

1. Write a c program for insertion sort.

```
main.c
1 #include <stdio.h>
2 void insertionSort(int array[], int n) {
3     for (int i = 1; i < n; i++) {
4         int key = array[i];
5         int j = i - 1;
6         while (j >= 0 && array[j] > key) {
7             array[j + 1] = array[j];
8             j = j - 1;
9         }
10        array[j + 1] = key;
11    }
12 }
13 void printArray(int array[], int size) {
14     for (int i = 0; i < size; i++) {
15         printf("%d ", array[i]);
16     }
17     printf("\n");
18 }
19 int main() {
20     int array[] = {12, 11, 13, 5, 6};
21     int n = sizeof(array) / sizeof(array[0]);
22     printf("Original array: \n");
23     printArray(array, n);
24     insertionSort(array, n);
25     printf("Sorted array: \n");
26     printArray(array, n);
27     return 0;
28 }
29
```

```
Output
/tmp/1D1DG3eT1u.o
Original array:
12 11 13 5 6
Sorted array:
5 6 11 12 13

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

2. Write a c program for merge sort.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 void merge(int array[], int left, int mid, int right) {
4     int i, j, k;
5     int n1 = mid - left + 1;
6     int n2 = right - mid;
7     int L[n1], R[n2];
8     for (i = 0; i < n1; i++)
9         L[i] = array[left + i];
10    for (j = 0; j < n2; j++)
11        R[j] = array[mid + 1 + j];
12    i = 0;
13    j = 0;
14    k = left;
15    while (i < n1 && j < n2) {
16        if (L[i] <= R[j]) {
17            array[k] = L[i];
18            i++;
19        } else {
20            array[k] = R[j];
21            j++;
22        }
23        k++;
24    }
25    while (i < n1) {
26        array[k] = L[i];
27        i++;
28        k++;
29    }
30    while (j < n2) {
31        array[k] = R[j];
32        j++;
33        k++;
34    }
}
```

```
/tmp/mKkb6zpasM.o
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13

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

```

27     i++;
28     k++;
29 }
30 while (j < n2) {
31     array[k] = R[j];
32     j++;
33     k++;
34 }
35 }
36 void mergeSort(int array[], int left, int right) {
37     if (left < right) {
38         int mid = left + (right - left) / 2;
39         mergeSort(array, left, mid);
40         mergeSort(array, mid + 1, right);
41         merge(array, left, mid, right);
42     }
43 }
44 void printArray(int array[], int size) {
45     for (int i = 0; i < size; i++) {
46         printf("%d ", array[i]);
47     }
48     printf("\n");
49 }
50 int main() {
51     int array[] = {12, 11, 13, 5, 6, 7};
52     int array_size = sizeof(array) / sizeof(array[0]);
53     printf("Original array: \n");
54     printArray(array, array_size);
55     mergeSort(array, 0, array_size - 1);
56     printf("Sorted array: \n");
57     printArray(array, array_size);
58     return 0;
59 }
60

```

```

/tmp/mKB6zpasw.o
Original array:
12 11 13 5 6 7
Sorted array:
5 6 7 11 12 13

=== Code Execution Successful ===

```

3. Write a c program for Radix sort.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 int getMax(int array[], int n) {
4     int max = array[0];
5     for (int i = 1; i < n; i++)
6         if (array[i] > max)
7             max = array[i];
8     return max;
9 }
10 void countingSort(int array[], int n, int exp) {
11     int output[n]; // Output array
12     int i, count[10] = {0};
13     for (i = 0; i < n; i++)
14         count[(array[i] / exp) % 10]++;
15     for (i = 1; i < 10; i++)
16         count[i] += count[i - 1];
17     for (i = n - 1; i >= 0; i--) {
18         output[count[(array[i] / exp) % 10] - 1] = array[i];
19         count[(array[i] / exp) % 10]--;
20     }
21     for (i = 0; i < n; i++)
22         array[i] = output[i];
23 }
24 void radixSort(int array[], int n) {
25     int m = getMax(array, n);
26     for (int exp = 1; m / exp > 0; exp *= 10)
27         countingSort(array, n, exp);
28 }
29 void printArray(int array[], int size) {
30     for (int i = 0; i < size; i++)
31         printf("%d ", array[i]);
32     printf("\n");
33 }
34 int main() {
35     int array[] = {170, 45, 75, 90, 802, 24, 2, 66};
36     int n = sizeof(array) / sizeof(array[0]);
37     printf("Original array: \n");
38     printArray(array, n);
39     radixSort(array, n);
40     printf("Sorted array: \n");
41     printArray(array, n);
42     return 0;
43 }
44

```

```

/tmp/BXuehdePou.o
Original array:
170 45 75 90 802 24 2 66
Sorted array:
2 24 45 66 75 90 170 802

=== Code Execution Successful ===

```