

Nama:M.AGUNG BALYA

NIM:1203230095

Kelas:IF 03-01

Code

```
Go Run Terminal Help < -> Februari
Welcome Fghijk.c Sad.c Tugas-Struck1.c yu.c Tugas-Struck2#1.c Tugas-Struck2#2.c
Week 4 > C Tugas-Struck2#1.c > min_swaps_to_sort(char *, int)
5 int min_swaps_to_sort(char *cards, int n) {
25     if (card_values[sorted_cards[j]] > card_values[sorted_cards[j + 1]]) {
26         char temp = sorted_cards[j];
27         sorted_cards[j] = sorted_cards[j + 1];
28         sorted_cards[j + 1] = temp;
29
30         printf("Urutan kartu setelah pertukaran ke-%d: %s\n", (i * (n - 1)) + j + 1, sorted_cards);
31     }
32 }
33
34
35 int swaps = 0;
36 for (int i = 0; i < n; i++) {
37     if (cards[i] != sorted_cards[i]) {
38         swaps++;
39     }
40 }
41
42 int min_swaps = swaps / 2;
43
44 free(sorted_cards);
45 return min_swaps;
46 }
47
48 int main() {
49     int n;
50     scanf("%d", &n);
51     char cards[n + 1];
52     scanf("%s", cards);
53     printf("Jumlah minimal langkah pertukaran: %d\n", min_swaps_to_sort(cards, n));
54
55     return 0;
56 }
57
```

```
Go Run Terminal Help < > Februari
Welcome C Fghijk.c C Sad.c C Tugas-Struck1.c C yu.c C Tugas-Struck2#1.c C Tugas-Struck2#2.c

Week 4 > C Tugas-Struck2#1.c > min_swaps_to_sort(char *, int)
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int min_swaps_to_sort(char *cards, int n) {
6      int card_values[128];
7      card_values['1'] = 1;
8      card_values['2'] = 2;
9      card_values['3'] = 3;
10     card_values['4'] = 4;
11     card_values['5'] = 5;
12     card_values['6'] = 6;
13     card_values['7'] = 7;
14     card_values['8'] = 8;
15     card_values['9'] = 9;
16     card_values['J'] = 11;
17     card_values['Q'] = 12;
18     card_values['K'] = 13;
19
20     char *sorted_cards = malloc(n * sizeof(char));
21     memcpy(sorted_cards, cards, n * sizeof(char));
22
23     for (int i = 0; i < n; i++) {
24         for (int j = 0; j < n - 1; j++) {
25             if (card_values[sorted_cards[j]] > card_values[sorted_cards[j + 1]]) {
26                 char temp = sorted_cards[j];
27                 sorted_cards[j] = sorted_cards[j + 1];
28                 sorted_cards[j + 1] = temp;
29
30                 printf("Urutan kartu setelah pertukaran ke-%d: %s\n", (i * (n - 1)) + j + 1, sorted_cards);
31             }
32         }
33     }
34
35     int swaps = 0;
36     for (int i = 0; i < n; i++) {
37         if (cards[i] != sorted_cards[i]) {
```

Output

```
Week 4 > C Tugas-Struck2#1.c > min_swaps_to_sort(char *cards, int n)
5 int min_swaps_to_sort(char *cards, int n)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\agung\Februari> cd "c:\agung\Februari\Week 4"
9
J K Q
Urutan kartu setelah pertukaran ke-4: J
Urutan kartu setelah pertukaran ke-6: J
Urutan kartu setelah pertukaran ke-7: J
Urutan kartu setelah pertukaran ke-8: J
Urutan kartu setelah pertukaran ke-11: J
Urutan kartu setelah pertukaran ke-14: J
Urutan kartu setelah pertukaran ke-18: Jë
Urutan kartu setelah pertukaran ke-25: ëJ
Jumlah minimal langkah pertukaran: 3
PS C:\agung\Februari\Week 4>
```

Penjelasan

```
int min_swaps_to_sort(char *cards, int n) {
    int card_values[128];
    card_values['1'] = 1;
    card_values['2'] = 2;
    card_values['3'] = 3;
    card_values['4'] = 4;
    card_values['5'] = 5;
    card_values['6'] = 6;
    card_values['7'] = 7;
    card_values['8'] = 8;
    card_values['9'] = 9;
    card_values['J'] = 11;
    card_values['Q'] = 12;
    card_values['K'] = 13;
```

fungsi 'min\_swaps\_to\_sort' mengonversi nilai-nilai kartu ke menjadi nilai numerik

int 'card\_values[128];' Membuat array 'card\_values' dengan ukuran 128, yang akan digunakan untuk menyimpan nilai numerik dari kartu-kartu.

'card\_values['1'] = 1;' = Nilai kartu angka 1

'card\_values['2'] = 2;' = Nilai kartu angka 2

'card\_values['3'] = 3;' = Nilai kartu angka 3  
'card\_values['4'] = 4;' = Nilai kartu angka 4  
'card\_values['5'] = 5;' = Nilai kartu angka 5  
'card\_values['6'] = 6;' = Nilai kartu angka 6  
'card\_values['7'] = 7;' = Nilai kartu angka 7  
'card\_values['8'] = 8;' = Nilai kartu angka 8  
'card\_values['9'] = 9;' = Nilai kartu angka 9  
'card\_values['J'] = 11;' = Nilai kartu angka 11  
'card\_values['Q'] = 12;' = Nilai kartu angka 12  
'card\_values['K'] = 13;' = Nilai kartu angka 13

```
char *sorted_cards = malloc(n * sizeof(char));
memcpy(sorted_cards, cards, n * sizeof(char));

for (int i = 0; i < n; i++) {
    for (int j = 0; j < n - 1; j++) {
        if (card_values[sorted_cards[j]] > card_values[sorted_cards[j +
1]]) {
            char temp = sorted_cards[j];
            sorted_cards[j] = sorted_cards[j + 1];
            sorted_cards[j + 1] = temp;

            printf("Urutan kartu setelah pertukaran ke-%d: %s\n", (i * (n
- 1)) + j + 1, sorted_cards);
        }
    }
}
```

Bubble kode diatas merupakan proses dari pengurutan kartu menggunakan Bubble Sort

char \*sorted\_cards = malloc(n \* sizeof(char));: Membuat sebuah array dinamis baru bernama 'sorted\_cards' dengan ukuran sepanjang n (jumlah kartu) dan tipe data char.

memcpy(sorted\_cards, cards, n \* sizeof(char));: Menggunakan fungsi 'memcpy' untuk menyalin isi dari array cards (kartu-kartu yang belum diurutkan) ke dalam array sorted\_cards.

for (int i = 0; i < n; i++) {: Memulai loop pertama yang akan melakukan iterasi sebanyak n kali, mewakili jumlah kartu yang belum diurutkan.

for (int j = 0; j < n - 1; j++) {: Memulai loop kedua yang akan melakukan iterasi sebanyak n - 1 kali. Ini adalah jumlah perbandingan antara dua kartu yang berdekatan dalam proses Bubble Sort.

if (card\_values[sorted\_cards[j]] > card\_values[sorted\_cards[j + 1]]) {: Memeriksa apakah nilai kartu pada posisi j lebih besar dari nilai kartu pada posisi j + 1. Jika ya, maka kartu tersebut perlu ditukar posisinya agar sesuai dengan urutan yang diinginkan.

`char temp = sorted_cards[j]; sorted_cards[j] = sorted_cards[j + 1]; sorted_cards[j + 1] = temp;`  
Melakukan pertukaran posisi antara kartu pada posisi `j` dengan kartu pada posisi `j + 1`, menggunakan sebuah variabel sementara `temp` untuk menyimpan nilai sementara dari kartu.

`printf("Urutan kartu setelah pertukaran ke-%d: %s\n", (i * (n - 1)) + j + 1, sorted_cards);`  
Menampilkan urutan kartu setelah terjadi pertukaran. Ini membantu untuk melihat bagaimana urutan kartu berubah pada setiap langkah pertukaran.

```
int swaps = 0;
for (int i = 0; i < n; i++) {
    if (cards[i] != sorted_cards[i]) {
        swaps++;
    }
}

int min_swaps = swaps / 2;

free(sorted_cards);
return min_swaps;
}
```

`int swaps = 0;`, Inisialisasi variabel `swaps`

`for (int i = 0; i < n; i++)`, Memulai loop untuk iterasi melalui setiap kartu.

`if (cards[i] != sorted_cards[i])`, Memeriksa apakah kartu pada posisi `i` dalam array `cards` tidak sama dengan kartu pada posisi `i` dalam array `sorted_cards`.

`swaps++`, Menambahkan nilai counter `swaps` jika terjadi perbedaan antara kartu

`free(sorted_cards);`, Melakukan dealokasi memori yang dialokasikan sebelumnya

`return min_swaps;`, Mengembalikan nilai jumlah minimal langkah pertukaran yang telah dihitung.

```
int main() {
    int n;
    scanf("%d", &n);
    char cards[n + 1];
    scanf("%s", cards);
    printf("Jumlah minimal langkah pertukaran: %d\n", min_swaps_to_sort(cards,
n));

    return 0;
}
```

`char cards[n + 1];`, Mendeklarasikan array `cards` dengan panjang sebanyak `n + 1`

`printf("Jumlah minimal langkah pertukaran: %d\n", min_swaps_to_sort(cards, n));`, Memanggil fungsi '`min_swaps_to_sort`' dengan argumen `cards` (array nilai-nilai kartu) dan `n` (jumlah kartu), dan mencetak hasilnya ke layar. Ini akan mencetak jumlah minimal langkah