insertion:

```c
#include <stdio.h>

void insertion_sort(int arr[], int n)
{
    int i, j, temp;
    for (i = 0; i < n; i++)
    {
        j = i;
        while (j > 0 && arr[j - 1] > arr[j])
        {
            temp = arr[j - 1];
            arr[j - 1] = arr[j];
            arr[j] = temp;
            j--;
        }
    }
    printf("After Using insertion sort:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    printf("\n");
}

int main()
{
    int n, i, arr[100];
    printf("Enter the number of Elements: ");
    scanf("%d", &n);
    printf("Array input: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    insertion_sort(arr, n);
    return 0;
}
```

SELECTION:

```c
#include <stdio.h>
#include <conio.h>

void selectionSort(int arr[], int n)
{
    int i, j, min, temp;
    for (i = 0; i < n - 1; i++)
    {
        min = i;
        for (j = i + 1; j < n; j++)
        {
            if (arr[j] < arr[min])
            {
                min = j;
            }
        }
        temp = arr[i];
        arr[i] = arr[min];
        arr[min] = temp;
    }
}

int main()
{
    int arr[100], n, i;
    clrscr();
    printf("Enter the number of Elements: ");
    scanf("%d", &n);
    printf("Array input: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    selectionSort(arr, n);
    printf("Resulted sorted array using selection sort approach:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    getch();
    return 0;
}
```

MERGE:

```
#include <stdio.h>
#include <conio.h>

void combine(int arr[], int low, int mid, int high)
{
    int i = low, j = mid + 1, k = 0;
    int temp[100];
    while (i <= mid && j <= high)
    {
        if (arr[i] <= arr[j])
        {
            temp[k++] = arr[i++];
        }
        else
        {
            temp[k++] = arr[j++];
        }
    }
    while (i <= mid)
    {
        temp[k++] = arr[i++];
    }
    while (j <= high)
    {
        temp[k++] = arr[j++];
    }
    for (i = 0; i < k; i++)
    {
        arr[low + i] = temp[i];
    }
}

void mergeSort(int arr[], int low, int high)
{
    int mid;
    if (low < high)
    {
        mid = (low + high) / 2;
        mergeSort(arr, low, mid);
        mergeSort(arr, mid + 1, high);
        combine(arr, low, mid, high);
    }
}

int main()
{
    int arr[100], n, i;
    clrscr();
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Array input: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    mergeSort(arr, 0, n - 1);
    printf("Resulted sorted array using merge sort approach:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    getch();
    return 0;
}
```

QUICK SORT:

```
#include <stdio.h>
#include <conio.h>

int partition(int arr[], int low, int high)
```

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

void quickSort(int arr[], int low, int high)
{
    int index;
    if (low < high)
    {
        index = partition(arr, low, high);
        quickSort(arr, low, index - 1);
        quickSort(arr, index + 1, high);
    }
}

int main()
{
    int arr[100], n, i;
    clrscr();
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    printf("Array input: ");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    quickSort(arr, 0, n - 1);
    printf("Resulted sorted array using Quick sort approach:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    getch();
    return 0;
}
```

FRACTIONAL KNAPSACK:

```c
#include <stdio.h>
#include <conio.h>

struct item {
    int value, weight;
};

void sortItems(struct item arr[], int n) {
    int i, j;
    for (i = 0; i < n - 1; i++) {
        for (j = 0; j < n - i - 1; j++) {
            double ratio1 = (double)arr[j].value / arr[j].weight;
            double ratio2 = (double)arr[j + 1].value / arr[j + 1].weight;
            if (ratio1 < ratio2) {
                struct item temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

double fractionalKnapsack(int W, struct item arr[], int n) {
    int i;
    double maxValue = 0.0;
    sortItems(arr, n);
    for (i = 0; i < n; i++) {
        if (W >= arr[i].weight) {
            W -= arr[i].weight;
            maxValue += arr[i].value;
        } else {
            maxValue += ((double)arr[i].value / arr[i].weight) * W;
            break;
        }
    }
    return maxValue;
}

int main() {
    int n, W, i;
    struct item arr[100];
    clrscr();
    printf("Enter number of items: ");
    scanf("%d", &n);
    printf("Enter total weight capacity: ");
    scanf("%d", &W);
    printf("Enter values of items:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i].value);
    }
    printf("Enter weights of items:\n");
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i].weight);
    }
    printf("Maximum value in knapsack = %.2f\n", fractionalKnapsack(W, 
arr, n));
    getch();
    return 0;
}
```

LONGEST COMMON SUBSEQUENCE:

```
#include <stdio.h>
#include <string.h>
#include <conio.h>

void lcs(char s1[], char s2[]) {
    int n = strlen(s1);
    int m = strlen(s2);
    int dp[101][101];
    int i, j;

    for (i = 0; i <= n; i++) {
        for (j = 0; j <= m; j++) {
            if (i == 0 || j == 0)
                dp[i][j] = 0;
            else if (s1[i - 1] == s2[j - 1])
                dp[i][j] = 1 + dp[i - 1][j - 1];
            else
                dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : 
dp[i][j - 1];
        }
    }

    int lcs_length = dp[n][m];
    char lcs_str[101];
    lcs_str[lcs_length] = '\0';

    i = n;
    j = m;
    int index = lcs_length - 1;

    while (i > 0 && j > 0) {
        if (s1[i - 1] == s2[j - 1]) {
            lcs_str[index--] = s1[i - 1];
            i--;
            j--;
        } else if (dp[i - 1][j] > dp[i][j - 1]) {
            i--;
        } else {
            j--;
        }
    }
    printf("The Length of Longest Common Subsequence is %d\n",
lcs_length);
    printf("The Longest Common Subsequence is: %s\n", lcs_str);
}

int main() {
    char s1[100], s2[100];
    clrscr();
    printf("Enter first string: ");
    scanf("%s", s1);
    printf("Enter second string: ");
    scanf("%s", s2);
    lcs(s1, s2);
    getch();
    return 0;
}
```

NQUEENS PROBLEM:

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int a[30], count = 0;

int place(int pos) {
    int i;
    for (i = 1; i < pos; i++) {
        if ((a[i] == a[pos]) || (abs(a[i] - a[pos]) == abs(i - pos)))
            return 0;
    }
    return 1;
}

void print_sol(int n) {
    int i, j;
    count++;
    printf("\n\nSolution #%d:\n", count);
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n; j++) {
            if (a[i] == j)
                printf("Q\t");
            else
                printf("*\t");
        }
        printf("\n");
    }
}

void queen(int n) {
    int k = 1;
    a[k] = 0;
    while (k != 0) {
        a[k] = a[k] + 1;
        while ((a[k] <= n) && !place(k))
            a[k]++;
        if (a[k] <= n) {
            if (k == n)
                print_sol(n);
            else {
                k++;
                a[k] = 0;
            }
        } else {
            k--;
        }
    }
}

int main() {
    int n;
    clrscr();
    printf("Enter the number of Queens: ");
    scanf("%d", &n);
    queen(n);
    getch();
    return 0;
}
```

NAIVE MATCHING:

```
#include <stdio.h>
#include <conio.h>
#include <string.h>

int main() {
    char txt[] = "tutorialsPointisthebestplatformforprogrammers";
    char pat[] = "a";
    int M = strlen(pat);
    int N = strlen(txt);
    int i, j;

    clrscr();

    for (i = 0; i <= N - M; i++) {
        for (j = 0; j < M; j++) {
            if (txt[i + j] != pat[j])
                break;
        }
        if (j == M)
            printf("Pattern matches at index %d\n", i);
    }

    getch();
    return 0;
}
```