SEM-2 > DSA-ASS-11 > C 11_1_k_small_quicksort.c > ...

```c
/* Find the kth smallest element in an array using quicksort. */

#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int partition(int *arr, int low, int high)
{
    int pivot = arr[low], i = low + 1, j = high, temp;
    do
    {
        while (arr[i] <= pivot)
        {
            i++;
        }
        while (arr[j] > pivot)
        {
            j--;
        }
        if (i < j)
        {
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    } while (i <= j);
    temp = arr[j];
    arr[j] = arr[low];
    arr[low] = temp;
    return j;
}
```

```c
int quicksortksmall(int *arr, int low, int high, int k)
{
    int indexpos;
    if (low < high)
    {
        indexpos = partition(arr, low, high);
        if (k - 1 == indexpos)
        {
            return arr[k];
        }
        else if (k - 1 < indexpos)
        {
            quicksortksmall(arr, low, indexpos - 1, k);
        }
        else
        {
            quicksortksmall(arr, indexpos + 1, high, k);
        }
    }
}

int main()
{
    int ch;
    while (1)
    {
        printf("\nEnter your choice :--\n1 - Find kth smallest element using quick sort\n2 - Exit\nChoice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
        {
            int n, k;
            printf("\nEnter size of array : ");
            scanf("%d", &n);
            int arr[n];
            printf("Enter %d numbers : ", n);
            for (int _ = 0; _ < n; _++)
            {
                scanf("%d", &arr[_]);
            }
            printf("Enter k : ");
            scanf("%d", &k);
            printf("%d th smallest element : %d", k, quicksortksmall(arr, 0, n - 1, k));
            break;
        }
        case 2:
            exit(0);
        default:
            printf("INVALID INPUT - TRY AGAIN.\n");
        }
    }
    return 0;
}
```

```
Enter your choice :--
1 - Find kth smallest element using quick sort
2 - Exit
Choice : 1

Enter size of array : 10
Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9
Enter k : 2
2 th smallest element : 1

Enter your choice :--
1 - Find kth smallest element using quick sort
2 - Exit
Choice : 2
PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>
```

```c
/* Given an array of non-negative integers, implement the Counting Sort algorithm to sort the array in ascending order. */

#include <stdio.h>
#include <stdlib.h>

void display(int *arr, int n)
{
    for (int _ = 0; _ < n; _++)
    {
        printf("%d ", arr[_]);
    }
    printf("\n");
}

int maximum(int *arr, int n)
{
    int max = 0;
    for (int i = 0; i < n; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
    }
    return max;
}
```

```c
void countsort(int *arr, int n)
{
    int max = maximum(arr, n);
    int count[max + 1];
    for (int i = 0; i <= max; i++)
    {
        count[i] = 0;
    }
    for (int i = 0; i < n; i++)
    {
        count[arr[i]]++;
    }
    int icount = 0, imain = 0;
    while (icount <= max)
    {
        if (count[icount] > 0)
        {
            arr[imain++] = icount;
            count[icount]--;
        }
        else
        {
            icount++;
        }
    }
}

int main()
{
    int ch;
    while (1)
    {
        printf("\nEnter your choice :--\n1 - Count sort given array\n2 - Exit\nChoice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
        {
```

```c
        case 1:
        {
            int n;
            printf("\nEnter size of array : ");
            scanf("%d", &n);
            int arr[n];
            printf("Enter %d numbers : ", n);
            for (int _ = 0; _ < n; _++)
            {
                scanf("%d", &arr[_]);
            }
            countsort(arr, n);
            printf("Sorted Array : ");
            display(arr, n);
            break;
        }
        case 2:
            exit(0);
        default:
            printf("INVALID INPUT - TRY AGAIN.\n");
        }
    }
    return 0;
}
```

TERMINAL COMMENTS

```
Enter your choice :--
1 - Count sort given array
2 - Exit
Choice : 1

Enter size of array : 10
Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9
Sorted Array : 0 1 2 3 3 5 7 9 10 10

Enter your choice :--
1 - Count sort given array
2 - Exit
Choice : 2
PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>
```

SEM-2 > DSA-ASS-11 > C 11_3_radix_sort.c > main()

```c
/* Given an array of non-negative integers, implement the Radix Sort algorithm to sort the array in ascending order. */

#include <stdio.h>
#include <stdlib.h>

void display(int *arr, int n)
{
    for (int _ = 0; _ < n; _++)
    {
        printf("%d ", arr[_]);
    }
    printf("\n");
}

int maximum(int *arr, int n)
{
    int max = 0;
    for (int i = 0; i < n; i++)
    {
        if (arr[i] > max)
        {
            max = arr[i];
        }
    }
    return max;
}

```

```c
void countsort(int *arr, int n, int exp)
{
    int output[n];
    int i, count[10] = {0};
    for (i = 0; i < n; i++)
    {
        count[(arr[i] / exp) % 10]++;
    }
    for (i = 1; i < 10; i++)
    {
        count[i] += count[i - 1];
    }
    for (i = n - 1; i >= 0; i--)
    {
        output[count[(arr[i] / exp) % 10] - 1] = arr[i];
        count[(arr[i] / exp) % 10]--;
    }
    for (i = 0; i < n; i++)
    {
        arr[i] = output[i];
    }
}

void radixsort(int *arr, int n)
{
    int m = maximum(arr, n);
    for (int exp = 1; m / exp > 0; exp *= 10)
        countsort(arr, n, exp);
}

int main()
{
    int ch;
    while (1)
    {
        printf("\nEnter your choice :--\n1 - Radix sort given array\n2 - Exit\nChoice : ");
        scanf("%d", &ch);
        switch (ch)
        {
```

```c
        {
        case 1:
        {
            int n;
            printf("\nEnter size of array : ");
            scanf("%d", &n);
            int arr[n];
            printf("Enter %d numbers : ", n);
            for (int _ = 0; _ < n; _++)
            {
                scanf("%d", &arr[_]);
            }
            radixsort(arr, n);
            printf("Sorted Array : ");
            display(arr, n);
            break;
        }
        case 2:
            exit(0);
        default:
            printf("INVALID INPUT - TRY AGAIN.\n");
        }
    }
    return 0;
}
```

TERMINAL COMMENTS

```
Enter your choice :--
1 - Radix sort given array
2 - Exit
Choice : 1

Enter size of array : 10
Enter 10 numbers : 10 2 5 1 0 3 10 3 7 9
Sorted Array : 0 1 2 3 3 5 7 9 10 10

Enter your choice :--
1 - Radix sort given array
2 - Exit
Choice : 2
PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes>
```