PRAKTIKUM ALGORITMA STRUKTUR DATA

“MERGE & QUICK SORT - PRAKTIKUM M12”

DOSEN :

Umi Sa’adah S.Kom., M.Kom. (197404162000032003)

PENYUSUN :

Mohammad Ihsan Septiano Firdaus (3125600041)

PROGRAM STUDI SARJANA TERAPAN

TEKNIK INFORMATIKA

POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

Praktikum 12.1. SORTING (MERGE & QUICK)

1. Implementasikan algoritma sorting metode:
    a. Selection
    b. Insertion
    c. Bubble
    d. Shell
    e. Merge
    f. Quick
   - Data berupa *array of int* sebanyak 10 elemen yang diinisialisasi
   - Tampilkan data sebelum dan sesudah diurutkan

2. Modifikasi soal no 1. Berikan input dengan jumlah elemen >= 30000 yang digenerate secara random → disable fungsi tampil()

3. Tambahkan perhitungan waktu komputasinya. Tampilkan performance dari masing-masing metode tersebut untuk mengurutkan data yang sama.

→Terbentuk program menu sorting terhadap *array of int* sbb :
    a. memiliki opsi 6 metode
    b. Jumlah data diinputkan, minimal 30000 → digenerate secara random
    c. Pilihan mode_urut secara asc atau desc
    d. Perhitungan waktu komputasi untuk setiap metode

Program n = 10

```c
#include <stdio.h>
#include <windows.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

void insertionSort(int a[], int b, int n);
void selectionSort(int a[], int b, int n);
void bubbleSort(int a[], int b, int n);
void shellSort(int a[], int b, int n);
void mSortRek(int a[], int b , int l, int r);
void mSort(int a[], int med, int l, int r, int b);
void quickSort(int a[], int b, int l, int r);
void tukar(int *a, int *b);
void copy(int a[], int b[]);
void generate(int a[], int n);
void tampilkan(int a[]);
int n;

int main() {
    int sort, alur;
```

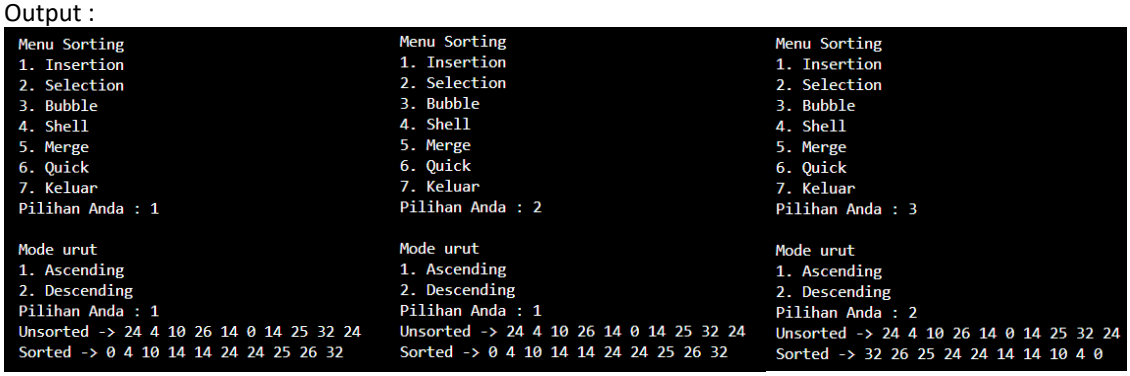
```
    //printf("Berapa jumlah data (maks 100000) ? ");
    //scanf("%d", &n);
    n = 10;
    printf("jml = %d\n", n);

    int backup[n], a[n];
    srand(time(NULL));
    generate(backup, n);

    do {
          printf("\nMenu Sorting\n1. Insertion\n2. Selection\n3. Bubble\n4.
Shell\n5. Merge\n6. Quick\n7. Keluar\nPilihan Anda : ");
        scanf("%d", &sort);

        if (sort == 7) {
            break;
        }
        if (sort < 1 || sort > 7) {
            printf("Masukkan Opsi yang benar!\n");
            continue;
        }

        printf("\nMode urut\n1. Ascending\n2. Descending\nPilihan Anda : ");
        scanf("%d", &alur);

        copy(a, backup);
        printf("Unsorted -> ");
        tampilkan(a);
        // clock_t start = clock();
        switch(sort) {
            case 1:
                insertionSort(a, alur, n);
                break;
            case 2:
                selectionSort(a, alur, n);
                break;
            case 3:
                bubbleSort(a, alur, n);
                break;
            case 4:
                shellSort(a, alur, n);
                break;
            case 5:
                mSortRek(a, alur,0, n-1 );
                break;
            case 6:
                quickSort(a, alur, 0, n-1);
                break;
```

```
        }
        // clock_t end = clock();

        // double waktu = (double)(end - start) / CLOCKS_PER_SEC;
        // printf("Durasi = %g\n", waktu);
        // printf("-----------------------------------\n");
        printf("Sorted -> ");
        tampilkan(a);
        Sleep(3000);
    } while (sort != 7);

    return 0;
}

void tampilkan(int a[]) {
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

void generate(int a[], int n) {
    for(int i=0; i<n; i++)
        a[i] = rand()/1000;
}

void copy(int a[], int b[]) {
    for (int i = 0; i < n; i++)
        a[i] = b[i];
}

void insertionSort(int a[], int b, int n) {
    for (int i = 1; i < n; i++) {
        int key = a[i];
        int j=i-1;
        if(b == 1)
            while ( j >= 0 && a[j] > key ) {
                a[j+1] = a[j];
                j--; }
        if(b == 2)
            while ( j >= 0 && a[j] < key ) {
                a[j+1] = a[j];
                j--; }
        a[j+1]=key; }
}

void selectionSort(int a[], int b, int n) {
    for (int i = 0; i < n; i++) {
```

```
        int j = i+1, min = i;
        while (j < n) {
            if(b == 1)
                if (a[j] < a[min]) min= j;
            if(b == 2)
                if (a[j] > a[min]) min= j;
            j++ ; }
        tukar (&a[i], &a[min]);
    }
}

void bubbleSort(int a[], int b, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] > a[j + 1]);
            if (b == 2) kondisi = (a[j] < a[j + 1]);
            if (kondisi) {
                tukar(&a[j], &a[j + 1]);
            }
        }
    }
}

void shellSort(int a[], int b, int n) {
    for (int gap = n / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i++) {
            for (int j = i - gap; j >= 0; j -= gap) {
                int kondisi = 0;
                if (b == 1) kondisi = (a[j] > a[j + gap]);
                if (b == 2) kondisi = (a[j] < a[j + gap]);
                if (kondisi) {
                    tukar(&a[j], &a[j + gap]);
                } else {
                    break;
                }
            }
        }
    }
}

void mSortRek(int a[], int b , int l, int r) {
    if (l<r) {
        int med = (l+r) / 2;
        mSortRek (a, b, l, med);
        mSortRek (a, b, med+1, r);
        mSort (a, med, l, r, b);
    }
```

```
}

void mSort(int a[], int med, int l, int r, int b) {
    int n1 = med - l + 1;
    int n2 = r - med;
    int f[n1], g[n2];
    for(int i = 0; i < n1; i++) f[i]  = a[l + i];
    for(int i = 0; i < n2; i++) g[i]  = a[med + 1 + i];
    int i = 0, j = 0, m = l;
    while (i < n1 && j < n2) {
int kondisi = 0;
        if (b == 1) kondisi = (f[i] <= g[j]);
        if (b == 2) kondisi = (f[i] >= g[j]);

        if (kondisi) a[m++] = f[i++];
        else a[m++] = g[j++];
    }
    while (i < n1) a[m++] = f[i++];
    while (j < n2) a[m++] = g[j++];
}

void quickSort(int a[], int b, int l, int r) {
    if (l < r) {
        int pivot = a[l];
        int i = l;
        for (int j = l + 1; j <= r; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] < pivot);
            if (b == 2) kondisi = (a[j] > pivot);

            if (kondisi) {
                i++;
                tukar(&a[i], &a[j]);
            }
        }

        tukar(&a[l], &a[i]);
        int pi = i;
        quickSort(a, b, l, pi - 1);
        quickSort(a, b, pi + 1, r);
    }
}


void tukar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Output :



Program N>=20000 dan bandingkan performa

```c
#include <stdio.h>
#include <windows.h>
#include <math.h>
#include <stdlib.h>
#include <time.h>

void insertionSort(int a[], int b, int n);
void selectionSort(int a[], int b, int n);
void bubbleSort(int a[], int b, int n);
void shellSort(int a[], int b, int n);
void mSortRek(int a[], int b , int l, int r);
void mSort(int a[], int med, int l, int r, int b);
void quickSort(int a[], int b, int l, int r);
void tukar(int *a, int *b);
void copy(int a[], int b[]);
void generate(int a[], int n);
void tampilkan(int a[]);
void compare(int a[]);
int n;

int main() {
    int sort, alur;
    printf("Berapa jumlah data (maks 100000) ? ");
    scanf("%d", &n);
    printf("jml = %d\n", n);

    int backup[n], a[n];
    srand(time(NULL));
    generate(backup, n);

    do {
          printf("\nMenu Sorting\n1. Insertion\n2. Selection\n3. Bubble\n4.
Shell\n5. Merge\n6. Quick\n7. Banding Performa\n8. Keluar\nPilihan Anda : ");
        scanf("%d", &sort);
```

```
        if (sort == 8) {
            break;
        }
        if (sort < 1 || sort > 7) {
            printf("Masukkan Opsi yang benar!\n");
            continue;
        }
        if (sort > 0 || sort < 7)
         { printf("\nMode urut\n1. Ascending\n2. Descending\nPilihan Anda :
");
        scanf("%d", &alur);}

        copy(a, backup);
        // printf("Unsorted -> ");
        // tampilkan(a);
        clock_t start = clock();
        switch(sort) {
            case 1:
                insertionSort(a, alur, n);
                break;
            case 2:
                selectionSort(a, alur, n);
                break;
            case 3:
                bubbleSort(a, alur, n);
                break;
            case 4:
                shellSort(a, alur, n);
                break;
            case 5:
                mSortRek(a, alur,0, n-1 );
                break;
            case 6:
                quickSort(a, alur, 0, n-1);
                break;
            case 7:
                compare(a);
                continue;
        }
        clock_t end = clock();

        double waktu = (double)(end - start) / CLOCKS_PER_SEC;
        printf("Durasi = %g\n", waktu);
        printf("-------------------------------------\n");
        // printf("Sorted -> ");
        // tampilkan(a);
        Sleep(3000);
    } while (sort != 8);
```

```
    return 0;
}

void tampilkan(int a[]) {
    for (int i = 0; i < n; i++) {
        printf("%d ", a[i]);
    }
    printf("\n");
}

void generate(int a[], int n) {
    for(int i=0; i<n; i++)
        a[i] = rand()/1000;
}

void copy(int a[], int b[]) {
    for (int i = 0; i < n; i++)
        a[i] = b[i];
}

void insertionSort(int a[], int b, int n) {
    for (int i = 1; i < n; i++) {
        int key = a[i];
        int j=i-1;
        if(b == 1)
            while ( j >= 0 && a[j] > key ) {
                a[j+1] = a[j];
                j--; }
        if(b == 2)
            while ( j >= 0 && a[j] < key ) {
                a[j+1] = a[j];
                j--; }
        a[j+1]=key; }
}

void selectionSort(int a[], int b, int n) {
    for (int i = 0; i < n; i++) {
        int j = i+1, min = i;
        while (j < n) {
            if(b == 1)
                if (a[j] < a[min]) min= j;
            if(b == 2)
                if (a[j] > a[min]) min= j;
            j++ ; }
        tukar (&a[i], &a[min]);
    }
}
```

```
void bubbleSort(int a[], int b, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] > a[j + 1]);
            if (b == 2) kondisi = (a[j] < a[j + 1]);
            if (kondisi) {
                tukar(&a[j], &a[j + 1]);
            }
        }
    }
}

void shellSort(int a[], int b, int n) {
    for (int gap = n / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i++) {
            for (int j = i - gap; j >= 0; j -= gap) {
                int kondisi = 0;
                if (b == 1) kondisi = (a[j] > a[j + gap]);
                if (b == 2) kondisi = (a[j] < a[j + gap]);
                if (kondisi) {
                    tukar(&a[j], &a[j + gap]);
                } else {
                    break;
                }
            }
        }
    }
}

void mSortRek(int a[], int b , int l, int r) {
    if (l<r) {
        int med = (l+r) / 2;
        mSortRek (a, b, l, med);
        mSortRek (a, b, med+1, r);
        mSort (a, med, l, r, b);
    }
}

void mSort(int a[], int med, int l, int r, int b) {
    int n1 = med - l + 1;
    int n2 = r - med;
    int f[n1], g[n2];
    for(int i = 0; i < n1; i++) f[i]  = a[l + i];
    for(int i = 0; i < n2; i++) g[i]  = a[med + 1 + i];
    int i = 0, j = 0, m = l;
    while (i < n1 && j < n2) {
```

```
int kondisi = 0;
        if (b == 1) kondisi = (f[i] <= g[j]);
        if (b == 2) kondisi = (f[i] >= g[j]);

        if (kondisi) a[m++] = f[i++];
        else a[m++] = g[j++];
    }
    while (i < n1) a[m++] = f[i++];
    while (j < n2) a[m++] = g[j++];
}

void quickSort(int a[], int b, int l, int r) {
    if (l < r) {
        int pivot = a[l];
        int i = l;
        for (int j = l + 1; j <= r; j++) {
            int kondisi = 0;
            if (b == 1) kondisi = (a[j] < pivot);
            if (b == 2) kondisi = (a[j] > pivot);

            if (kondisi) {
                i++;
                tukar(&a[i], &a[j]);
            }
        }

        tukar(&a[l], &a[i]);
        int pi = i;
        quickSort(a, b, l, pi - 1);
        quickSort(a, b, pi + 1, r);
    }
}


void tukar(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
void compare(int a[]) {
    int temp[n];
    clock_t start, end;
    double waktu;

    printf("\n--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---\n");

    // 1. Insertion Sort
    printf("\nInsertion Sort\n");
    copy(temp, a); start = clock(); insertionSort(temp, 1, n); end = clock();
```

```
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); insertionSort(temp, 2, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 2. Selection Sort
    printf("\nSelection Sort\n");
    copy(temp, a); start = clock(); selectionSort(temp, 1, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); selectionSort(temp, 2, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 3. Bubble Sort
    printf("\nBubble Sort\n");
    copy(temp, a); start = clock(); bubbleSort(temp, 1, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); bubbleSort(temp, 2, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 4. Shell Sort
    printf("\nShell Sort\n");
    copy(temp, a); start = clock(); shellSort(temp, 1, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); shellSort(temp, 2, n); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 5. Merge Sort
    printf("\nMerge Sort\n");
    copy(temp, a); start = clock(); mSortRek(temp, 1, 0, n - 1); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

    copy(temp, a); start = clock(); mSortRek(temp, 2, 0, n - 1); end = clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    // 6. Quick Sort
```

```
    printf("\nQuick Sort\n");
     copy(temp, a); start = clock(); quickSort(temp, 1, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Ascending  = %g detik\n", waktu);

     copy(temp, a); start = clock(); quickSort(temp, 2, 0, n - 1); end =
clock();
    waktu = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Descending = %g detik\n", waktu);

    printf("\n----------------------------------------\n");
    Sleep(5000);
}
```

Output :

```
Menu Sorting
1. Insertion
2. Selection
3. Bubble
4. Shell
5. Merge
6. Quick
7. Banding Performa
8. Keluar
Pilihan Anda : 7

--- HASIL PERBANDINGAN WAKTU EKSEKUSI ---

Insertion Sort
Ascending  = 2.909 detik
Descending = 2.721 detik

Selection Sort
Ascending  = 4.572 detik
Descending = 4.573 detik

Bubble Sort
Ascending  = 22.191 detik
Descending = 23.218 detik

Shell Sort
Ascending  = 0.01 detik
Descending = 0.013 detik

Merge Sort
Ascending  = 0.011 detik
Descending = 0.013 detik

Quick Sort
Ascending  = 0.149 detik
Descending = 0.145 detik
```

Praktikum 12.2.
SORTING (QUICK & MERGE)

1. Bukalah file pengurutan yang sudah ada untuk melakukan pengurutan data struct siswa yang memiliki 3 field sebagai berikut:
   - NO, bertipe int
   - Nama, bertipe string
   - Nilai, bertipe int

   Diawali dengan menu menginputkan sejumlah N buah data siswa dengan tampilan sebagai berikut:

```
Masukkan jumlah data: _
No:  _
Nama: _
Nilai: _
```

   Selanjutnya, pada program utama di mana terdapat pilihan untuk melakukan pengurutan data tambahkan metode pengurutan Merge sort dan Quick sort. Tampilan menu yang diharapkan:

```
MENU METODE SORTING
   1. Insertion Sort
   2. Selection Sort
   3. Bubble Sort
   4. Shell Sort
   5. Quick Sort Rekursif
   6. Merge Sort Rekursif
   7. Keluar
Pilihan anda [1-6]: _

Pengurutan yang dipilih:
1.  Ascending
2.  Descending
Pilihan anda [1/2]: _

Diurutkan berdasarkan:
1. No
2. Nama
3. Nilai
Pilihan anda [1/2/3]: _
```

   Dengan kata lain, program yang dibuat dapat melakukan sorting array of struct dengan karakteristik sebagai berikut:
   a. Menerima input jumlah data yang akan diinputkan
   b. Fungsi yang menangani pemasukan data satu persatu (No, Nama, dan Nilai siswa)
   c. Memiliki opsi 6 metode sorting
   d. Pilihan mode_urut secara ascending atau descending
   e. Pilihan pengurutan berdasarkan (sort by) : No, Nama, atau Nilai
   f. Penampilan output yang telah diurutkan.

1

Program

```c
#include <stdio.h>
#include <windows.h>
#include <string.h>

typedef struct {
    int NO;
    char Nama[50];
    int Nilai;
} Siswa;

void insertionSort(Siswa a[], int b, int sortBy, int n);
void selectionSort(Siswa a[], int b, int sortBy, int n);
void bubbleSort(Siswa a[], int b, int sortBy, int n);
```

```
void shellSort(Siswa a[], int b, int sortBy, int n);
void mSortRek(Siswa a[], int b, int sortBy, int l, int r);
void mSort(Siswa a[], int med, int l, int r, int b, int sortBy);
void quickSort(Siswa a[], int b, int sortBy, int l, int r);

void tukar(Siswa *a, Siswa *b);
void copy(Siswa a[], Siswa b[], int n);
void tampilkan(Siswa a[], int n);

int main() {
    int n, sort, alur, sortBy;

    printf("Masukkan jumlah data: ");
    scanf("%d", &n);

    Siswa backup[n], a[n];

    for(int i = 0; i < n; i++) {
        printf("No    : ");
        scanf("%d", &backup[i].NO);
        printf("Nama  : ");
        scanf(" %[^\n]s", backup[i].Nama);
        printf("Nilai : ");
        scanf("%d", &backup[i].Nilai);
        printf("\n");
    }

    do {
        printf("\nMENU METODE SORTING\n");
       printf("1. Insertion Sort\n2. Selection Sort\n3. Bubble Sort\n4. Shell
Sort\n5. Quick Sort Rekursif\n6. Merge Sort Rekursif\n7. Keluar\n");
        printf("Pilihan anda [1-7]: ");
        scanf("%d", &sort);

        if (sort == 7) {
            break;
        }
        if (sort < 1 || sort > 7) {
            printf("Masukkan Opsi yang benar!\n");
            continue;
        }

                printf("\nPengurutan   yang   dipilih:\n1.   Ascending\n2.
Descending\nPilihan anda [1/2]: ");
        scanf("%d", &alur);

         printf("\nDiurutkan berdasarkan:\n1. No\n2. Nama\n3. Nilai\nPilihan
anda [1/2/3]: ");
```

```
        scanf("%d", &sortBy);

        copy(a, backup, n);

        switch(sort) {
            case 1: insertionSort(a, alur, sortBy, n); break;
            case 2: selectionSort(a, alur, sortBy, n); break;
            case 3: bubbleSort(a, alur, sortBy, n); break;
            case 4: shellSort(a, alur, sortBy, n); break;
            case 5: quickSort(a, alur, sortBy, 0, n-1); break;
            case 6: mSortRek(a, alur, sortBy, 0, n-1); break;
        }

        printf("\n--- Output Setelah Diurutkan ---\n");
        tampilkan(a, n);
        printf("--------------------------------\n");

    } while (sort != 7);

    return 0;
}

void tampilkan(Siswa a[], int n) {
    printf("%-5s %-20s %-5s\n", "NO", "NAMA", "NILAI");
    for (int i = 0; i < n; i++) {
        printf("%-5d %-20s %-5d\n", a[i].NO, a[i].Nama, a[i].Nilai);
    }
}

void copy(Siswa a[], Siswa b[], int n) {
    for (int i = 0; i < n; i++)
        a[i] = b[i];
}

void tukar(Siswa *a, Siswa *b) {
    Siswa temp = *a;
    *a = *b;
    *b = temp;
}

void insertionSort(Siswa a[], int b, int sortBy, int n) {
    for (int i = 1; i < n; i++) {
        Siswa key = a[i];
        int j = i - 1;

        while (j >= 0) {
            int kondisi = 0;
            if (b == 1) {
```

```
                if (sortBy == 1) kondisi = (a[j].NO > key.NO);
                 if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, key.Nama) >
0);
                if (sortBy == 3) kondisi = (a[j].Nilai > key.Nilai);
            }
            if (b == 2) {
                if (sortBy == 1) kondisi = (a[j].NO < key.NO);
                 if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, key.Nama) <
0);
                if (sortBy == 3) kondisi = (a[j].Nilai < key.Nilai);
            }

            if (kondisi) {
                a[j + 1] = a[j];
                j--;
            } else {
                break;
            }
        }
        a[j + 1] = key;
    }
}

void selectionSort(Siswa a[], int b, int sortBy, int n) {
    for (int i = 0; i < n - 1; i++) {
        int targetIdx = i;
        for (int j = i + 1; j < n; j++) {
            int kondisi = 0;
            if (b == 1) {
                if (sortBy == 1) kondisi = (a[j].NO < a[targetIdx].NO);
                        if  (sortBy  ==  2)  kondisi  =  (strcmpi(a[j].Nama,
a[targetIdx].Nama) < 0);
                if (sortBy == 3) kondisi = (a[j].Nilai < a[targetIdx].Nilai);
            }
            if (b == 2) {
                if (sortBy == 1) kondisi = (a[j].NO > a[targetIdx].NO);
                        if  (sortBy  ==  2)  kondisi  =  (strcmpi(a[j].Nama,
a[targetIdx].Nama) > 0);
                if (sortBy == 3) kondisi = (a[j].Nilai > a[targetIdx].Nilai);
            }

            if (kondisi) {
                targetIdx = j;
            }
        }
        tukar(&a[targetIdx], &a[i]);
    }
}
```

```
void bubbleSort(Siswa a[], int b, int sortBy, int n) {
    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            int kondisi = 0;
            if (b == 1) {
                if (sortBy == 1) kondisi = (a[j].NO > a[j + 1].NO);
               if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, a[j + 1].Nama) >
0);
                if (sortBy == 3) kondisi = (a[j].Nilai > a[j + 1].Nilai);
            }
            if (b == 2) {
                if (sortBy == 1) kondisi = (a[j].NO < a[j + 1].NO);
               if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, a[j + 1].Nama)
< 0);
                if (sortBy == 3) kondisi = (a[j].Nilai < a[j + 1].Nilai);
            }

            if (kondisi) {
                tukar(&a[j], &a[j + 1]);
            }
        }
    }
}

void shellSort(Siswa a[], int b, int sortBy, int n) {
    for (int gap = n / 2; gap > 0; gap /= 2) {
        for (int i = gap; i < n; i++) {
            for (int j = i - gap; j >= 0; j -= gap) {
                int kondisi = 0;
                if (b == 1) {
                    if (sortBy == 1) kondisi = (a[j].NO > a[j + gap].NO);
                      if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, a[j +
gap].Nama) > 0);
                   if (sortBy == 3) kondisi = (a[j].Nilai > a[j + gap].Nilai);
                }
                if (b == 2) {
                    if (sortBy == 1) kondisi = (a[j].NO < a[j + gap].NO);
                      if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, a[j +
gap].Nama) < 0);
                   if (sortBy == 3) kondisi = (a[j].Nilai < a[j + gap].Nilai);
                }

                if (kondisi) {
                    tukar(&a[j], &a[j + gap]);
                } else {
                    break;
                }
```

```
            }
        }
    }
}

void quickSort(Siswa a[], int b, int sortBy, int l, int r) {
    if (l < r) {
        Siswa pivot = a[l];
        int i = l;
        for (int j = l + 1; j <= r; j++) {
            int kondisi = 0;
            if (b == 1) {
                if (sortBy == 1) kondisi = (a[j].NO < pivot.NO);
                if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, pivot.Nama) <
0);
                if (sortBy == 3) kondisi = (a[j].Nilai < pivot.Nilai);
            }
            if (b == 2) {
                if (sortBy == 1) kondisi = (a[j].NO > pivot.NO);
                if (sortBy == 2) kondisi = (strcmpi(a[j].Nama, pivot.Nama) >
0);
                if (sortBy == 3) kondisi = (a[j].Nilai > pivot.Nilai);
            }

            if (kondisi) {
                i++;
                tukar(&a[i], &a[j]);
            }
        }
        tukar(&a[l], &a[i]);
        int pi = i;
        quickSort(a, b, sortBy, l, pi - 1);
        quickSort(a, b, sortBy, pi + 1, r);
    }
}

void mSortRek(Siswa a[], int b, int sortBy, int l, int r) {
    if (l < r) {
        int med = (l + r) / 2;
        mSortRek(a, b, sortBy, l, med);
        mSortRek(a, b, sortBy, med + 1, r);
        mSort(a, med, l, r, b, sortBy);
    }
}

void mSort(Siswa a[], int med, int l, int r, int b, int sortBy) {
    int n1 = med - l + 1;
    int n2 = r - med;
```

```
    Siswa f[n1], g[n2];

    for(int i = 0; i < n1; i++) f[i] = a[l + i];
    for(int i = 0; i < n2; i++) g[i] = a[med + 1 + i];

    int i = 0, j = 0, m = l;
    while (i < n1 && j < n2) {
        int kondisi = 0;
        if (b == 1) {
            if (sortBy == 1) kondisi = (f[i].NO <= g[j].NO);
            if (sortBy == 2) kondisi = (strcmpi(f[i].Nama, g[j].Nama) <= 0);
            if (sortBy == 3) kondisi = (f[i].Nilai <= g[j].Nilai);
        }
        if (b == 2) {
            if (sortBy == 1) kondisi = (f[i].NO >= g[j].NO);
            if (sortBy == 2) kondisi = (strcmpi(f[i].Nama, g[j].Nama) >= 0);
            if (sortBy == 3) kondisi = (f[i].Nilai >= g[j].Nilai);
        }

        if (kondisi) a[m++] = f[i++];
        else a[m++] = g[j++];
    }
    while (i < n1) a[m++] = f[i++];
    while (j < n2) a[m++] = g[j++];
}
```

Output

```
Masukkan jumlah data: 5
No    : 5
Nama  : Ihsan Sept
Nilai : 99

No    : 67
Nama  : Natsa
Nilai : 91

No    : 2
Nama  : Anakin Skywalker
Nilai : 88

No    : 23
Nama  : Natasya Meisyawahyunis
Nilai : 99

No    : 4
Nama  : Arthur Morgan
Nilai : 45
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble Sort
4. Shell Sort
5. Quick Sort Rekursif
6. Merge Sort Rekursif
7. Keluar
Pilihan anda [1-7]: 1

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 1

Diurutkan berdasarkan:
1. No
2. Nama
3. Nilai
Pilihan anda [1/2/3]: 1

--- Output Setelah Diurutkan ---
NO    NAMA                 NILAI
2     Anakin Skywalker     88
4     Arthur Morgan        45
5     Ihsan Sept           99
23    Natasya Meisyawahyunis 99
67    Natsa                91
--------------------------------
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble Sort
4. Shell Sort
5. Quick Sort Rekursif
6. Merge Sort Rekursif
7. Keluar
Pilihan anda [1-7]: 2

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 2

Diurutkan berdasarkan:
1. No
2. Nama
3. Nilai
Pilihan anda [1/2/3]: 2

--- Output Setelah Diurutkan ---
NO    NAMA                 NILAI
67    Natsa                91
23    Natasya Meisyawahyunis 99
5     Ihsan Sept           99
4     Arthur Morgan        45
2     Anakin Skywalker     88
--------------------------------
```

```
MENU METODE SORTING
1. Insertion Sort
2. Selection Sort
3. Bubble Sort
4. Shell Sort
5. Quick Sort Rekursif
6. Merge Sort Rekursif
7. Keluar
Pilihan anda [1-7]: 3

Pengurutan yang dipilih:
1. Ascending
2. Descending
Pilihan anda [1/2]: 2

Diurutkan berdasarkan:
1. No
2. Nama
3. Nilai
Pilihan anda [1/2/3]: 3

--- Output Setelah Diurutkan ---
NO    NAMA                 NILAI
5     Ihsan Sept           99
23    Natasya Meisyawahyunis 99
67    Natsa                91
2     Anakin Skywalker     88
4     Arthur Morgan        45
--------------------------------
```