TUGAS JOBSHEET 8

Dosen pengampu :

Randi Proska Sandra, M.Sc

Disusun Oleh:

Rendi Aigo Brandon

NIM : 23343082

PROGRAM STUDI

INFORMATIKA(NK) DEPARTEMEN

ELEKTRONIKA FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

SOURCE CODE

```
#include <stdio.h>

// Bubble Sort
void bubbleSort(int arr[], int n) {
    int i, j;
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Tukar elemen jika urutan salah
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Insertion Sort
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;

        // Geser elemen yang lebih besar dari key ke posisi setelahnya
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
```

```
int main() {
    int arr1[] = {64, 25, 12, 22, 11};
    int n1 = sizeof(arr1)/sizeof(arr1[0]);

    printf("Array sebelum diurutkan (Bubble Sort):\n");
    for (int i=0; i < n1; i++) {
        printf("%d ", arr1[i]);
    }
    printf("\n");

    bubbleSort(arr1, n1);

    printf("Array setelah diurutkan (Bubble Sort):\n");
    for (int i=0; i < n1; i++) {
        printf("%d ", arr1[i]);
    }
    printf("\n");

    int arr2[] = {64, 25, 12, 22, 11};
    int n2 = sizeof(arr2)/sizeof(arr2[0]);

    printf("Array sebelum diurutkan (Insertion Sort):\n");
    for (int i=0; i < n2; i++) {
        printf("%d ", arr2[i]);
    }
    printf("\n");

    insertionSort(arr2, n2);

    printf("Array setelah diurutkan (Insertion Sort):\n");
    for (int i=0; i < n2; i++) {
        printf("%d ", arr2[i]);
    }
    printf("\n");
```

```
    return 0;
}
```

PENJELASAN

1. Bubble Sort:
    - Pada awalnya, dalam fungsi bubbleSort, terdapat dua loop bersarang yang mengiterasi melalui seluruh array.
    - Di dalam loop pertama (for (i = 0; i < n-1; i++)), loop ini akan berjalan sebanyak n-1 kali, di mana n adalah jumlah elemen dalam array.
    - Di dalam loop kedua (for (j = 0; j < n-i-1; j++)), loop ini akan berjalan sebanyak n-i-1 kali pada setiap iterasi loop pertama, dimana i adalah indeks iterasi loop pertama. Ini dilakukan untuk memastikan bahwa elemen terbesar akan berada di akhir array setelah setiap iterasi loop pertama.
    - Setiap pasangan elemen berturut-turut akan dibandingkan. Jika elemen kedua lebih kecil dari elemen pertama, kedua elemen tersebut akan ditukar.
    - Proses pertukaran akan terus berlanjut hingga seluruh array terurut.
2. Insertion Sort:
    - Dalam fungsi insertionSort, setiap elemen dari indeks kedua hingga terakhir dalam array akan dipindahkan ke posisi yang benar di dalam bagian yang sudah diurutkan dari array.
    - Pada setiap iterasi dari loop pertama (for (i = 1; i < n; i++)), nilai key akan disimpan sementara untuk elemen saat ini yang akan dimasukkan ke dalam bagian yang sudah diurutkan.
    - Di dalam loop kedua (while (j >= 0 && arr[j] > key)), iterasi mundur dilakukan untuk mencari posisi yang tepat untuk memasukkan elemen saat ini (key) di dalam bagian yang sudah diurutkan. Jika nilai elemen di posisi saat ini (j) lebih besar dari key, elemen tersebut akan digeser ke kanan untuk memberikan ruang bagi key.
    - Proses ini akan berlanjut hingga semua elemen dalam array dipindahkan ke posisi yang benar di dalam bagian yang sudah diurutkan.