j + 1] = a[j];
14            j--;
15        }
16        a[j + 1] = key;
17    }
18    printf("Sorted array:\n");
19    for (i = 0; i < n; i++)
20        printf("%d ", a[i]);
21    return 0;
22 }
```

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:

```
1 #include <stdio.h>
2 int main() {
3     int a[50], n, i, j, min, temp;
4     printf("Enter number of elements: ");
5     scanf("%d", &n);
6     printf("Enter elements:\n");
7     for (i = 0; i < n; i++)
8         scanf("%d", &a[i]);
9     for (i = 0; i < n - 1; i++) {
10        min = i;
11        for (j = i + 1; j < n; j++) {
12            if (a[j] < a[min])
13                min = j;}
14        temp = a[i];
15        a[i] = a[min];
16        a[min] = temp;}
17     printf("Sorted array:\n");
18     for (i = 0; i < n; i++)
19         printf("%d ", a[i]);
20     return 0;
21 }
```

OUTPUT:

```
amma@amma22:~/Documents/125$ gcc -o selectionsort selectionsrt.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:

```
1 #include <stdio.h>
2 void heapify(int a[], int n, int i) {
3     int largest = i;
4     int left = 2 * i + 1;
5     int right = 2 * i + 2;
6     int temp;
7     if (left < n && a[left] > a[largest])
8         largest = left;
9     if (right < n && a[right] > a[largest])
10        largest = right;
11    if (largest != i) {
12        temp = a[i];
13        a[i] = a[largest];
14        a[largest] = temp;
15        heapify(a, n, largest);}}
16 int main() {
17     int a[50], n, i, temp;
18     printf("Enter number of elements: ");
19     scanf("%d", &n);
20     printf("Enter elements:\n");
21     for (i = 0; i < n; i++)
22         scanf("%d", &a[i]);
23     for (i = n / 2 - 1; i >= 0; i--)
24         heapify(a, n, i);
25     for (i = n - 1; i > 0; i--) {
26         temp = a[0];
27         a[0] = a[i];
28         a[i] = temp;
29         heapify(a, i, 0);}
30     printf("Sorted array:\n");
31     for (i = 0; i < n; i++)
32         printf("%d ", a[i]);
33     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:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void bucketSort(float arr[], int n)
4 {
5     int i, j;
6     int bucketCount = n;
7     float **buckets = (float **)malloc(bucketCount * sizeof(float *));
8     int *count = (int *)calloc(bucketCount, sizeof(int));
9     for (i = 0; i < bucketCount; i++)
10        buckets[i] = (float *)malloc(n * sizeof(float));
11     for (i = 0; i < n; i++)
12     {
13         int index = bucketCount * arr[i];
14         buckets[index][count[index]++] = arr[i];
15     }
16     for (i = 0; i < bucketCount; i++)
17     {
18         for (j = 1; j < count[i]; j++)
19         {
20             float key = buckets[i][j];
21             int k = j - 1;
22             while (k >= 0 && buckets[i][k] > key)
23             {
24                 buckets[i][k + 1] = buckets[i][k];
25                 k--;
26             }
27             buckets[i][k + 1] = key;
28         }
29     }
30     int idx = 0;
31     for (i = 0; i < bucketCount; i++)
32     {
```

```
32     {
33         for (j = 0; j < count[i]; j++)
34     {
35         arr[idx++] = buckets[i][j];
36     }
37 }
38 for (i = 0; i < bucketCount; i++)
39 free(buckets[i]);
40 free(buckets);
41 free(count);
42 }
43 int main()
44 {
45     float arr[] = {0.42, 0.32, 0.23, 0.52, 0.25, 0.47, 0.51};
46     int n = sizeof(arr) / sizeof(arr[0]);
47     bucketSort(arr, n);
48     printf("Sorted array:\n");
49     for (int i = 0; i < n; i++)
50     {
51         printf("%.2f ", arr[i]);
52     }
53 }
```

OUTPUT:

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

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

6. BFS:

CODE:

```
1 #include <stdio.h>
2 int queue[20], front = -1, rear = -1;
3 int visited[20];
4 int n;
5 int graph[20][20];
6 void enqueue(int v)
7 {
8     if (rear == n - 1)
9         return;
10    if (front == -1)
11        front = 0;
12    queue[++rear] = v;
13 }
14 int dequeue()
15 {
16     return queue[front++];
17 }
18 int isEmpty()
19 {
20     return front > rear || front == -1;
21 }
22 void bfs(int start)
23 {
24     int i, v;
25     enqueue(start);
26     visited[start] = 1;
27     while (!isEmpty())
28     {
29         v = dequeue();
30         printf("%d ", v);
31         for (i = 0; i < n; i++)
32             if
```

```
33         if (graph[v][i] == 1 && visited[i] == 0)
34         {
35             enqueue(i);
36             visited[i] = 1;
37         }
38     }
39 }
40 }
41 int main()
42 {
43     int i, j, start;
44     printf("Enter number of vertices: ");
45     scanf("%d", &n);
46     printf("Enter adjacency matrix:\n");
47     for (i = 0; i < n; i++)
48         for (j = 0; j < n; j++)
49             scanf("%d", &graph[i][j]);
50     for (i = 0; i < n; i++)
51         visited[i] = 0;
52     printf("Enter starting vertex: ");
53     scanf("%d", &start);
54     printf("BFS Traversal: ");
55     bfs(start);
56     return 0;
57 }
```

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:

```
1 #include <stdio.h>
2 int visited[20];
3 int n;
4 int graph[20][20];
5 void dfs(int v)
6 {
7     int i;
8     visited[v] = 1;
9     printf("%d ", v);
10    for (i = 0; i < n; i++)
11    {
12        if (graph[v][i] == 1 && visited[i] == 0)
13        {
14            dfs(i);
15        }
16    }
17 }
18 }
19 int main()
20 {
21     int i, j, start;
22     printf("Enter number of vertices: ");
23     scanf("%d", &n);
24     printf("Enter adjacency matrix:\n");
25     for (i = 0; i < n; i++)
26     {
27         for (j = 0; j < n; j++)
28             scanf("%d", &graph[i][j]);
29     for (i = 0; i < n; i++)
30         visited[i] = 0;
31     printf("Enter starting vertex: ");
32     scanf("%d", &start);
33     printf("DFS Traversal: ");
34     dfs(start);
35 }
```

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 ==
```