

LAB ASSIGNMENT 7

Sorting algorithms

1. Write a program to implement bubble sort for sorting n elements in an array.

```
1  #include <stdio.h>
2- void bubbleSort(int arr[], int n) {
3      int i, j, temp;
4-      for (i = 0; i < n - 1; i++) {
5-          for (j = 0; j < n - i - 1; j++) {
6-              if (arr[j] > arr[j + 1]) {
7-                  // Swap elements
8-                  temp = arr[j];
9-                  arr[j] = arr[j + 1];
10-                 arr[j + 1] = temp;
11-             }
12-         }
13-     }
14 }

15- int main() {
16     int arr[100], n, i;
17     printf("Enter the number of elements (maximum 100): ");
18     scanf("%d", &n);
19     printf("Enter %d elements:\n", n);
20-     for (i = 0; i < n; i++) {
21         scanf("%d", &arr[i]);
22     }
23     printf("Unsorted array: \n");
24-     for (i = 0; i < n; i++) {
25         printf("%d ", arr[i]);
26     }
27     bubbleSort(arr, n);
28     printf("\nSorted array: \n");
29-     for (i = 0; i < n; i++) {
30         printf("%d ", arr[i]);
31     }
32     printf("\n");
33     return 0;
34 }
```

2. Write a program to implement Selection sort for sorting n elements in an array.

```
1 #include <stdio.h>
2- void selectionSort(int arr[], int n) {
3     int i, j, min_idx, temp;
4     // One by one move boundary of unsorted subarray
5-     for (i = 0; i < n - 1; i++) {
6         min_idx = i;
7-         for (j = i + 1; j < n; j++) {
8-             if (arr[j] < arr[min_idx]) {
9                 min_idx = j;
10            }
11        }
12-        if (min_idx != i) {
13            temp = arr[min_idx];
14            arr[min_idx] = arr[i];
15            arr[i] = temp;
16        }
17    }
18 }
19- int main() {
20     int arr[100], n, i;
21     printf("Enter the number of elements (maximum 100): ");
22     scanf("%d", &n);
23     printf("Enter %d elements:\n", n);
24-     for (i = 0; i < n; i++) {
25         scanf("%d", &arr[i]);
26     }
27     printf("Unsorted array: \n");
28-     for (i = 0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     selectionSort(arr, n);
32     printf("\nSorted array: \n");
33-     for (i = 0; i < n; i++) {
34         printf("%d ", arr[i]);
35     }
36     printf("\n");
37     return 0;
38 }
```

3. Write a program to implement Insertion sort for sorting n elements in an array.\

BHAVLEEN KAUR
102155010
3EIC-2

```
1  #include <stdio.h>
2- void insertionSort(int arr[], int n) {
3      int i, key, j;
4-      for (i = 1; i < n; i++) {
5          key = arr[i];
6          j = i - 1;
7-          while (j >= 0 && arr[j] > key) {
8              arr[j + 1] = arr[j];
9              j--;
10         }
11         arr[j + 1] = key;
12     }
13 }
14- int main() {
15     int arr[100], n, i;
16     printf("Enter the number of elements (maximum 100): ");
17     scanf("%d", &n);
18     printf("Enter %d elements:\n", n);
19-     for (i = 0; i < n; i++) {
20         scanf("%d", &arr[i]);
21     }
22     printf("Unsorted array: \n");
23-     for (i = 0; i < n; i++) {
24         printf("%d ", arr[i]);
25     }
26     insertionSort(arr, n);
27     printf("\nSorted array: \n");
28-     for (i = 0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     printf("\n");
32     return 0;
33 }
```

4. Write a program to implement Shell sort for sorting n elements in an array.

```
1 #include <stdio.h>
2 void shellSort(int arr[], int n) {
3     int gap = n / 2;
4     while (gap > 0) {
5         for (int i = gap; i < n; i++) {
6             int key = arr[i];
7             int j = i - gap;
8             // Shift elements of arr[j] to the right by 1 if arr[j] is greater
              than key
9             while (j >= 0 && arr[j] > key) {
10                arr[j + gap] = arr[j];
11                j -= gap;
12            }
13            arr[j + gap] = key;
14        }
15        // Reduce the gap for the next iteration
16        gap /= 2;
17    }
18 }
19 int main() {
20     int arr[100], n, i;
21     printf("Enter the number of elements (maximum 100): ");
22     scanf("%d", &n);
23     printf("Enter %d elements:\n", n);
24     for (i = 0; i < n; i++) {
25         scanf("%d", &arr[i]);
26     }
27     printf("Unsorted array: \n");
28     for (i = 0; i < n; i++) {
29         printf("%d ", arr[i]);
30     }
31     shellSort(arr, n);
32     printf("\nSorted array: \n");
33     for (i = 0; i < n; i++) {
34         printf("%d ", arr[i]);
35     }
36     printf("\n");
37     return 0;
38 }
```

5. Write a program to implement Radix sort for sorting n elements in an array.

```
1  #include <stdio.h>
2  #define MAX_DIGITS 5
3- void countingSort(int arr[], int n, int place) {
4      int output[n + 1];
5      int count[10] = {0};
6-   for (int i = 0; i < n; i++) {
7       count[(arr[i] / place) % 10]++;
8   }
9-   for (int i = 1; i < 10; i++) {
10      count[i] += count[i - 1];
11  }
12-   for (int i = n - 1; i >= 0; i--) {
13       output[count[(arr[i] / place) % 10] - 1] = arr[i];
14       count[(arr[i] / place) % 10]--;
15  }
16-   for (int i = 0; i < n; i++) {
17       arr[i] = output[i];
18  }
19 }
20- int getMax(int arr[], int n) {
21     int max = arr[0];
22-   for (int i = 1; i < n; i++) {
23       if (arr[i] > max) {
24           max = arr[i];
25       }
26   }
27   return max;
28 }
29- void radixSort(int arr[], int n) {
30     int max = getMax(arr, n);
31-   for (int place = 1; max / place > 0; place *= 10) {
32       countingSort(arr, n, place);
33   }
34 }
35- int main() {
36     int arr[100], n, i;
37     printf("Enter the number of elements (maximum 100): ");
38     scanf("%d", &n);
39     printf("Enter %d elements:\n", n);
40-   for (i = 0; i < n; i++) {
41       scanf("%d", &arr[i]);
42   }
43   printf("Unsorted array: \n");
44-   for (i = 0; i < n; i++) {
45       printf("%d ", arr[i]);
46   }
47   radixSort(arr, n);
48   printf("\nSorted array: \n");
49-   for (i = 0; i < n; i++) {
50       printf("%d ", arr[i]);
51   }
52   printf("\n");
53   return 0;
54 }
```

6. Write a program to implement Merge sort.

```
1  #include <stdio.h>
2- void merge(int arr[], int left, int mid, int right) {
3      int n1 = mid - left + 1;
4      int n2 = right - mid;
5      int leftArr[n1], rightArr[n2];
6-     for (int i = 0; i < n1; i++) {
7         leftArr[i] = arr[left + i];
8     }
9-     for (int j = 0; j < n2; j++) {
10        rightArr[j] = arr[mid + 1 + j];
11    }
12    int i = 0, j = 0, k = left;
13-    while (i < n1 && j < n2) {
14-        if (leftArr[i] <= rightArr[j]) {
15            arr[k] = leftArr[i];
16            i++;
17-        } else {
18            arr[k] = rightArr[j];
19            j++;
20        }
21        k++;
22    }
23-    while (i < n1) {
24        arr[k] = leftArr[i];
25        i++;
26        k++;
27    }
28-    while (j < n2) {
29        arr[k] = rightArr[j];
30        j++;
31        k++;
32    }
33 }
34- void mergeSort(int arr[], int left, int right) {
35-     if (left < right) {
36         // Find the middle point
37         int mid = left + (right - left) / 2;
38
39         // Sort first and second halves
40         mergeSort(arr, left, mid);
41         mergeSort(arr, mid + 1, right);
42
43         // Merge the sorted halves
44         merge(arr, left, mid, right);
45     }
46 }
47- int main() {
48     int arr[100], n, i;
49     printf("Enter the number of elements (maximum 100): ");
50     scanf("%d", &n);
51     printf("Enter %d elements:\n", n);
52-     for (i = 0; i < n; i++) {
53         scanf("%d", &arr[i]);
54     }
55     printf("Unsorted array: \n");
56-     for (i = 0; i < n; i++) {
57         printf("%d ", arr[i]);
58     }
59     mergeSort(arr, 0, n - 1);
60     printf("\nSorted array: \n");
61-     for (i = 0; i < n; i++) {
62         printf("%d ", arr[i]);
63     }
64     printf("\n");
65     return 0;
66 }
```

BHAVLEEN KAUR
102155010
3EIC-2

7. Write a program to implement Quick sort.

```
1 #include <stdio.h>
2 int partition(int arr[], int low, int high) {
3     int pivot = arr[high];
4     int i = (low - 1); // index of smaller element
5
6     for (int j = low; j <= high - 1; j++) {
7         // If current element is smaller than the pivot
8         if (arr[j] <= pivot) {
9             i++;
10            int temp = arr[i];
11            arr[i] = arr[j];
12            arr[j] = temp;
13        }
14    }
15    int temp = arr[i + 1];
16    arr[i + 1] = arr[high];
17    arr[high] = temp;
18    return (i + 1);
19 }
20 void quickSort(int arr[], int low, int high) {
21     if (low < high) {
22         // pi is partitioning index, arr[p] is now at right place
23         int pi = partition(arr, low, high);
24
25         // Recursively sort elements before and after partition
26         quickSort(arr, low, pi - 1);
27         quickSort(arr, pi + 1, high);
28     }
29 }
30 int main() {
31     int arr[100], n, i;
32     printf("Enter the number of elements (maximum 100): ");
33     scanf("%d", &n);
34     printf("Enter %d elements:\n", n);
35     for (i = 0; i < n; i++) {
36         scanf("%d", &arr[i]);
37     }
38     printf("Unsorted array: \n");
39     for (i = 0; i < n; i++) {
40         printf("%d ", arr[i]);
41     }
42     quickSort(arr, 0, n - 1);
43     printf("\nSorted array: \n");
44     for (i = 0; i < n; i++) {
45         printf("%d ", arr[i]);
46     }
47     printf("\n");
48     return 0;
49 }
```